A Hybrid Model-Based and Data-Driven Framework for Automated Spacecraft Fault Detection

Eric Pesola[1], Ksenia Kolcio[2], Maurice Prather[3], and Adrian Ildefonso[4]

[1,4] *U.S. Naval Research Laboratory, Washington, DC, 20375, USA*
eric.pesola@nrl.navy.mil
adrian.ildefonsorosa@nrl.navy.mil

[2, 3] *Okean Solutions, Inc., Seattle, WA, 98122, USA*
ksenia@okean.solutions
maurice@okean.solutions

Abstract

Traditional fault management can be an onerous task and robust automated solutions are increasingly necessary to accommodate the complexities of modern space systems and mission operations. The present work proposes a hybrid framework for performing automated spacecraft fault detection by leveraging the benefits of both model-based and data-driven approaches. The framework uses a system model to generate residual data that are subsequently fed into a data-driven residual analysis stage. The framework was verified by using data from a hardware-in-the-loop test campaign in which faults were injected into a spacecraft attitude control system, and successfully identified. The fault detection approach implemented using this framework outperformed results obtained from expert-tuned fault detection parameters. Overall, the proposed framework is a promising alternative for sustainable fault detection and mission operations suitable for complex space systems.

1. Introduction

Fault management is one of the fundamental responsibilities of spacecraft operation. Failure to detect and mitigate faults can have a wide range of adverse effects, including loss of mission. Traditionally, spacecraft fault management is a largely manual process, where ground-based fault detection and diagnosis is performed by operators monitoring and analyzing telemetry down-linked from the spacecraft (Djebko, Puppe, & Kayal, 2019). Current automated ground systems often operate using simple limit checks that declare faults when a telemetry value exceeds a pre-defined threshold. An illustration of this approach is shown in Figure 1. Faults are detected in a given telemetry channel if the telemetry value strays outside of a defined nominal range. Selection of thresholds for these systems is typically performed manually by subject matter experts who use prior mission data, spacecraft testing data, and engineering judgement to establish nominal data ranges. Threshold selection is often performed multiple times throughout the mission as components naturally degrade or fail over time. Considerable time and effort must be invested to establish and maintain these systems over a full mission lifetime.

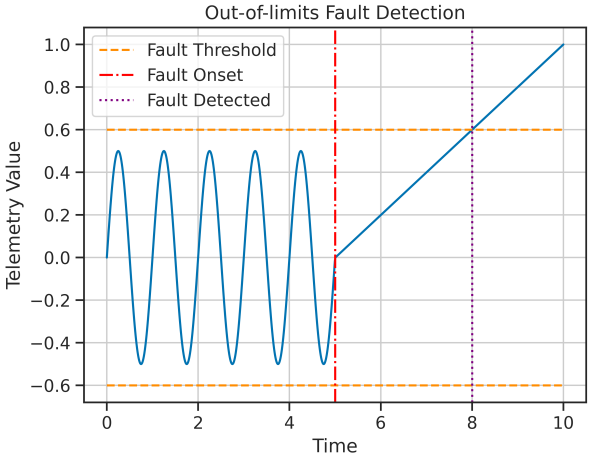

Figure 1. Traditional limit-based fault detection. Limits are set around the estimated nominal telemetry range and faults are declared when limits are exceeded.

As space systems become more complex, the traditional approach to fault management becomes unsustainable. Increasing data rates and sensing capabilities have resulted in spacecraft that generate more telemetry and mission data (Mukai, Towfic, Danos, Shihabi, & Bell, 2020). There are numerous

telemetry streams generated by a spacecraft and manual monitoring of all telemetry would require a prohibitive amount of labor. Instead, a subset of telemetry channels may be chosen for inspection, but this reduces the amount of available system information, making it more difficult to detect faults. Furthermore, space systems are increasingly composed of multi-spacecraft constellations, multiplying the number of vehicles that must be monitored. This drastic increase in the quantity of data can quickly overwhelm operators and the existing systems in place to support fault detection efforts. In response to this paradigm shift, the community increasingly looks to more automated fault management solutions.

One family of approaches for automated fault detection utilizes data-driven or machine learning systems. Fault detection based on machine learning broadly falls into two categories: supervised and unsupervised. Supervised approaches rely on labeled training data that contains both nominal and faulty samples, which are typically expensive to obtain and may not be representative of all possible faults (K. H. Park, Park, & Kim, 2021). In contrast, unsupervised approaches do not require labeled data and typically frame fault detection as a one-class classification or anomaly detection problem. In this case, the machine learning model detects faults by estimating the nominal operating conditions of the system and identifying anomalous deviations from this nominal state (Nalepa et al., 2022). Regardless of learning paradigm, the main drawback of a purely data-driven fault detection approach is that it fails to incorporate explicit knowledge about the physical system. This can result in reduced fault detection performance if the model is unable to learn the complete system dynamics (Hundman, Constantinou, Laporte, Colwell, & Soderstrom, 2018). In addition, data-driven approaches often provide limited understanding of the origin of faults, complicating the isolation and recovery process. Thus, fault mitigation techniques that rely solely on data-driven approaches may not be sufficient for practical spacecraft operations.

Model-based fault detection is an alternative method that has received significant attention (Marzat, Piet-Lahanier, Damongeot, & Walter, 2012). The approach is based on generation and analysis of residuals, which are measurements of deviation between the observed state of the system and the modeled nominal state. Nominal state estimates are derived from system models that emulate the expected or desired behavior of the system. The most common model-based approach leverages residual generators that produce a set of residuals that are sensitive to specific types of faults (Isermann, 2005). The process for developing appropriate residual generators typically relies on domain knowledge and analysis of the system, and requires that the set of faults to be detected is known in advance. Consequently, it can be a fairly inflexible and expensive approach. Other model-based approaches detect faults by identifying when a given residual value exceeds a set threshold. This approach is more flexible but, as with simpler limit-checking systems, threshold tuning is often performed manually by a subject matter expert. As a result, there is substantial interest in increasing the efficiency of model-based systems (Jung & Sundström, 2019).

Model-based and data-driven fault detection approaches in isolation each have unique strengths and weaknesses. Therefore, a number of hybrid approaches to fault detection that combine model-based and data-driven techniques have been proposed. Many proposed techniques couple fault detection and diagnosis into a single step using residual generators (Tidriri, Chatti, Verron, & Tiplica, 2016). One proposed approach uses a data-driven method to create an ensemble diagnosis from multiple model-based systems (Slimani, Ribot, Chanthery, & Rachedi, 2018). Another approach uses a data-driven method to automatically generate residual sets and an anomaly classifier to classify faults (Jung, Ng, Frisk, & Krysander, 2018). A framework combining both model-based and data-driven systems as feature extractors for a downstream diagnostic engine has also been proposed (Khorasgani, Farahat, Ristovski, Gupta, & Biswas, 2018). The primary limitation of such systems is that the use of residual generators typically requires manual analysis and prior knowledge of all fault classes to be detected, which may not be feasible for some missions. While much of the existing literature has explored various hybrid frameworks for systems that explicitly model faults, there is an opportunity to develop hybrid fault detection methods without explicit fault modeling.

The goal of the present work is to propose and demonstrate a new hybrid framework that combines the strengths of model-based and data-driven fault detection approaches. The proposed framework uses nominal system models and physical system data to generate residuals that are then used to train an unsupervised anomaly detector. Anomalies identified in the residual data are indicative of faults. To demonstrate the concept, data taken from a hardware-in-the-loop test campaign for a particular model-based fault detection system were used to fit an outlier detection model, which was then used to automatically determine fault detection thresholds. The results of the analysis indicate that the proposed framework is capable of improved fault detection performance, without explicit fault modeling, and with a significant reduction in labor requirements from subject matter experts. Overall, the proposed framework is a promising alternative for sustainable fault detection and mission operations in the face of the growing complexity of space systems.

2. PROPOSED HYBRID FAULT DETECTION FRAMEWORK

A fault in a system can be described as an unacceptable deviation from nominal state resulting in degraded system performance. In the case of a spacecraft, system state is determined

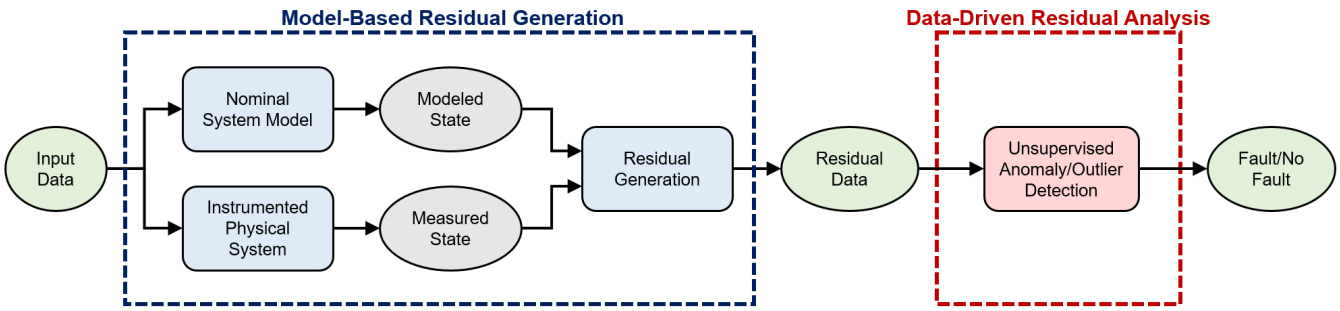

Figure 2. The proposed hybrid fault detection framework. In the first stage, modeled nominal data are compared with measured system state to form residuals. The residual data are then input to the second stage, which performs fault detection via unsupervised anomaly detection.

by the telemetry values at a given point in time. Nominal state refers to the expected state of the system under normal operating conditions, and nominal behavior refers to the expected evolution of system state over time. Because significant departures from nominal behavior indicate the presence of a fault, the fault detection problem can be framed in terms of anomaly detection, where the goal is to identify abnormal deviations between the expected and actual behavior of the system. An advantage of this approach is that faults do not have to be explicitly modeled. Rather than developing a classifier or fault-specific residuals that require prior knowledge of the faults to be detected, the fault detection process is based instead only on departure from a known nominal baseline.

Leveraging this principle, the proposed hybrid framework consists of two stages: (1) model-based residual generation and (2) data-driven residual analysis. A schematic representation of the general fault detection process utilizing this framework is shown in Figure 2. In the first stage, measurements taken from the physical system are compared to modeled system behavior to generate a set of residuals. The data are then provided to the second stage, which detects faults by identifying anomalies in the residual set. A description of these two stages is provided below.

2.1. Model-Based Residual Generation

The first stage of the framework is responsible for generating the residuals that are used for fault detection. A residual is a value that quantifies the degree of difference between the nominal behavior of the system and the measured or actual system behavior (Gertler, 1991). Because residuals represent deviations from nominal behavior, large or abnormal residual values indicate a higher likelihood of a fault. In this work, a residual is defined as a quantitative difference between a modeled and measured signal. For spacecraft, measurable signals in the system are telemetry channels. The simplest version of a residual is the error signal obtained by subtracting the modeled signal from the measured signal. An example of this type of residual generation is shown in Figure 3, where

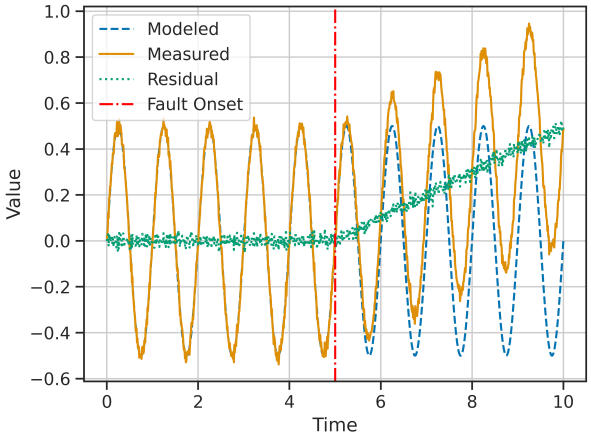

Figure 3. An example of generating a residual signal. The modeled signal is subtracted from the measured signal to obtain the residual. Abnormal variations in the residual signal — such as the sudden change of slope in this example — suggest the presence of a fault.

the residual value abruptly increases after the onset of a fault that leads the measured signal to deviate unexpectedly. Other measures of distance or spread may also be used as residuals, such as statistical deviation measures.

Generation of a residual requires a nominal or expected signal, which can be obtained from a system model, as well as a measurement of the corresponding signal in the physical system. Most real-world systems will have multiple residuals where each represents a different descriptor of system behavior. Each individual residual generated from this process forms the complete residual set. The form of the complete residual set is a $t \times m$ time series, where t indexes the time dimension and m is the number of individual residual signals obtained from the system. Under this definition, there are two requirements for residual generation: (i) sensors instrumented on the physical system and (ii) nominal system models.

2.1.1. Instrumented Physical System

To detect faults using residuals, a measurement of system state must be available for comparison with the output of a system model. Examples of these types of signals include those obtained from rotary encoders for attitude control, thermistors for spacecraft thermal management, and ammeters for proper operation of electronic systems. Modern spacecraft are typically designed with a plethora of sensors that are used to generate telemetry for various subsystems, which can be used to generate residuals. However, for missions requiring additional fault detection capabilities, it may be necessary to instrument the system with additional sensors that are more descriptive of the relevant system behavior. Nonetheless, satisfying this requirement is accessible to spacecraft designers who routinely deploy these types of sensors in physical space systems.

2.1.2. Nominal System Model

The primary requirement of the system model is that it emulates the expected behavior of the system for all nominal operating conditions. Because faults are detected according to the system's departure from expected behavior, failure to correctly model nominal system behavior could cause normal spacecraft state to be incorrectly declared as faulty. Any means of estimating the nominal state of the system may be used in this step, provided that the corresponding measurement can be obtained from the system. Examples of system state may include electrical currents, data communication bit rates, and spacecraft attitude rates. Custom models may be designed for this stage if required, or re-used from other engineering tasks if possible. For instance, any engineering models created as part of the system design may be used in this stage. Process, mathematical, and logical models are all suitable candidates for use in system modeling.

The required fidelity of the system model depends on the fault detection requirements of the mission. Higher-fidelity models offer greater ability to characterize nominal behavior for all operating conditions and thus enable improved fault detection. For complex spacecraft systems, a complete system-level model may contain subsystem models which are themselves composed of component-level models. As an example, a spacecraft's system model may contain a subsystem model for attitude control that contains component models for reaction wheels. Ultimately, there is a trade-off between the effort required to create a higher-fidelity system model and any resulting fault detection performance gains. Therefore, modeling effort should be considered along with fault detection requirements when implementing the framework. Finally, due to the residual generation requirement of the framework, it is not generally required to model component behaviors which are not directly measurable in the physical system.

The full residual set is generated from the nominal system model and the instrumented physical system. A residual is generated from each pair of modeled and measured signals, and the resulting residual set is provided to the second stage of the framework to identify faults.

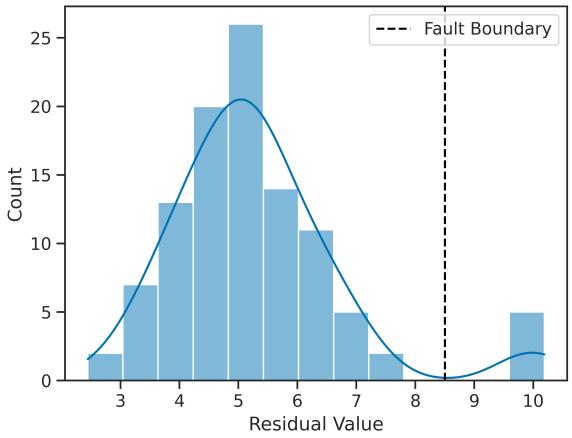

Figure 4. An example of a nominal residual distribution (left of boundary) and outliers indicating faulty behavior (right). The boundary between nominal and faulty behavior is determined through outlier or anomaly detection.

2.2. Data-Driven Residual Analysis

Once the residual set has been generated, it is analyzed in the residual analysis stage to perform fault detection. Each data point in the residual set represents the degree to which the system is deviating from its nominal state. Due to various effects such as sensor noise and modeling inaccuracies, residual values will often be non-zero. However, values obtained during nominal operation will tend to form a nominal distribution. An anomalous value suggests an off-nominal system state and thus indicates a higher probability of a fault. Figure 4 illustrates the concept for a single residual value. For real dynamic systems, the nominal data distribution will be multivariate due to multiple residuals and may be time-dependent due to a variety of influences. Because identification of faulty data points relies only on their deviation from the nominal data, faults do not have to be explicitly modeled. This work therefore frames fault detection as an unsupervised anomaly detection problem, where sufficiently anomalous points in the residual data indicate the presence of a fault. Note that hereafter the terms outlier and anomaly are used interchangeably.

Any unsupervised outlier or anomaly detection algorithm, or some combination thereof, may be used in this step. A considerable amount of work has been performed in the field of unsupervised anomaly detection (Pang, Shen, Cao, & Hengel, 2021; Chandola, Banerjee, & Kumar, 2009), some of which has focused on spacecraft (Shriram & Sivasankar, 2019; Gao,

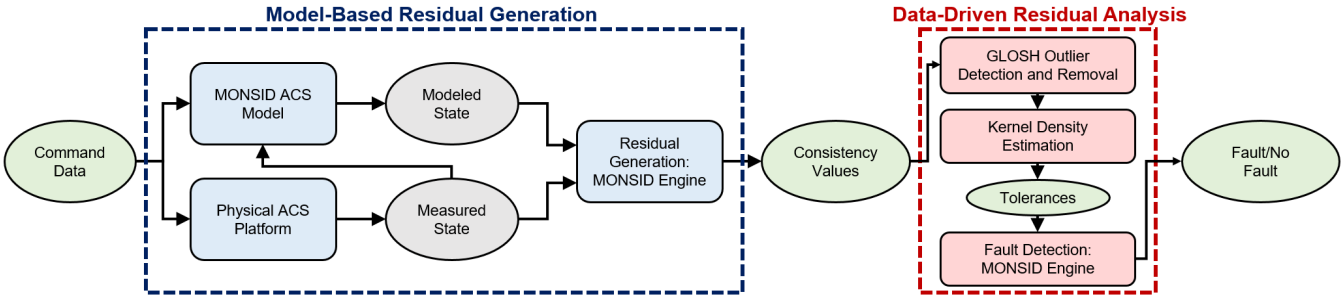

Figure 5. Schematic diagram of the implemented framework in the case study. The MONSID model and test bed were used to generate residual data in the form of consistency values. The GLOSH algorithm was used to identify and remove outliers to form a nominal data distribution, and kernel density estimation was applied to the nominal data distribution to estimate the tolerances responsible for detecting faults. Note that the incorporation of state measurements into the propagation of the system model is a feature of analytic redundancy but is not generally a requirement for generating a nominal system response.

Yang, Xu, & Xing, 2012; Yu, Song, Tang, Han, & Dai, 2021). Because of the extensive amount of prior work performed in this field, there are many available options for implementation of the framework.

Two notable considerations for this stage include the dimension of the residual set and temporal dependence between data points. Because spacecraft are dynamic systems, the residual data is time series data, and therefore time series anomaly detection approaches that can identify temporal anomalies should be considered. This is not requisite, however, and depending on the mission it may be valid to incorporate more traditional outlier detectors based on clustering or partitioning. For multivariate residual sets, a fault may be characterized by an anomaly in a single residual, or it may be formed by a more complex multivariate anomaly. For multivariate residual sets it is therefore recommended to select a multivariate anomaly detector. As with other machine learning tasks, ensembles of models may be used in an effort to obtain more robust anomaly detection performance (Zimek, Campello, & Sander, 2014).

2.3. Summary

The proposed framework detects faults through unsupervised anomaly detection on the residuals formed from nominal system models and measured system data. The approach is modular in that it is agnostic to the specific model-based and data-driven approaches used, provided that they fulfill the requirements introduced in their respective sections. This flexibility makes the framework robust to physical systems of varying complexity, albeit at the potential expense of greater modeling effort. Further, no explicit fault modeling is required because the approach is based on deviation from expected system performance.

3. CASE STUDY

To demonstrate the effectiveness of the proposed framework, a proof of concept is shown in this section. This case study is introduced in an order similar to the framework description above to allow straightforward connection between the practical implementation and the theoretical framework requirements. The framework has been applied to detect faults in a spacecraft attitude control system (ACS) test campaign. Data for the study were collected using a hardware-in-the-loop ACS platform along with a specific model-based fault detection and isolation (FDI) system. The Model-based Off-Nominal State Identification and Detection (MONSID) system used in this study is an analytic redundancy-based FDI system composed of an application-specific nominal system model and a generic diagnostic engine (Kolcio & Fesq, 2016). The full implementation of the framework is shown in Figure 5. Due to the architecture and requirements of the MONSID system, this implementation leverages MONSID for three tasks: nominal system modeling, generation of residual values, and fault detection using tolerances automatically obtained in the residual analysis stage.

The physical test bed is a hardware-in-the-loop system owned and operated by the Air Force Research Laboratory that emulates a spacecraft bus. The test bed allows for direct communication with the ACS, enabling operators to send command information, receive telemetry data, and inject faults. Telemetry data can be read from the system for further analysis.

3.1. Residual Generation

During operation, the same commands sent to the test bed were communicated to MONSID. Concurrently, MONSID received telemetry data transferred from the test bed. The commands and telemetry values were used to propagate the system model and generate residuals.

3.1.1. Instrumented Physical System: ACS Test Bed

The ACS hardware platform simulates a spacecraft bus using three-axis rotation provided by six control moment gyroscopes (CMGs), an inertial measurement unit (IMU), and a motion capture (MOCAP) system that simulates the function of a star tracker. Sensors such as rotary encoders and hall sensors are instrumented in each ACS component to record relevant telemetry including angular displacement rates. Commands and telemetry from sensors can be transmitted between the test bed and the controller via a RS-422 communication line. The MONSID engine was integrated with the test bed using the cFS flight software framework (Kolcio & Prather, 2023).

To emulate faults in the physical system, the ACS test campaign was completed in a sequence of individual runs of the system. During some runs, the system was allowed to perform without any faults present, while other runs contained faults injected into the system by the test operators. The 29 total runs considered in this analysis were split between 15 fault injection runs and 14 nominal runs. A single fault injection run was arbitrarily chosen to be used as training data, leaving 14 fault injection runs and 14 nominal runs as test data. For the fault injection runs, a fault persisted for the duration of the run after it was injected. Ground truth information was generated for all evaluation runs by assigning a 0 ("no fault") to all time steps prior to fault injection and a 1 ("fault") to all subsequent time steps. Fault detection performance is measured on a per-time step basis; in other words, each time step is considered an independent data point for the purposes of evaluation.

3.1.2. Nominal System Model: MONSID ACS Model

System-level MONSID models are composed of interconnected components that characterize nominal behavior. A notional example of a MONSID model is shown in Figure 6(a). Each component (shown in 6(b)) contains input and output nodes that accept sensor and command data to allow for emulation of a wide array of physical systems. At each processing time step, sensor and command data are propagated throughout the model and multiple state values are stored in each node to generate consistency values. Consistency values are a type of residual that measure the deviation between state estimates at a node. A fault-free system should contain consistency values close to or equal to zero. Larger consistency values correspond to a larger degree of inconsistency between expected and observed behavior and therefore indicate the presence of a fault. The MONSID engine declares a fault in the system when a consistency value exceeds a value called a tolerance for a number of consecutive time steps called a fault window. For this analysis, the fault window was fixed at 17 time steps due to the specific implementation used during the test campaign. Further discussion of the general MONSID framework, FDI process, and generation of consistency values is available in earlier work (Kolcio & Fesq, 2016).

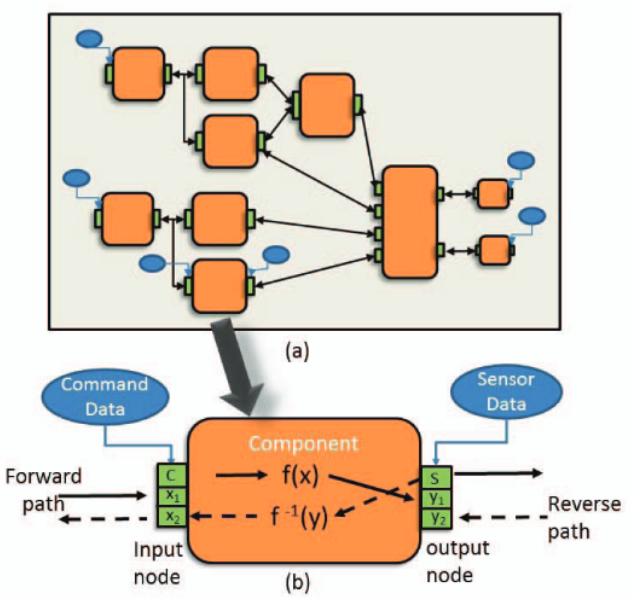

Figure 6. (a) A notional MONSID system model and (b) the anatomy of a component [after (Kolcio & Fesq, 2016)]. Consistency values are generated at each node by propagating data forward and backward through the model, and computing a measurement of deviation between all state values present at the node. Larger consistency values indicate more inconsistent state and thus a higher likelihood of a fault.

The MONSID model of the hardware system used in this case study contains 28 components and 32 sensors. It contains components modeling the test bed hardware including the rotors, gimbals, the IMU, and the MOCAP. Additionally, a CMG Assembly Pseudo-component (CAP) representing the CMG assembly dynamics and a Dynamics Kinematics Pseudo-component (DKP) representing the spacecraft bus dynamics and kinematics were modeled.

The tolerance selection process was performed using the previously mentioned fault injection run. The MONSID engine records consistency values for every node at each time step, resulting in a $t \times m$ matrix of consistency values, where t is the number of time steps in the run and m is the number of nodes in the model. Each row in this residual matrix represents the residual values for each node at a given time step. While there are a total of 89 nodes in the MONSID ACS model, 60 nodes showed consistency values equal to zero for the entire duration of the run (i.e., these nodes did not deviate from nominal). Therefore, these 60 have been omitted from the analysis and the remaining 29 have been considered.

3.2. Residual Analysis

The MONSID engine requires a tolerance to be set for each node in the system to perform fault detection. To accom-

modate this requirement, this study's implementation of the framework therefore utilizes a two-step residual data analysis process. In the first step, an outlier detection algorithm detects and removes outliers in the residual set, resulting in an estimate of each node's nominal data distribution. In the second step, these nominal distributions are used to estimate fault detection tolerances. Once tolerances are obtained, they can be provided to the MONSID engine to detect faults on incoming data.

3.2.1. Outlier Detection and Removal

The residual data analysis for this implementation begins with an unsupervised outlier detection and removal step. The outlier removal step is necessary to ensure that tolerances are established using only nominal data. An illustration of this is provided in Figure 7, which demonstrates the difference between data distributions contaminated by outliers versus the subset of data containing only inliers. Tolerances obtained on data contaminated with faulty samples could significantly overestimate the true range of nominal values, resulting in decreased fault detection performance.

The outlier detection and removal process in this work was performed through an outlier scoring and binarization procedure. Identified anomalies were removed and the remaining data were passed on to the tolerance selection step. The outlier detection algorithm used in this analysis is Global-Local Outlier Score from Hierarchies (GLOSH) (McInnes, Healy, & Astels, 2017). GLOSH was selected for this case study because it is a density-based and noise-aware algorithm capable of identifying both global and localized outliers. It requires no prior assumptions about the training data and often requires little to no hyper-parameter tuning to obtain reasonable results (Campello, Moulavi, Zimek, & Sander, 2015). Additionally, it is a suitable algorithm for smaller data set sizes, in contrast to deep learning-based methods, which may require much larger amounts of data to train an effective model (Pang et al., 2021). The model was fit to the residual matrix with mostly default hyper-parameters; the only exception was the minimum cluster size parameter, which was set to 10% of the training data size. This selection was based on an assumption that any nominal data cluster should contain at least that percentage of the total data. The output of the GLOSH algorithm is a continuous outlier score for each input data point in the range [0, 1], where scores closer to zero indicate inliers and scores closer to one indicate a higher probability of an outlier. A common approach for converting continuous outlier scores to labels is to select a crossover threshold based on the distribution of scores. Here, the 90% quantile was chosen as the transition threshold between inlier and outlier points. All points with scores larger than the threshold were identified as outliers and removed from the data set.

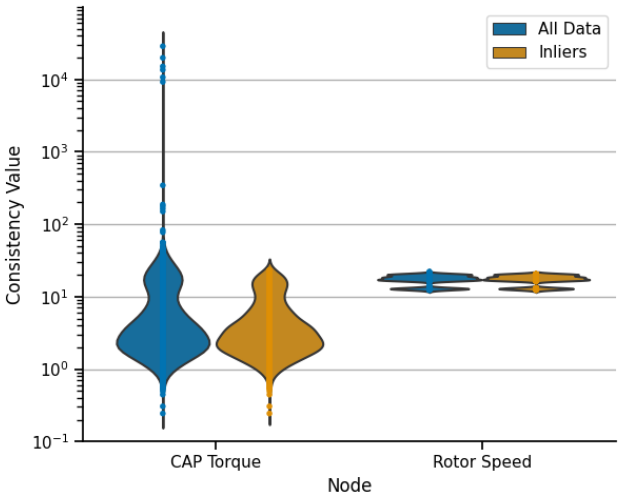

Figure 7. A comparison of initial and outlier-removed data distributions. Importantly, the outlier detection and removal process significantly modifies the probability distribution for nodes contaminated by outliers (CAP Torque) but does not noticeably modify already-nominal distributions (Rotor Speed). The inlier distributions are used for tolerance estimation.

3.2.2. Tolerance Selection

Once outlying data points were identified and removed, tolerances were selected for each node. Tolerances should be selected such that consistency values larger than the set tolerance are attributed to faulty behavior. The task thus reduces to the determination of a threshold for sufficiently unlikely observations of faulty behavior within the nominal data. By computing an estimate of the nominal probability distribution, it is straightforward to select a tolerance. Kernel density estimation was chosen to perform the tolerance estimation process for each node.

Kernel density estimation is a method for determining a continuous probability distribution from samples of data. It has been shown that the bandwidth parameter has a far greater effect on the density estimation than the choice of kernel function. Therefore, a standard Gaussian kernel was used. The bandwidth was estimated via cross-validation (B. U. Park & Marron, 1990). The upper and lower bounds for the bandwidth grid used in the cross-validation procedure were determined for each node via a heuristic approach that seeks to automatically approximate bounds surrounding the optimal bandwidth (Minnotte & Scott, 1993). A final density estimation was performed with the estimated optimal bandwidth to obtain the distribution used for tolerance selection.

Tolerances were selected via the estimated distribution's quantile function. For a given probability p, a tolerance T can be obtained such that the likelihood of obtaining a value more extreme than T is equal to p. In practice, p is a free parameter

representing the the likelihood that an observed value in excess of T is actually a nominal value. This can be tuned to the specific risk posture of a given mission if required, or left to a reasonable default, e.g., $1\% \leq p \leq 20\%$. For this analysis, p was chosen as 10%. Once estimated, all node tolerances were provided to the MONSID engine to perform fault detection.

3.3. Experimental Results

To provide a baseline, the fault detection performance for tolerances obtained by the automated framework implementation is compared to the performance of the tolerances manually selected for the system by human experts. Tuning tolerances manually requires engineering judgement, characterization of sensor noise, and iteration as more data become available (Kolcio & Prather, 2023). The manual tolerances for this study were obtained by reviewing telemetry data taken from the test bed. A comparison of the tolerances obtained by both methods is given in Figure 8. In general, the automated tolerance values are well correlated with the manually determined tolerances. An interesting result is that the framework as implemented selected tolerances that were uniformly smaller (i.e., more prone to declaring a fault) than those manually tuned.

One way to modify detection performance for the system as implemented would be to change the probability threshold p on the distribution obtained by the kernel density estimate in the previous step. For example, a lower probability would result in larger tolerances less prone to detecting faults. Another way to modify detection behavior would be to alter the GLOSH outlier detection threshold in an attempt to modify the nominal data distribution. As shown in comparisons between the original and trimmed data sets, the anomaly detection and removal step has a strong influence on the resulting nominal distribution, which in turn influences the resulting tolerance. Because the density estimation step is primarily dependent on the underlying distribution, care should be taken to implement a robust anomaly detector. Most anomaly detection techniques offer a direct or implicit mechanism for tuning model bias toward precision or recall, thereby influencing the amount of anomalies detected and allowing for further control over the final results if needed. An advantage of this specific implementation is that fault sensitivity can be tuned directly using the probability introduced in the tolerance estimation section.

The efficacy of the proposed framework is evaluated as a binary classification problem where a “fault“ or “no fault“ label is assigned at each time step. The ground truth labels are compared with the predictions from the MONSID engine for both sets of tolerances. A comparison of results for the automated and manual solutions for all time steps in the test data set is shown in Figure 9. Deviations between approaches in all elements of the confusion matrix spanned approximately three percentage points, demonstrating similar but not equal performance. The manually-tuned tolerances showed a slightly lower false positive rate whereas the automated approach demonstrated higher sensitivity to faults. Both of these results are sensible in relation to the Figure 8 comparison demonstrating the more restrictive tolerances obtained by the automated approach.

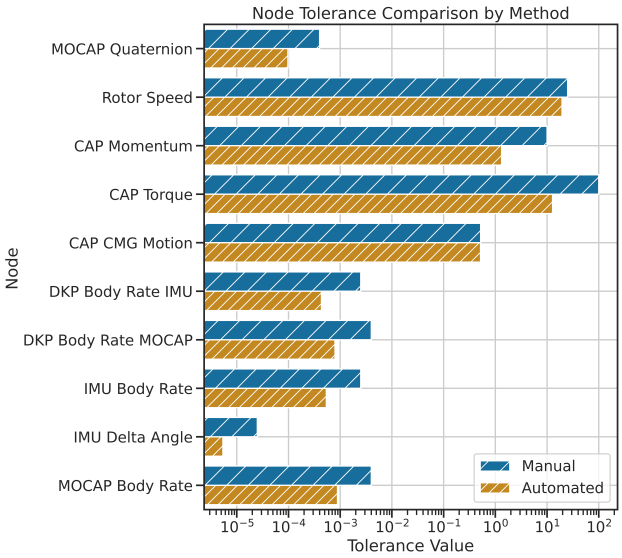

Figure 8. Comparison of tolerance values obtained manually and by the automatic framework for a representative subset of nodes. The automated approach uniformly estimated smaller tolerances than those manually tuned, making the automated approach more conservative (i.e., fault-sensitive).

Similarly to most fault detection problems, the evaluation data set used in this work is imbalanced, with nominal data samples significantly outnumbering faulty samples. For this reason, F1 score and Matthews Correlation Coefficient (MCC) were used to estimate overall performance (Chicco & Jurman, 2020). Precision and recall scores were also obtained. A comparison of metrics for the manual and automated approaches is shown in Figure 10. The results show that the automated framework outperformed the manual method in all metrics except precision. The increase in F1 score and MCC indicate that in addition to detecting more total faults, the automated approach resulted in a superior overall fault detection system.

4. DISCUSSION

A potential benefit of the data-driven stage is that it may automatically compensate for minor modeling inaccuracies or sensor noise prevalent in model-based systems (Gertler, 1991). For example, a small constant bias offset in a residual signal due to a modeling inaccuracy is a temporal pattern that a temporal anomaly detector may correctly learn to ig-

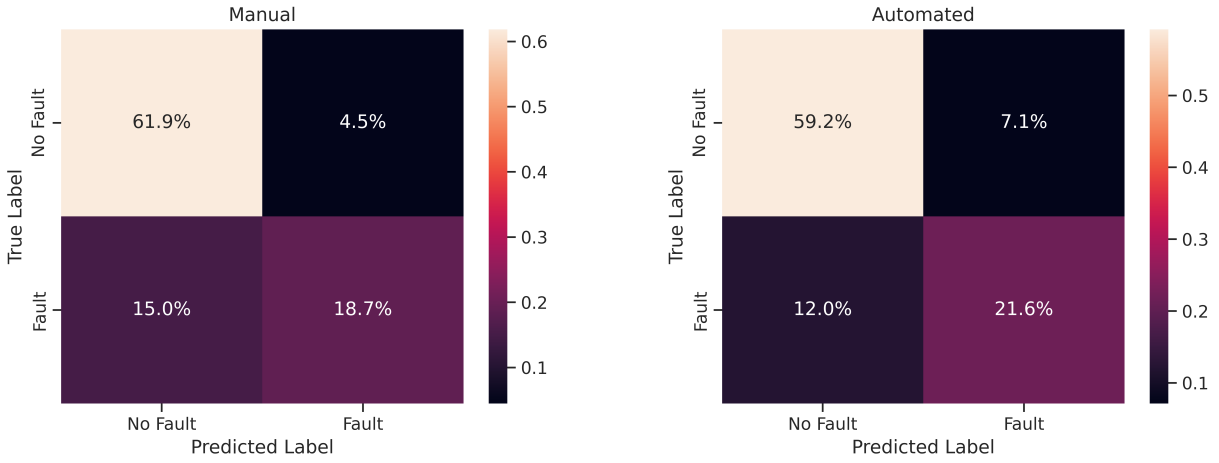

Figure 9. Fault detection confusion matrices for (a) manual and (b) automated tolerances. Compared to the manual approach, the automated method detected more and missed fewer faults at the expense of more false detects.

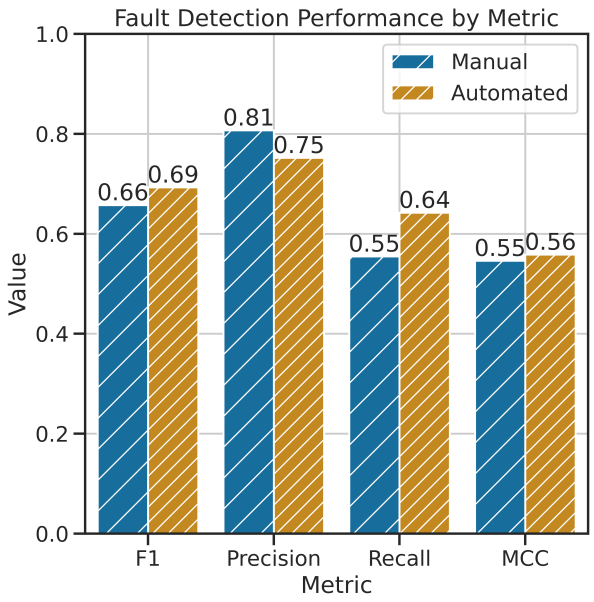

Figure 10. Comparison of various fault detection performance metrics for the manual and automated tolerances. The automatically tuned system slightly outperformed the manual system in terms of overall performance as measured by both F1 and MCC scores. Note that the minimum score for MCC is -1 whereas the minimum score is 0 for all other metrics.

nore as nominal. However, this phenomenon should not be relied upon when modeling the system and significant modeling errors are likely to lead to reduced fault detection performance as previously discussed. Another notable consideration for the implementation of the data-driven stage is that both the model and data may be monitored for drift as the system operates over longer time scales (Lu et al., 2018). Both natural degradation of components as well as incipient faults (i.e., faults that develop slowly) have the potential to gradually modify the residual data distribution over time; if this occurs, off-nominal behavior may gradually become considered as nominal and the anomaly detector could mask faults in the system.

The outlier detection method selected for the implementation of the framework is a point-outlier detector that is not designed to detect contextual or temporal anomalies which may occur. Fault detection performance was not negatively impacted in this implementation, but for many spacecraft missions it will be necessary to use time-series anomaly detection methods capable of detecting both point and temporal anomalies. This may be achieved for traditional anomaly detectors via a windowing method or with deep learning architectures utilizing an inbuilt temporal mechanism such as recurrence or attention (Vaswani et al., 2017).

Finally, for those operating on large-scale and/or high-dimensional data, deep learning methods for anomaly detection offer the potential for increased performance compared to traditional methods and should be increasingly explored as the underlying techniques become more easily automated and computationally efficient. Adding model-based contextual information such as model topology into the data-driven stage may offer the opportunity to increase the performance of the framework due to the additional context provided regarding nominal behavior. For example, a graph neural network trained using model topology may be capable of detecting faults that may elude models without such context (Ma et al., 2021).

Because the framework is modular, it offers wide applica-

bility and flexibility in implementation. Depending on the specific implementation and system hardware the framework may be run onboard the physical system, remotely from a control center, or via a hybrid of the two. Complexity of the implemented framework can be increased or decreased in the system modeling and/or anomaly detection stages as required for a given mission risk posture.

5. CONCLUSION

As performance requirements continue to increase for both onboard and ground-processing operations, it is increasingly important for fault management solutions to be both efficient and robust. This work presents a hybrid framework for performing automated spacecraft fault detection by leveraging the benefits of both model-based and data-driven systems. The examined case study demonstrates that the hybrid framework can produce practical results for real-world systems, without explicitly modeling faults. Experimental results show that the proposed framework is able to outperform a manually tuned fault detection system. Further, implementation of the framework is modular and easily customized to fit the needs of specific missions. This work demonstrates the viability of hybrid fault detection and encourages the community to continue incorporating systems that leverage the strength of combining model-based and data-driven approaches.

ACKNOWLEDGMENT

This work was sponsored in part by a Phase II SBIR contract with the U.S. Air Force Research Laboratory at Kirtland Air Force Base and a Karle's Fellowship at the U.S. Naval Research Laboratory.

REFERENCES

Campello, R. J. G. B., Moulavi, D., Zimek, A., & Sander, J. (2015). Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data*, *10*(1), 1–51. doi: 10.1145/2733381

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys*, *41*(3), 1-58. doi: 10.1145/1541880.1541882

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(6), 1–13. doi: 10.1186/s12864-019-6413-7

Djebko, K., Puppe, F., & Kayal, H. (2019). Model-based fault detection and diagnosis for spacecraft with an application for the sonate triple cube nanosatellite. *Aerospace*, *6*(10), 105. doi: 10.3390/aerospace6100105

Gao, Y., Yang, T., Xu, M., & Xing, N. (2012). An unsupervised anomaly detection approach for spacecraft based on normal behavior clustering. In *Proceedings of the Fifth International Conference on Intelligent Computation Technology and Automation* (p. 478-481). doi: 10.1109/ICICTA.2012.126

Gertler, J. (1991). Analytical Redundancy Methods in Fault Detection and Isolation - Survey and Synthesis. *IFAC/IMACS Symposium on Fault Detection, Supervision and Safety for Technical Processes*, *24*(6), 9-21. doi: 10.1016/S1474-6670(17)51119-2

Hundman, K., Constantinou, V., Laporte, C., Colwell, I., & Soderstrom, T. (2018). Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (p. 387-395). doi: 10.1145/3219819.3219845

Isermann, R. (2005). Model-based fault-detection and diagnosis – status and applications. *Annual Reviews in Control*, *29*(1), 71-85. doi: 10.1016/j.arcontrol.2004.12.002

Jung, D., Ng, K. Y., Frisk, E., & Krysander, M. (2018). Combining model-based diagnosis and data-driven anomaly classifiers for fault isolation. *Control Engineering Practice*, *80*, 146-156. doi: 10.1016/j.conengprac.2018.08.013

Jung, D., & Sundström, C. (2019). A Combined Data-Driven and Model-Based Residual Selection Algorithm for Fault Detection and Isolation. *IEEE Transactions on Control Systems Technology*, *27*(2), 616-630. doi: 10.1109/TCST.2017.2773514

Khorasgani, H., Farahat, A., Ristovski, K., Gupta, C., & Biswas, G. (2018). A framework for unifying model-based and data-driven fault diagnosis. In *Proceedings of the Annual Conference of the PHM Society* (p. 1-10). doi: 10.36001/phmconf.2018.v10i1.530

Kolcio, K., & Fesq, L. (2016). Model-based off-nominal state isolation and detection system for autonomous fault management. In *Proceedings of IEEE Aerospace Conference* (p. 1-13). doi: 10.1109/AERO.2016.7500793

Kolcio, K., & Prather, M. (2023). Implementation and evaluation of model-based health assessment for spacecraft autonomy. In *Proceedings of IEEE Aerospace Conference.* doi: 10.1109/AERO55745.2023.10116001

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346–2363. doi: 10.1109/TKDE.2018.2876857

Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q. Z., ... Akoglu, L. (2021). A Comprehensive Survey on Graph Anomaly Detection with Deep Learning. *IEEE Transactions on Knowledge and Data Engineering*. doi: 10.1109/TKDE.2021.3118815

Marzat, J., Piet-Lahanier, H., Damongeot, F., & Walter, E.

(2012). Model-based fault diagnosis for aerospace systems: A survey. *Proceedings of the Institution of Mechanical Engineers Part G Journal of Aerospace Engineering*, *226*(10), 1329-1360. doi: 10.1177/0954410011421717

McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, *2*(11). doi: 10.21105/joss.00205

Minnotte, M. C., & Scott, D. W. (1993). The Mode Tree: A Tool for Visualization of Nonparametric Density Features. *Journal of Computational and Graphical Statistics*, *2*(1), 51-68. doi: 10.1080/10618600.1993.10474599

Mukai, R., Towfic, Z., Danos, M., Shihabi, M., & Bell, D. (2020). MSL telecom automated anomaly detection. In *Proceedings of IEEE Aerospace Conference* (p. 1-6). doi: 10.1109/AERO47225.2020.9172573

Nalepa, J., Myller, M., Andrzejewski, J., Benecki, P., Piechaczek, S., & Kostrzewa, D. (2022). Evaluating algorithms for anomaly detection in satellite telemetry data. *Acta Astronautica*, *198*, 689-701. doi: 10.1016/j.actaastro.2022.06.026

Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep Learning for Anomaly Detection: A Review. *ACM Computing Surveys*, *54*(2), 1–38. doi: 10.1145/3439950

Park, B. U., & Marron, J. S. (1990). Comparison of Data-Driven Bandwidth Selectors. *Journal of the American Statistical Association*, *85*(409), 66–72. doi: 10.2307/2289526

Park, K. H., Park, E., & Kim, H. K. (2021). Unsupervised Fault Detection on Unmanned Aerial Vehicles: Encoding and Thresholding Approach. *Sensors*, *21*(6). doi: 10.3390/s21062208

Shriram, S., & Sivasankar, E. (2019). Anomaly detection on shuttle data using unsupervised learning techniques. In *Proceedings of International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)* (p. 221-225). doi: 10.1109/ICCIKE47802.2019.9004325

Slimani, A., Ribot, P., Chanthery, E., & Rachedi, N. (2018). Fusion of Model-based and Data-based Fault Diagnosis Approaches. *IFAC-PapersOnLine*, *51*(24), 1205-1211. doi: 10.1016/j.ifacol.2018.09.698

Tidriri, K., Chatti, N., Verron, S., & Tiplica, T. (2016). Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges. *Annual Reviews in Control*, *42*, 63-81. doi: 10.1016/j.arcontrol.2016.09.008

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is All you Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (p. 6000–6010). doi: 10.48550/arXiv.1706.03762

Yu, J., Song, Y., Tang, D., Han, D., & Dai, J. (2021). Telemetry data-based spacecraft anomaly detection with spatial–temporal generative adversarial networks. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1-9. doi: 10.1109/TIM.2021.3073442

Zimek, A., Campello, R. J., & Sander, J. (2014). Ensembles for Unsupervised Outlier Detection: Challenges and Research Questions. *ACM SIGKDD Explorations Newsletter*, *15*(1), 11–22. doi: 10.1145/2594473.2594476