Drone Fault Detection Based on Failure Mode Databases

Subjects: Engineering, Aerospace

Contributor: Defei Hou , Qingran Su , Yi Song , Yongfeng Yin

Drones have become an indispensable component of modern equipment systems, and as the number of components in these systems continues to increase, so does their complexity. Consequently, the requirements for system quality—particularly reliability, maintainability, and functionality—are also becoming increasingly stringent. Sensor data fusion, fault detection, fault-tolerant estimation, and fault-tolerant control are all important means of ensuring UAV safety. Among these methods, researchers tend to place particular emphasis on fault detection, which can be broadly divided into two categories: model-based approaches and data-driven approache.

drone fault detection failure mode databases runtime verification safety model-based approaches

1. Introduction

The traditional way to detect drone failures is through model-based approaches that use mathematical models to analyze UAV safety. These models can be further subdivided into qualitative or quantitative methods (**Table 1**). However, because model-based approaches often focus only on specific components, they may not always detect failures in other parts of the system. This means that, if an unmanned aerial system suddenly fails due to its complexity, resulting in higher costs, multiple model pairs may need to be deployed. Therefore, in this research, faults at the system level are mainly studied. Building a fault mode database using a large amount of fault data can facilitate the analysis of fault modes and fault propagation processes at the system level. The analysis process can help fault diagnosis personnel quickly locate the problem and save diagnostic time. At present, the research on failure mode databases lacks depth. Drone systems are complex systems with multiple tasks and configurable elements. There are both common and unique failure modes among different systems. The evaluation center where the researchers of the current research have a large body of failure data. However, due to the lack of organization and induction of these data, there has been no in-depth analysis of the mechanisms of common faults. How to use historical fault data to generate fault mode databases and utilize these databases to better achieve fault detection is a problem worth studying.

Table 1. Model-based approaches to Fault Detection.

Authors	Year	Fault Location	Used Methods
D'Amato et al. [1]	2021	sensor	sensor fusion algorithm based on particle filter
Fu et al. [2]	2021	sensor, actuator	adaptive observer to realize sensor and brake fault detection
Maqsood et al. [3]	2021	sensor	sensor fault detection and isolation method for quadrotor aircraft
Miao et al. [4]	2021	sensor	adaptive nonlinear proportional integral (PI) observer
Nejati et al. [5]	2021	actuator	three-level central difference Kalman filter (RThSCDKF)
Sun et al. [6]	2021	pitot tube	two identical synthetic air data systems
Cao et al. [7]	2022	actuator	interval observer and extended state observer
Gai et al. [8]	2022	elevator, event-triggered intervals	dynamic event-triggered Hi/H∞ optimization method
Lin et al. [9]	2022	sensor	crash probability density (CPD)

Another typical method used for drone fault detection is the data-based method. In recent years, the reduction in the difficulty of data collection and analysis has driven the development of the field of data analysis. With the support of various high-speed processors, the means of mining required information from data have become increasingly powerful. Similarly, data-based methods can be divided into two categories: quantitative and qualitative. Quantitative methods include those based on statistical theory and neural networks. Qualitative methods typically include expert systems, fuzzy logic, pattern recognition, and qualitative trend analysis. **Table 2** lists some typical studies, including the location of fault detection and the methods used.

Table 2. Data-driven approaches to Fault Detection.

Authors	Year	Fault Location	Used Methods
Altinors et al. [10]	2021	motor, propeller	method combining statistical feature extraction and machine learning
Park et al. [11]	2021	GPS spoofing, DoS attack, rudder, elevator, aileron, engine	unsupervised learning
Souza et al. [12]	2021	motor	signal analysis technique based on chaos
Zheng et al. [13]	2022	actuator, aero engine, equipment	data assembly annotation method based on flight data and BIT records

Authors	Year	Fault Location	Used Methods
Cabahug et al. [14]	2022	actuator	k-means clustering algorithm

Working out how to optimize the original data or improve the relevant algorithms has also become a major research trend in recent years. Zhang et al. [15] proposed a robust deformed denoising CNN (RDDCNN) to solve the offset pixels of feature maps from noisy images. This approach can effectively improve data quality and support data-based UAV fault diagnosis. Tian et al. [16] proposed a multi-stage image denoising CNN with a wavelet transform (MWDCNN) via three stages. A dynamic convolution is used in a CNN to address the limitations in depth and width of lightweight CNNs to achieve good denoising performance.

The performance of data-based methods mainly depends on the quality of the original data. The fault data of drones is often multi-level linkage and cross-influence data, which is not conducive to analyzing complex types of faults. In addition, data-based methods are often offline and difficult to analyze in real time. This method cannot support the real-time safety monitoring of drones. It is more of a means of analysis after accidents, and cannot detect and handle abnormal situations during the operation of drones. Therefore, this method cannot guarantee the safety of drones during operation.

2. Failure Mode Databases

In recent years, failure mode databases have become a research hot spot in database system engineering, knowledge engineering, and other fields, and this approach has been applied in many fields. Failure mode databases are structured, easy-to-use, comprehensive, and organized failure clusters—a collection of interrelated patterns stored, organized, managed, and used in computer memory in a certain way in order to solve the needs of problems in some fields. These failure modes include theoretical knowledge related to the domain, factual data, and heuristic knowledge derived from expert experience. UAV failure mode databases are of great significance to the study of reliability trends. How to form a real scientific failure mode database is one of the key contents of the research.

A common solution to this problem is to extract UAV failure patterns from the records contained in a UAV accident database by cleaning, parsing, and normalizing the fault data. For example, by obtaining accident records from different countries and analyzing the causes of accidents. Suitable data sources include:

- NASA and the FAA Aviation Safety Reporting System (ASRS) [17];
- U.S. Aviation Safety Communique (SAFECOM) data [18];
- Australian Transport Safety Bureau (ATSB) aviation safety investigations and reports [19];
- UK Air Accident Investigation Branch (AAIB) data [20].

The attribute fields in the database should include the data source, accident year, month, location, flight phase, aircraft model, flight hour history, flight time of the mission prior to the accident, altitude, weather, mission being performed, root cause, and accident outcome. At present, the research on UAV accident databases mainly focuses on the collection and analysis of accident data. Wild et al. [21] collected and analyzed 152 incident and accident records, but only 40 records deal with UAV accidents, and the data date back to 2006–2015. More recently, a total of 160 accident reports from 2015 to 2020 were collected and documented in an integrated database [22].

3. Runtime Verification

Runtime verification techniques are formal verification techniques, which use logical formulas to describe properties and transform them into formal structures. Runtime verification tends to check that there is a poor path, that is, up to the current time, whether the system's running path is within safety parameters. After being proposed, this method has received continuous attention from academia and industry. One of its characteristics is that it plays a role in the system software—it is in the real run environment used to monitor the system—so it can find potential defects that the traditional software testing method misses. It complements classical verification techniques (for example, theorem proving and model checking) and provides a more practical method for the verification of system running trajectories. At the cost of limited execution coverage, runtime validation provides accurate information about the runtime behavior of the monitored system for subsequent analysis. The object of action can be a software system, a hardware or information physical system, a sensor network, or any system where dynamic behavior can be observed. The following is a brief overview of some recent research on runtime validation.

Abbas et al. [23] introduced private runtime verification. Liu et al. [24] introduced the idea of incremental verification and proposed an incremental probabilistic model-checking method based on heuristics. Y-Rozier et al. [25] explore the connection and difference between simulation and runtime verification. Stockmann et al. [26] proposed architecture runtime verification. Teixeira et al. [27] studied the properties of a runtime verification-specified API and wrote a minimalist specification language. Bicevskis et al. [28] discuss data quality checking during the execution of a business process by using runtime verification. Lee et al. [29] developed an effective model-checking method for IoT system operation verification. Legunsen et al. [30] proposed an aware runtime verification technique. Geng et al. [31] introduced a new smart tagging method to verify the completion of tasks involving one-to-many and many-to-one dependencies. Ring et al. [32] significantly reduced the size of the state space of the verification process and reduced the complexity of the detection process. Ye et al. [33] proposed an adaptive runtime verification method based on multi-agent systems. Tsigkanos et al. [34] proposed a service-oriented software architecture and technical framework to support the runtime verification of decentralized edge-dense systems. Hu et al. [35] proposed a runtime verification method based on the Robotic Operating System (ROS). Tracy et al. [36] introduced an open-source framework to achieve efficient and high-performance runtime monitoring. Ye et al. [37] developed a new approach based on approximate computation to achieve sufficiently fast and accurate repeated execution of security verification. Kong et al. [38] proposed a method aimed at monitoring traces that reveal the runtime state of the software. Jung et al. [39] proposed an automatic runtime prioritization method based on a classification tree. Miranda et al. [40] proposed a method to automatically detect whether the errors reported by the monitor are real

errors. The authors of [41] proposed the concept of UAV safe operation monitoring and the operational limits to be monitored. A prominent example of such operational limitations is geofencing. Geofencing uses virtual fencing to prevent drones from entering restricted airspace. Felipe et al. [42] proposed a solution based on stream runtime verification, which offers a high-level declarative language to describe sophisticated monitors together with guarantees on execution time and memory usage. They showed how monitors can be combined with temporal planning not only to monitor assumptions but also to support mitigation and remediation in UAV missions. Bonnah et al. [43] presented a rewriting-based algorithm for runtime monitoring of safety requirements expressed in TWTL for specifying time-bounded serial tasks.

Runtime verification is widely used in academia and industry to ensure the reliability and security of the system, whether it is before deployment, testing, verifying, debugging, or after deployment. However, the setting of monitoring conditions still lacks a solid basis. Therefore, this research intends to analyze the failure mechanism of UAVs through the UAV failure mode database to extract the monitoring conditions. According to the above monitoring conditions, the UAV is monitored in real time to improve its reliability.

References

1. D'Amato, E.; Nardi, V.A.; Notaro, I.; Scordamaglia, V. A Particle Filtering Approach for Fault Detection and Isolation of UAVs IMU Sensors: Design, Implementation and Sensitivity Analysis. Sensors 2021, 21, 3066.
2. Fu, X.; Geng, X. Fault Estimation and Robust Fault-tolerant Control for Singular Markov Switching Systems with Mixed Time-Delays and UAVs Applications. J. Control Eng. Appl. Inform. 2021, 23, 53–66.
3. Maqsood, H.; Taimoor, M.; Ullah, Z.; Ali, N.; Sohail, M. Novel Sensor Fault Detection and Isolation for an Unmanned Aerial Vehicle. In Proceedings of the 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST), Islamabad, Pakistan, 12–16 January 2021; pp. 486–493.
4. Miao, Q.; Wei, J.; Wang, J.; Chen, Y. Fault Diagnosis Algorithm Based on Adjustable Nonlinear PI State Observer and Its Application in UAVs Fault Diagnosis. Algorithms 2021, 14, 119.
5. Nejati, Z.; Faraji, A.; Abedi, M. Robust Three Stage Central Difference Kalman Filter for Helicopter Unmanned Aerial Vehicle Actuators Fault Estimation. Int. J. Eng. 2021, 34, 1290–1296.
6. Sun, K.; Gebre-Egziabher, D. Air data fault detection and isolation for small UAS using integrity monitoring framework. Navigation 2021, 68, 577–600.
7. Cao, L.; Yang, X.; Wang, G.; Liu, Y.; Hu, Y. Fault detection based on extended state observer and interval observer for UAVs. Aircr. Eng. Aerosp. Technol. 2022, 71, 1759–1771.

8. Gai, W.; Li, S.; Zhang, J.; Zheng, Y.; Zhong, M. Dynamic Event-Triggered Hi/H∞ Optimization Approach to Fault Detection for Unmanned Aerial Vehicles. IEEE Trans. Instrum. Meas. 2022, 71, 1–11.
9. Lin, C.E.; Shao, P.C. Failure analysis for an unmanned aerial vehicle using safe path planning. J. Aerosp. Inf. Syst. 2020, 17, 358–369.
10. Altinors, A.; Yol, F.; Yaman, O. A sound based method for fault detection with statistical feature extraction in UAVs motors. Appl. Acoust. 2021, 183, 108325.
11. Park, K.H.; Park, E.; Kim, H.K. Unsupervised Fault Detection on Unmanned Aerial Vehicles: Encoding and Thresholding Approach. Sensors 2021, 21, 2208.
12. Souza, J.S.; Bezerril, M.C.; Silva, M.A.; Veras, F.C.; Lima-Filho, A.; Ramos, J.G.; Brito, A.V. Motor speed estimation and failure detection of a small UAVs using density of maxima. Front. Inf. Technol. Electron. Eng. 2021, 22, 1002–1009.
13. Zheng, K.; Jia, G.; Yang, L.; Wang, J. A Compound Fault Labeling and Diagnosis Method Based on Flight Data and BIT Record of UAVs. Appl. Sci. 2021, 11, 5410.
14. Cabahug, J.; Eslamiat, H. Failure Detection in Quadcopter UAVs Using K-Means Clustering. Sensors 2022, 22, 6037.
15. Zhang, Q.; Xiao, J.; Tian, C.; Chun-Wei Lin, J.; Zhang, S. A robust deformed convolutional neural network (CNN) for image denoising. CAAI Trans. Intell. Technol. 2022, 8, 331–342.
16. Tian, C.; Zheng, M.; Zuo, W.; Zhang, B.; Zhang, Y.; Zhang, D. Multi-stage image denoising with the wavelet transform. Pattern Recognit. 2023, 134, 109050.
17. Aviation Safety Reporting System. NASA, FAA. Available online: https://asrs.arc.nasa.gov/ (accessed on 20 February 2022).
18. Aviation Safety Reporting System. The Department of the Interior (DOI) and the U.S. Forest Service (USFS). Available online: https://www.safecom.gov/about (accessed on 20 February 2022).
19. Australian Transport Safety Bureau (ATSB). Available online: https://www.atsb.gov.au/ (accessed on 20 July 2023).
20. Air Accidents Investigation Branch (AAIB). Available online: https://www.gov.uk/government/organisations/air-accidents-investigation-branch (accessed on 20 July 2023).
21. Wild, G.; Murray, J.; Baxter, G. Exploring civil drone accidents and incidents to help prevent potential air disasters. Aerospace 2016, 3, 22.

22. Real-World Faults and Their Injection into Autonomous Unmanned Aerial Vehicles. Available online: https://haotianchen.net/project/adfi (accessed on 20 July 2023).
23. Abbas, H. Work-in-Progress: Private Runtime Verification. In Proceedings of the 2019 International Conference on Embedded Software (EMSOFT), New York, NY, USA, 13–18 October 2019; pp. 1–2.
24. Liu, Y.; He, C. A Heuristics-Based Incremental Probabilistic Model Checking at Runtime. In Proceedings of the 2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 16–18 October 2020; pp. 355–358.
25. Rozier, K.Y. From Simulation to Runtime Verification and Back: Connecting Single-Run Verification Techniques. In Proceedings of the 2019 Spring Simulation Conference (SpringSim), Tucson, AZ, USA, 29 April–2 May 2019; pp. 1–10.
26. Stockmann, L.; Laux, S.; Bodden, E. Architectural Runtime Verification. In Proceedings of the 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), Hamburg, Germany, 25–26 March 2019; pp. 77–84.
27. Teixeira, L.; Miranda, B.; Rebêlo, H.; d'Amorim, M. Demystifying the Challenges of Formally Specifying API Properties for Runtime Verification. In Proceedings of the 2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST), Porto de Galinhas, Brazil, 12–16 April 2021; pp. 82–93.
28. Bicevskis, J.; Bicevska, Z.; Nikiforova, A.; Oditis, I. Towards Data Quality Runtime Verification. In Proceedings of the 2019 Federated Conference on Computer Science and Information Systems (FedCSIS), Leipzig, Germany, 1–4 September 2019; pp. 639–643.
29. Lee, E.; Seo, Y.-D.; Kim, Y.-G. A Cache-Based Model Abstraction and Runtime Verification for the Internet-of-Things Applications. IEEE Internet Things J. 2020, 7, 8886–8901.
30. Legunsen, O.; Zhang, Y.; Hadzi-Tanovic, M.; Rosu, G.; Marinov, D. Techniques for Evolution-Aware Runtime Verification. In Proceedings of the 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST), Xi'an, China, 22–27 April 2019; pp. 300–311.
31. Geng, T.; Njilla, L.; Huang, C.-T. Smart Markers in Smart Contracts: Enabling Multiway Branching and Merging in Blockchain for Decentralized Runtime Verification. In Proceedings of the 2021 IEEE Conference on Dependable and Secure Computing (DSC), Aizuwakamatsu, Japan, 30 January–2 February 2021; pp. 1–8.
32. Ring, M.; Bornebusch, F.; Luth, C.; Wille, R.; Drechsler, R. Verification Runtime Analysis: Get the Most Out of Partial Verification. In Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 873–878.
33. Ye, X.; Liu, W.; Wang, N. Runtime Verification of Multi-Agent Self-Adaptive System. In Proceedings of the 2021 IEEE 24th International Conference on Computer Supported

Cooperative Work in Design (CSCWD), Dalian, China, 5–7 May 2021; pp. 12–17.

34. Tsigkanos, C.; Bersani, M.M.; Frangoudis, P.A.; Dustdar, S. Edge-Based Runtime Verification for the Internet of Things. IEEE Trans. Serv. Comput. 2021, 15, 16.
35. Hu, C.; Dong, W.; Yang, Y.; Shi, H.; Zhou, G. Runtime Verification on Hierarchical Properties of ROS-Based Robot Swarms. IEEE Trans. Reliab. 2020, 69, 674–689.
36. Tracy, T.; Tabajara, L.M.; Vardi, M.; Skadron, K. Runtime Verification on FPGAs with LTLf Specifications; TU Wien Academic Press: Vienna, Austria, 2020; pp. 36–46.
37. Ye, M.; Feng, X.; Wei, S. Runtime Hardware Security Verification Using Approximate Computing: A Case Study on Video Motion Detection. In Proceedings of the 2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), Xi'an, China, 16–17 December 2019; pp. 1–6.
38. Kong, S.; Lu, M.; Li, L.; Gao, L. Runtime Monitoring of system Execution Trace: Method and Tools. IEEE Access 2020, 8, 114020–114036.
39. Jung, B.; Kruse, P.M. Runtime Prioritization with the Classification Tree Method for Test Automation. In Proceedings of the 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Porto, Portugal, 24–28 October 2020; pp. 376–379.
40. Miranda, B.; Lima, I.; Legunsen, O.; d'Amorim, M. Prioritizing Runtime Verification Violations. In Proceedings of the 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), Porto, Portugal, 24–28 October 2020; pp. 297–308.
41. Schirmer, S.; Torens, C. Safe operation monitoring for specific category unmanned aircraft. In Automated Low-Altitude Air Delivery: Towards Autonomous Cargo Transportation with Drones; Springer: Berlin/Heidelberg, Germany, 2021; pp. 393–419.
42. Gorostiaga, F.; Zudaire, S.; Sánchez, C.; Schneider, G.; Uchitel, S. Assumption monitoring of temporal task planning using stream runtime verification. In International Symposium on Leveraging Applications of Formal Methods; Springer: Cham, Switzerland, 2022; pp. 397–414.
43. Bonnah, E.; Hoque, K.A. Runtime monitoring of time window temporal logic. IEEE Robot. Autom. Lett. 2022, 7, 5888–5895.