SSC22-WKI-06

Autonomous System-Level Fault Diagnosis in Satellites using Housekeeping Telemetry

Evana Gizzi
NASA Goddard Space Flight Center
8800 Greenbelt Rd, Greenbelt MD; 301-286-2000
Evana.Gizzi@nasa.gov

Hayley Owens
Louisiana Tech University
201 Mayfield Ave, Ruston, LA 71272; 318-257-2000
hjo005@latech.edu

Nicholas Pellegrino
Binghamton University
4400 Vestal Pkwy E, Binghamton, NY 13902; 607-777-2000
npelleg1@binghamton.edu

Christopher Trombley, James Marshall
NASA Goddard Space Flight Center
8800 Greenbelt Rd, Greenbelt MD, 20771; 301-286-2000
christopher.m.trombley@nasa.gov

Jivko Sinapov
Tufts University
19 Boston Ave, Medford, MA 02155; 617-628-5000
Jivko.Sinapov@tufts.edu

ABSTRACT

To continue the growing success of scientific discovery through deep space exploration, missions need to be able to be able to travel further and operate more efficiently than ever before. To ensure resilience in this capability, on-board autonomous fault mitigation methods must be developed and matured. To this end, we present a system for cross-subsystem fault diagnosis of satellites using spacecraft telemetry. Our system leverages a combination of Kalman Filters, Autoencoders, and Causality algorithms. We test our system for accuracy against three data sets of varying complexity levels, along with baseline testing. Additionally, we perform an ablation study to evaluate on-board tractability.

INTRODUCTION

Next-generation autonomous methods for fault recovery need to be considered and developed to enable resilience in spaceflight missions. Despite the success of conventional on board fault handling procedures, there exist limitations to these approaches. To date, the most widely used fault handling approach onboard satellites employs basic limit checking on telemetry value for fault detection.[1] Once a fault is detected, its recovery is determined using a set of onboard rules, preemptively engineered by subject matter experts to trace back fault symptoms to causes. In the case of anomalous, or "never before seen" faults, these approaches fail to provide a real time diagnosis on board, causing the spacecraft to remain in standby (called "safe mode") until ground controllers are able to upload the correct safe mode exiting command sequence. In the interim, vital science data is lost, deteriorating the success of the mission. Moreover, faults which affect the health and status of the spacecraft can be catastrophic if unaddressed with a proper diagnosis, and in a timely manner. To this end, intelligent on board fault diagnosis is crucial.

To address these challenges, we present a system-level approach to autonomous fault diagnosis on-board a spacecraft. We use a combination of artificial intelligence and predictive constructs to render diagnoses, including Kalman Filters, Autoencoders (AE), and Associative Causality. We test our system in three experiments on data sets of varying complexity levels, for both accuracy of diagnosis, and against alternative baseline approaches. Additionally, we present ablation performance tests to evaluate on-board tractability of our system. Our main contributions are as follows:

- We introduce a novel satellite fault diagnosis algorithm
- We demonstrate the algorithm using three sets of realistic spacecraft telemetry data
- We compare against baseline methods
- We demonstrate the tractability of the algorithm on realistic hardware

RELATED WORK

Research in autonomous fault recovery has grown significantly in recent years. While this problem can be decomposed into three main challenge areas (detection, diagnosis, and prediction), we focus our review specifically on diagnosis considerations.[2] Specifically regarding power systems, Wang *et al.* employ deep learning in generalized power systems, as a way to diagnose faults.[3] They use stacked autoencoders (SAE) on preprocessed power data for training, which is then used to initialize a deep learning neural network (DLNN) to classify the *type* of fault. Similar work has examined space-specific power subsystems for diagnosis. Fang et al. use both unsupervised (Denoising Autoencoder in training) and supervised (Deep Neural Network) learning to extract fault features from electrical power system (EPS) telemetry, as a way to identify fault states from the extraction of salient fault features in the data.[4] Carbone *et al.* develop a method for online monitoring of EPS data, which examines short circuits and sensor failures in telemetry using modeling in a Kalman Filter-based approach.[5] Daigle *et al.* also leverage a model-based approach based on residuals, which is an extension of the established Qualitative Event-based Diagnosis (QED) method.[6] Their work specifically examines a systems level approach to a subsystems (QED-PC - Possible Conflicts) approach, where they find pros and cons to each.

In application to attitude control system (ACS) of a spacecraft, Ahn *et al.* use a semi-supervised approach for anomalous fault detection, where a variational autoencoder (VAE) and Generative Adversarial Network model provides a means for producing reconstruction error values used to quantify norm deviation.[7] Gao *et al.* use Principle Component Analysis (PCA) for complexity reduction via feature extraction, which is fed into a binary Support Vector Machine (SVM) to classify faults. Lastly, fault vectors are processed through a multi-class SVM to determine the type of fault.[8]

While the reviewed methods provide a diverse landscape of viable methods for fault mitigation in satellite sub-systems, development in holisitic *system-level* spacecraft fault diagnosis remains an open area of research. Li *et al.* describe general data mining approaches to system level fault diagnosis, emphasizing the value of augmentation of established methods with autonomous methods. To this end, we present a framework for system-level fault diagnosis of satellite telemetry. To our knowledge, this proof-of-concept work is the first work which provides system-level, application agnostic methods for fault diagnosis, which is able to process both nominal and unforeseen faults in spacecraft.

PRELIMINARIES

We begin with a discussion of the preliminary artificial intelligence methods utilized by our system.

Kalman Filters

Kalman filters are dynamic linear estimation models able to provide a prediction from previously measured data. They were developed by Rudolf E. Kálmán in 1960, and are widely used in navigation related works such as self-driving vehicles. In this work, we leverage a Kalman filter to linearize data to a Gaussian curve and predict an expected mean (which we will call $\overline{X}$) and covariance (which we will call Σ) on satellite telemetry data. In this way, we can measure the level of error of a mnemonic (e.g., a measurement from a particular sensor), relative to itself (versus the whole data set), such that the predictions are:

$$\overline{X}' = A\overline{X}_{k-1} + W_{k-1}$$
$$\Sigma' = A\Sigma_{k-1}A^T + Q_{k-1}$$

Where A is the state transition matrix, W_k is the process noise, and Q_k is the covariance process noise. The state is measured as $y_k = H_k\overline{X}_k + r_k$ where H_k is the measurement model matrix and r_k is measurement noise.

As the updating process involves reshaping the Gaussian curve, this update step allows for the use of noisy sensor data. The updating process can be described as:

$$v_k = y_k - H_k$$
$$S = H_k \Sigma'_k H_k^T + R$$
$$K = \Sigma'_k H_k^T S^{-1}$$
$$\overline{X}_k = \overline{X}'_k + K v_k$$
$$\Sigma_k = \Sigma'_k - K S K^T$$

In previous studies, it has been shown Kalman filters are capable of error detection when taking the residual error of predicted vs. actual values (Carbone et al. 2019). The residual error will be used to determine if the value is an error, but a z-score will be used to determine the significance of the erring attribute where:

$$Z = \frac{x - \mu}{\sigma}$$

Autoencoders

Autoencoder(AE) Neural Networks are unsupervised representation learners. By enforcing a reconstruction loss between input and autoencoder output, the network learns an internal representation for our input data distribution over many samples. Maintaining an information bottleneck in our hidden layers leads to a rich representation where our network maps high-dimensional input data to a lower dimensional representation space. In practice, our network is split into encoder $f : \mathbb{R}^n \to \mathbb{R}^m$ and decoder networks $g : \mathbb{R}^m \to \mathbb{R}^n$ where $n > m$. Let $x \in \mathbb{R}^n$ be our data sample. We create a representation of our data $y = f(x)$. Let our reconstructed sample $x' = g(y) = g(f(x))$. Our reconstruction loss θ will be:

$$\theta = \mathbb{E}((x - x')^2),$$

which corresponds with the mean square error between our input and our reconstructed output.

Causality

Causal graphs provide a way to represent uncertain causalities based on Belief Networks.[9] Causal graphs include a few advantages. First, they are able to use either continuous or discrete input. For this work, we used continuous values as input to represent the sensor readings. They are also based on probability theory and have a strong theoretical foundation. The causal structure enables the ability to dynamically change the causal relationships in real-time. Finally, they have both directional and structural relationships. In causal graph $G = (E, V)$ each node $v \in V$ represents an event, or in case a sensor reading. Each edge $e \in E$ represents the direction and strength of the causal relationship. These relationships make causal graphs a well-suited method for diagnosing spacecraft telemetry faults.

We utilize the partial correlation algorithm described in Runge et al.[10] to compute the causal graph. Formally, consider the time series $X_t = \{X_t^1, X_t^2, ..., X_t^N\}$ with N datapoints of temporal length t with:

$$X_t^j = f_j(P(X_t^j), n_t^j)$$

where f_j is a nonlinear functional dependency and n_t^j represents mutually independent statistical noise. The nodes in the temporal causal graph represent the variable X_t^j at different time points and $P(X_t^j) \subset (X_{t-1}, X_{t-2}, ...)$ is the causal parents of X_t^j. A causal link $X_{t-\tau}^i \to X_t^j$ exists if $X_{t-\tau}^i \in P(X_t^j)$.

The PCMCI algorithm consists of two stages - (i) PC condition selection to identify $\hat{P}(X_t^j)$ where $X_t^j \in \{X_t^1, X_t^2, ..., X_t^N\}$ and (ii) the momentary conditional independence test (MCI) which tests the condition $X_{t-\tau}^i \to X_t^j$

$$MCI : X_{t-\tau}^i \not\perp\!\!\!\perp X_t^j | \hat{P}(X_t^j) \setminus \{X_{t-\tau}^i\}, \hat{P}(X_{t-\tau}^i)$$

Hence, MCI conditions on both the parents X_t^j and time-shifted parents $X_{t-\tau}^i$.

For the PC algorithm, for every variable X_t^j we initialize the preliminary parent $\hat{P}(X_t^j) = \{X_{t-1}, X_{t-2,...,X_{t-\tau_{max}}}\}$. During the initial iteration, if $H_0 : X_t^j \not\perp\!\!\!\perp X_{t-\tau}^i$ cannot be rejected at p_{PC}. The parents are then sorted by their test statistic value and then conduct conditional independence tests $X_t^j \perp\!\!\!\perp X_{t-\tau}^i | \mathbb{P}$ where $\mathbb{P}$ are the strongest parents in $\hat{P}(X_t^j) \setminus X_{t-\tau}^i$. After each iteration independent parents are removed from $\hat{P}(X_t^j)$, and the algorithm terminates if no more conditions can be tested. For testing in the MCI portion, $X_{t-2}^1 \to$ the conditions $\hat{P}(X_t^3)$ are sufficient to establish conditional independence.

THEORETICAL FRAMEWORK

We begin with a set of time-series data frames $\mathcal{F} = f_1, \dots f_{|\mathcal{F}|}$, representing all data frames in a flight mission from a starting time step 1, to ending time step $|\mathcal{F}|$ (see Table 1 for illustrative example). The finite set of input frames $\mathcal{F}_{a,b} = f_a, \dots f_b$ represents data frames of telemetry from time step a to b of the mission associated with $\mathcal{F}$. Each frame $f_i \in F$ occurring at time step i is composed of a finite set of continuous telemetry mnemonics $\mathcal{M} = m_1, \dots m_{|\mathcal{M}|}$. Each telemetry mnemonic has a range of permissible

values, and a range of faulting values. That is, for any mnemonic m_j, we define $\underline{m_j}$ and $\overline{m_j}$ as the lower and upper bounds of m_j, respectively. We consider value assignments for a given mnemonic which fall within the permissible range $[\underline{m_j}, \overline{m_j}]$ to be *nominal* assignments, and values outside of that range to be *fault* assignment. A spacecraft is in a `GREEN` status if all of its telemetry readings are nominal, and is in a `RED` status if one or more telemetry readings have a faulting value.

Problem Formulation

Suppose a spacecraft is in flight, with a `GREEN` mission status. Now, suppose that sometime during the spacecrafts flight, it encounters a scenario in which one or more of its telemetry mnemonics enters a faulting state, eliciting `RED` mission status. We assume that given a set of faulting mnemonics causing a `RED` mission status, there is a corresponding set of mnemonics which are responsible for the root cause of the symptomatic faulting mnemonics. We assume that these root cause mnemonics can be found using a diagnosis process which operates over the faulting mnemonics and a set of data frames. In this research, we simplify the diagnosis process to only consider a single root cause mnemonic instead of a full set of coordinated causes.

That is, given set of faulting mnemonics $\mathcal{S} = \{m_i, \dots m_j\}$ (symptomatic set), there exists a root cause mnemonic $r \in \mathcal{M}$ (not necessarily $r \in \mathcal{S}$) which can be found through a diagnosis process $\mathcal{D}$ such that $\mathcal{D} : (\mathcal{S}, \mathcal{F}_{a,b}) \mapsto r$. Note that, while the mapping $\mathcal{D}$ is one to one, it is not guaranteed that a set of symptoms maps to a unique cause, without consideration of a set of past data frames $\mathcal{F}_{a,b}$, representative of near-past flight.

Illustrative Example: Table 1 shows a simple flight sample of toy data, where there are three sets of faulting frames. The first fault occurs at f_3, where the current (m_3) drops to value outside of the lower threshold value, quickly recovering back to its original value in the next frame. This was caused by a faulty sensor reading, and thus the cause and symptom are one in the same (*SYMPTOM = current*, *CAUSE = current*). We discuss the implications of this fault in the next section. The next fault occurs at f_6 - f_9, triggered from a threshold breach in temperature. In this case, the increased acceleration of the spacecraft causes the skin temperature of the spacecraft to increase, which is expected behavior. Thus, in this case, (*SYMPTOM = temperature*, *CAUSE = acceleration*). The last fault is triggered at f_{13} - f_{15}, due first to a breach in threshold from the temperature, followed by an additional breach from current. In this case, the temperature increases rapidly, unlike its gradual increase due to spacecraft speed. Here, one of the batteries has unexpectedly caught fire, triggering an increase in temperature, and the eventual threshold breach in the current due to a dead battery. Thus, (*SYMPTOM = temperature, current*, *CAUSE = current*). The notation for these scenarios are shown below:

	$\mathbf{m_1}$	$\mathbf{m_2}$	$\mathbf{m_3}$
mnemonic	acceleration	temperature	current
$[\underline{\mathbf{m_j}}, \overline{\mathbf{m_j}}]$	[99,114]	[80, 121]	[10,23]

time step	$\mathbf{m_1}$	$\mathbf{m_2}$	$\mathbf{m_3}$	**status**
$\mathbf{f_1}$	100 m/s^2	97 °F	21 mA	`GREEN`
$\mathbf{f_2}$	100 m/s^2	97 °F	21 mA	`GREEN`
$\mathbf{f_3}$	100 m/s^2	99 °F	5 mA	`RED`
$\mathbf{f_4}$	109 m/s^2	99 °F	21 mA	`GREEN`
$\mathbf{f_5}$	110 m/s^2	106 °F	22 mA	`GREEN`
$\mathbf{f_6}$	110 m/s^2	115°F	22 mA	`RED`
$\mathbf{f_7}$	110 m/s^2	118 °F	21 mA	`RED`
$\mathbf{f_8}$	110 m/s^2	120 °F	21 mA	`RED`
$\mathbf{f_9}$	105 m/s^2	115 °F	21 mA	`RED`
$\mathbf{f_{10}}$	101 m/s^2	110 °F	22 mA	`GREEN`
$\mathbf{f_{11}}$	100 m/s^2	97 °F	21 mA	`GREEN`
$\mathbf{f_{12}}$	100 m/s^2	114 °F	21 mA	`GREEN`
$\mathbf{f_{13}}$	100 m/s^2	125 °F	21 mA	`RED`
$\mathbf{f_{14}}$	100 m/s^2	151 °F	4 mA	`RED`
$\mathbf{f_{15}}$	100 m/s^2	170 °F	2 mA	`RED`

Table 1: Illustrative sample a flight mission. The top table shows upper and lower thresholds of each mnemonic m_i (indicated $\underline{m_i}$ and $\overline{m_i}$, respectively). The bottom table shows 15 time steps expressed as time frames $f_1, \dots f_{15}$, with their corresponding status values, indicating a threshold breach of one or more mnemonics in the frame.

$$\mathcal{D} : (\{m_3\}, \mathcal{F}_{3,3}) \mapsto m_3$$
$$\mathcal{D} : (\{m_2\}, \mathcal{F}_{6,9}) \mapsto m_1$$
$$\mathcal{D} : (\{m_2\}, \mathcal{F}_{13,13}) \mapsto m_3$$
$$\mathcal{D} : (\{m_2, m_{13})\}, \mathcal{F}_{14,15}) \mapsto m_3$$

Types of Faults

Our system captures three major fault types, shown illustratively in Figure 1. The first fault type is an `ISOLATED_FAULT`, which broadly encompasses faults that are a result of bit flips from cosmic radiation (referred to as single event upsets, or "SEUs"), or

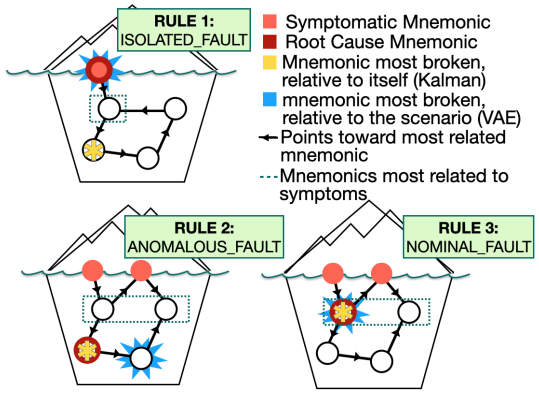

Figure 1: Types of faults rendered by diagnosis algorithm. We adopt an "iceberg" analogy, where "at the tip of the iceberg," above the water, symptoms are realized. Below the surface, the candidate root causes live within the body of the iceberg. Note that each of these images provide one of many possible illustrative examples of each kind of fault.

broken sensors. These faults can be thought of as "spoofs," where nothing is actually operationally wrong with the spacecrafts function. In our illustrative example, the fault occurring at $F_{3,3}$ would be considered an isolated fault. Next, we consider a `KNOWN_FAULT`, which encompasses faults which *have happened in the past* in some capacity. For example, a jammed reaction wheel is a well known culprit responsible for navigation related faults. These sorts of faults are ones which, in theory, could be preemptively engineered into onboard fault logic. In our illustrative example, the fault occurring at $F_{6,9}$ would be considered an a known fault. Lastly, and perhaps most importantly, we consider `ANOMALOUS_FAULTS`, which encompass faults that could *not* be preemptively engineered into onboard fault handling constructs. The novelty of this scenario is explained by the salience presented in a particular fault-telemetry-environment combination. Note that some faults can have multiple characterization assignments based on our described classification system. In our illustrative example, the fault occurring at $F_{13,15}$ would be considered an an anomalous fault.

Diagnosis Algorithm

We describe our diagnosis system, shown in Algorithm 1 and Figure 2. The inputs to our algorithm are the set of faulting mnemonics, a window of frames used for diagnosis, and a static list of ordered telemetry mnemonics to establish consistency in algorithmic calculations (lines 1 - 4). Pre-processed elements are shown on lines 5 - 6, where n represents the number of total mnemonics per frame, and n' gives an upper bound ranking to mnemonics considered from our main artificial intelligence constructs, shown on lines 8 - 10. The *Kalman* construct (set of mnemonic names and corresponding normalized z-score values; line 8) represents how a particular telemetry mnemonic is performing relative to its own individual past history of performance, independent of other data points in the frame. The *AE* construct (set of mnemonic names and corresponding normalized reconstruction error values; line 9) represents how a mnemonic is performing over a small window of time, relative to the other mnemonics in the frames. Lastly, the *Causality* construct (set of mnemonic names and corresponding $n \times n$ matrix of row-normalized association values; line 10) represents the relatedness of mnemonics to one another, over a small window of time. The Kalman construct shows mnemonic-specific data over the life of the mission, whereas AE and causality show context-specific holistic frame data over a smaller window of time.

Algorithm 1 Ensemble-based Diagnosis Algorithm

1: $\mathcal{S}$: set of faulting mnemonics
2: $\mathcal{F}_{a,b}$: frames used for diagnosis
3: $\odot$: ordered telemetry mnemonics
4: $\star$: bounds for ordered telemetry mnemonics
5: $n \leftarrow len(f_i \in \mathcal{F}_{a,b})$
6: $n' \leftarrow roundInt(\sqrt{n})$
7: $\mathcal{M}_\mathcal{S} \leftarrow \mathcal{S}$
8: $\mathcal{M}_\mathcal{K} = (\odot_\mathcal{K}, \star_\mathcal{K}) \leftarrow Kalman(\mathcal{F}_{a,b})$
9: $\mathcal{M}_\mathcal{V} = (\odot_\mathcal{V}, \star_\mathcal{V}) \leftarrow AE(\mathcal{F}_{a,b})$
10: $\mathcal{C} = (\odot_\mathcal{C}, \star_\mathcal{C}) \leftarrow Causality(\mathcal{F}_{a,b})$
11: **if** $|\mathcal{M}_\mathcal{S}| = 1 \wedge \mathcal{M}_\mathcal{S} = top_1[\mathcal{M}_\mathcal{V}]$ **then** ▷ Rule 1
12: $\quad \hat{m} \leftarrow \mathcal{M}_\mathcal{S}$
13: $\quad$ **return** `ISOLATED_FAULT`, $\hat{m}$
14: **else**
15: $\quad \hat{m} \leftarrow top_1[\mathcal{M}_\mathcal{K}]$
16: $\quad C' \leftarrow []$
17: $\quad$ **for** $s \in \mathcal{M}_\mathcal{S}$ **do**
18: $\quad\quad C' \leftarrow C' \cup top_1[C(s)]$
19: $\quad$ **end for**
20: $\quad$ **if** $\hat{m} \in [\mathcal{M}_\mathcal{V}] \vee \hat{m} \in C'$ **then** ▷ Rule 2
21: $\quad\quad$ **return Walkdown**$(\mathcal{M}_\mathcal{S}, \mathcal{C}, n, \odot_\mathcal{K}, \odot)$
22: $\quad$ **else**
23: $\quad\quad$ **return** `ANOMALOUS_FAULT`, $\hat{m}$ ▷ Rule 3
24: $\quad$ **end if**
25: **end if**

Rule 1 (lines 11-13) states that, if there is only *one* isolated faulting mnemonic $\hat{m}$, and that faulting

mnemonic is the same as the top-1 AE mnemonic, we return an `ISOLATED_FAULT`, with $\hat{m}$ as the responsible value. We offer the following interpretation: *if the only symptom of the fault is the same as the mnemonic which is contributing the most to the overall error, than it is likely the root cause as well.* This fault is likely in the form of a single event upset, a bit-flip, or a faulty sensor reading.

Rule 2 (lines 15-21) states that, if the top-1 faulting Kalman mnemonic $\hat{m}$ is a top-n' fault in the AE, *or* it is in the list composed of the mnemonics which are most related to each of the individual faulting mnemonics in $\mathcal{M}_\mathcal{S}$ (line 20), then perform the *walkdown* procedure to find the root cause. We offer the following interpretations: *if the mnemonic which is breaking the most individually is either (1) contributing largely to the overall error or (2) is directly related one of the symptoms, than it is likely also a symptom.* Therefore, the algorithm defaults to the walkdown method to find the root-cause.

The **Walkdown Method** (Algorithm 2) operates over the faulting mnemonics (symptoms), and the causality matrix. An initial candidate root cause $\hat{m}$ is generated by adding together the normalized causality vectors of the top faulting mnemonics (lines 6-9), and returning the mnemonic with the highest corresponding value (line 10). In this way, we capture the mnemonic which is most highly related to all symptoms. If this candidate value is breaking its own Kalman value (line 13), then return a `KNOWN_FAULT`, with $\hat{m}$ as the responsible mnemonic. However, if the candidate *is not* breaking its Kalman, consider the mnemonic with its top-1 most related to $\hat{m}$ as the new candidate root cause, and repeat the Kalman check criterion (lines 12-18). If no root cause is found in this traversal, an `ISOLATED_FAULT` is returned (line 19). Faulting mnemonics (line 11) and visited mnemonics (line 16) are not considered in the search. We offer the following interpretation: *if the value which is most highly related to all faulting mnemonics has substantial individual error, than its likely a fault which is not anomalous. If this candidate value does not have substantial individual error, find the next most related mnemonic to this candidate, and repeat the Kalman check.*

Rule 3 (line 23) states that, if the top-1 faulting Kalman mnemonic $\hat{m}$ is not a top-n' fault in the AE, *and* is not in a list composed of the mnemonics which are most related to each of the individual faulting mnemonics in $\mathcal{M}_\mathcal{S}$, an `ANOMALOUS_FAULT` is returned, with $\hat{m}$ as the responsible value. More simply, this case considers the high individual error candidates described in Rule 2 (line 15) to be root-cause candidates, instead of symptoms. We offer the following interpretation: *If the mnemonic with the highest individual error is not contributing to overall error and is not related to any symptoms, it is not considered a candidate symptom. Instead, its high individual error indicates it is a root cause.*

Algorithm 2 Walkdown Diagnosis Algorithm

1: $\mathcal{M}_\mathcal{S}$: faulting mnemonics
2: $\mathcal{C} = (\odot_\mathcal{C}, \star_\mathcal{C})$: causality matrix
3: n : number of mnemonics per frame
4: $\odot_\mathcal{K}$: telemetry mnemonics which break their Kalman value
5: $\odot$: static list of telemetry mnemonic names
6: $D \leftarrow zeros[n]$
7: **for** $s \in \mathcal{M}_\mathcal{S}$ **do**
8: $\quad D = D + \star_\mathcal{C}[s]$
9: **end for**
10: $\hat{m} \leftarrow top_1[(\odot, D)]$
11: $E \leftarrow \mathcal{M}_\mathcal{S}$
12: **while** $|E| < n$ **do**
13: $\quad$ **if** $\hat{m} \in \odot_\mathcal{K}$ **then**
14: $\quad\quad$ **return** `KNOWN_FAULT`, $\hat{m}$
15: $\quad$ **end if**
16: $\quad E = E \cup \{\hat{m}\}$
17: $\quad \hat{m} \leftarrow top_1[\mathcal{C}(\hat{m})]$
18: **end while**
19: **return** `ISOLATED_FAULT`, $\emptyset$

EXPERIMENTAL RESULTS

We tested our system in two main settings. The first setting was a software-only proof-of-concept (POC) setting, where we tested the accuracy and success of our diagnostic algorithm on three data sets with known (or in a special case, inferred) ground truth values. The second setting was a hardware-in-the-loop setting, in order to evaluate on-board tractability testing.

Diagnostic Accuracy Testing

The first portion of our testing evaluates the accuracy of our diagnostic system against baseline methods.

Methods: We tested our system on three data sets of varying complexity levels, across three separate experiments. Each data set included a set of time-synced frames of spacecraft telemetry, along with their corresponding fault criterion, in the form of thresholds, or allowable state-based values. Telemetry testing was simplified by converting all tests into threshold-based testing in the following manner: *Continuous telemetry values* were converted to decimal

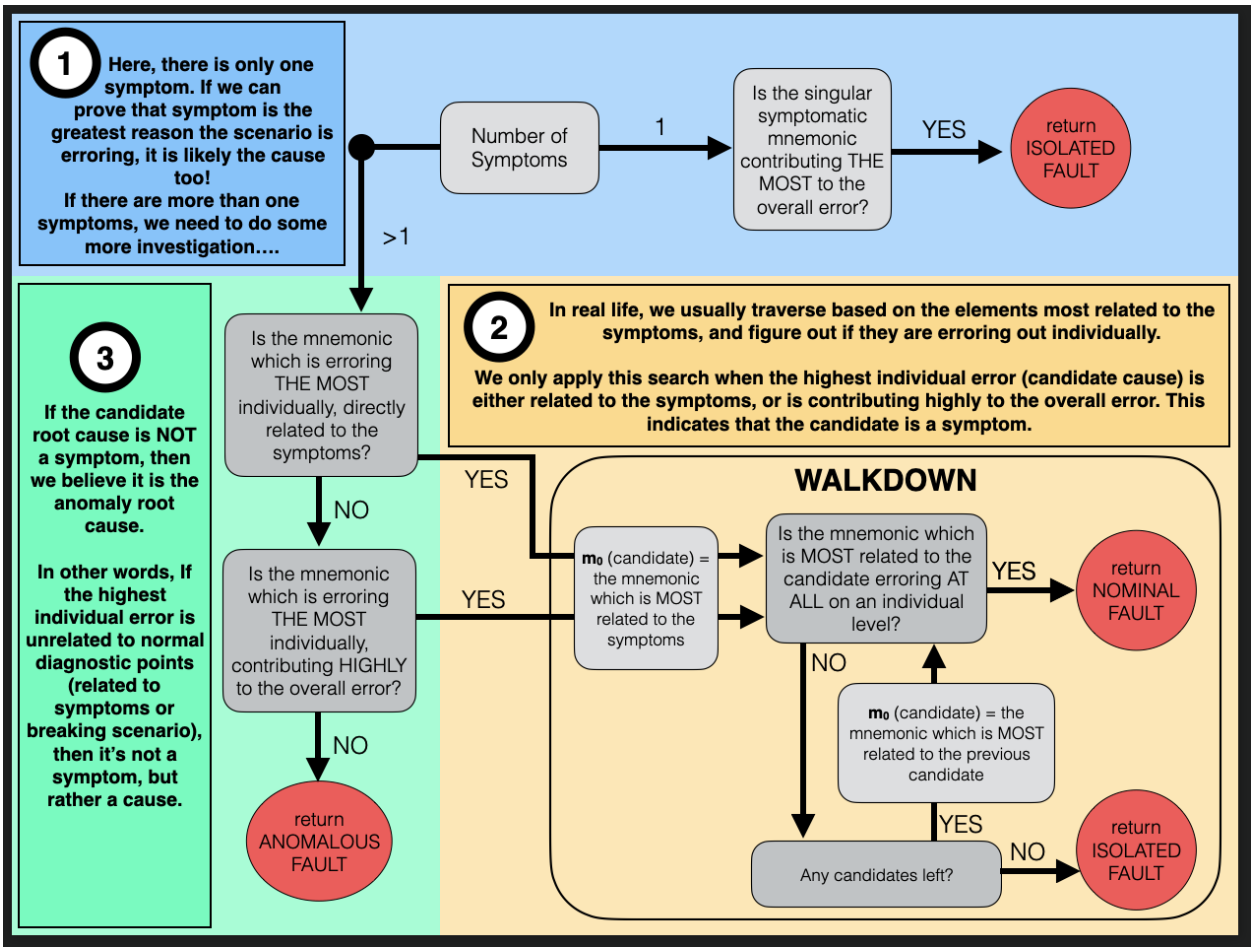

Figure 2: System diagram of fault diagnosis process

Exp #	# Missions	[min, max], sd	# mn
1	11	[1440,1440],0	8
2	10	[651,2028],554.9	17
3	1	[6448,6448],0	33

Table 2: Mission information for each experiment is shown, including number of total flights (# Missions), the min, max and standard deviation of mission length in time steps ([min, max],sd) and the total number of telemetry mnemonics per frame in the data sets (# mn).

values and used in threshold-based testing, and *discrete state based mnemonics* were converted to representative integer values, which were then converted into decimal equivalent values.

Prior to diagnosis testing, we trained the auto encoder using a data split on our available samples. Splits were made to provide 50% fault data, and 50% non-faulting data. After training, testing data sets were inputted into our diagnostic algorithm through a continuous data stream. In this case, our algorithm used the threshold-based testing to detect/trigger faults, at which point diagnosis would take place to render a root cause. In addition to overall accuracy, we test our algorithm against established baselines. We performed comprehensive hyperparameter tuning as well as ablation testing of each component trained individually.

Data Sets: We ran our system on three data sets, of varying (increasing) complexity levels, shown in Table 2. Data set 1 ("Toy Data Set") consisted of 11 flights of basic spacecraft housekeeping and physics data. Missions reflected a broad range of faults, tightly generated with known ground truth values. Examples of faults include the following – "voltage sensor broken" (`ISOLATED_FAULT`), "thruster broke max capacity" (`ANOMALOUS_FAULT`), and "unsafe altitude" (`KNOWN_FAULT`). Data set 2 ("Simulation Generated Data Set") consisted of 10 flights of sounding rocket data, generated from the Kerbal Space Program flight simulation, with self-generated known ground truths. Examples of faults include "pressure SEU" (`ISOLATED_FAULT`), "communication drop" (`ANOMALOUS_FAULT`), and "thrust caused crash to earth" (`KNOWN_FAULT`). Data set 3 ("Sounding Rocket Data Set") consisted of 1 flight of a real sounding rocket with a known ground truth value. The fault was a known fault, related to a threshold break in the skin temperature of the rocket due to an increased speed.

Results: Overall, Experiment 1 had a 0.77 accuracy rate for diagnosing the correct mnemonic responsible for faults. Of the correctly diagnosed faults, our system was able to successfully classify the *type* of fault in 70% cases. Classification was particularly successful for SEU's and anomalies. Surprisingly, our system mis-classified 2 of 9 faults as anomalous faults instead of nominal faults. This is not of acute concern, as the priority in this work is a correct mnemonic diagnosis. There were two missions where classification was not fully successful. The first is a mission where a current jump is caused by a science instrument onboard. Our system ultimately performed a walkdown method, but was unable to find a root cause, thus rendering `NO_DIAGNOSIS`. In this case, the science instrument readings were boolean indicators representing whether the instrument was on or not, and therefore we postulate that the simplicity of the readings was never able to break its own Kalman filter, which would have stopped the walkdown traversal. The other mission that was inaccurately diagnosed was one in which the spacecraft caught on fire (Figure 3(b)). We believe that a diagnosis was challenging, because the cause was external to the satellite telemetry set. An interesting observation from the data shown in Figure 3(a,c) is that given two missions that appear to be similar to the human eye, classification was accurate at detecting the two very separate root causes for faults. The first (a) was classified as a `NOMINAL_FAULT`, and the second (c) was classified as an `ISOLATED_FAULT`.

Overall, Experiment 2 had a 0.7 accuracy rate for diagnosing the correct mnemonic responsible for faults. Of the correctly diagnosed faults, our system was able to successfully classify the type of fault in 71% cases. Classification was successful for anomalies, and most isolated fault cases (see Figure 3(d-e) for examples of simple SEU faults). Among successfully diagnosed cases, there were no nominal faults. There were three missions where classification was not fully successful. The first is a mission where change in velocity caused a communication drop (Figure 3(f)). The algorithm instead diagnosed the change in the vessel's velocity as the ground truth, which we consider a nearly accurate classification. The other missions were a mission where changes in pressure caused a navigation SEU, and a mission where a decoupling on board caused a communication drop.

Experiment 3 was only composed of one mission. The ground truth diagnosis for the nominal fault in skin temperature caused by increased speed was the Z acceleration value. Our algorithm consistently diagnosed the Z magnetometer reading. We believe that the algorithm was unable to catch Z acceler-

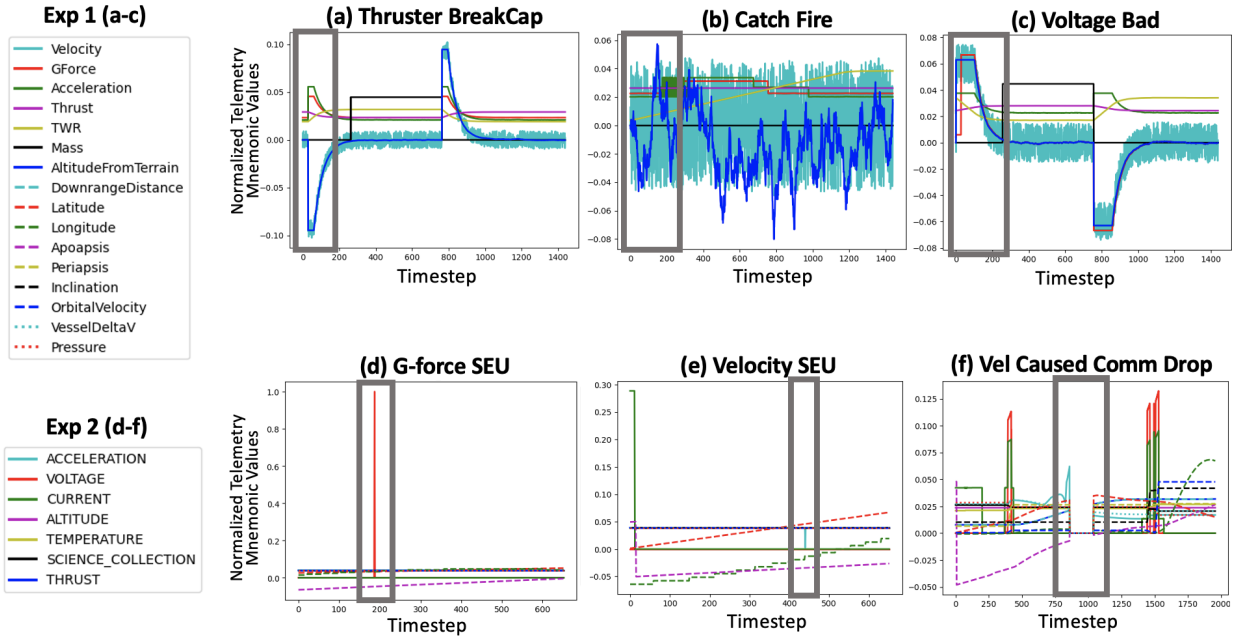

Figure 3: Shows three missions for Experiment 1 (top row) and three missions for Experiment 2 (bottom row). Missions depicted in (a,c,d,e) were diagnosed currently by the fault diagnosis method developed in this chapter. (a) Shows a mission where thrusters broke their expected capacity limits for flight, inducing symptoms in the voltage and current (symptom: VOLTAGE,CURRENT, cause:THRUSTER). (b) Shows a mission where a component of the spacecraft was pulling too much current and caught on fire, setting the entire craft into flames (symptom: TEMP, cause: CURRENT). (c) Shows a mission where the voltage readings were incorrect in an isolated fault (cause: VOLTAGE, symptom: VOLTAGE). (d) Shows a mission where the G-Force encountered a minor bit flip in the form of an SEU (symptom: G-Force, cause: G-Force). (e) Shows an event similar to (d), where the velocity reading temporarily flickers to the wrong value (symptom: Velocity, cause: Velocity). (f) Shows a mission where the velocity causes a temporary loss in communication with ground control (symptom: Velocity, cause: Velocity).

Ex #	Acc	PPO_1	PPO_2	AE_1	AE_2	WD
1	0.77	0.03	0.04	0.35	0.60	0.64
2	0.70	0	0	0.48	0.53	0.3

Table 3: Average accuracy of diagnosis compared to known ground truths. The compared accuracy of diagnosis, as seen in the above table, differentiates between methods such as the top one-two diagnoses of PPO, the top one-two diagnoses of AE, and the diagnosis of the walkdown method; these models are being compared to the accuracy of the ensemble approach.

Construct	10 mpf	50 mpf	100 mpf	200 mpf
Kalman	0.237	0.351	0.581	0.831
AE	0.211	0.589	1.05	1.98
Causality	0.845	4.5	5.05	5.42

Table 4: Average milliseconds (ms) per frame across 100 runs. We performed an ablation study on the AI components in the framework, where each isolation component is tested for an average number of ms per frame of processing time across 100 runs. This study is run against frame sizes composed of 10, 50, 100 and 200 mnemonics per frame (mpf).

ation as the root cause due to its sporadic behavior during the fault, whereas the magnetometer was consistently symptomatic to the change as well.

Baseline Tests: In addition to accuracy testing, we compared our experimental results to baseline methods, shown in Table 3. We tested each batch experiment against a reinforcement learning (RL) based Proximal Policy Optimization (PPO) algorithm, and against a standalone autoencoder (AE) approach. In both cases, we put a large amount of effort into tuning and optimizing the methods for success. Even so, our algorithm outperformed the top ranking baselines by 28% and 32% for experiments 1 and 2, respectively. No baseline testing was performed on Experiment 3.

Tractability Testing

In addition to our proof-of-concept (POC) software testing, we ran our algorithm on a Raspberry Pi 3 B+ with a 1.4GHz quad-core ARM Cortex-A53 and 1 GB of RAM. This hardware is representative of a realistic flight hardware test bed and allows us to perform tractability testing for on-board diagnostic capability. As a part of an ablation approach, we isolated performance tests on individual components of the framework to find average run time per frame across 100 runs, shown in Table 4. We used Python's time module to generate timestamps. Each component was tested against 10, 50, 100, and 200 mnemonics per frame (mpf). We consider growth rate of performance against growth from 10 mpf to 200 mpf (20 times the number of mnemonic per frame). The top performing component overall was the Kalman Filter, which performed only 0.026ms worse than the AE initially at 10 mpf, but had the smallest growth rate as the number of mnemonics increased. Causality performed most poorly overall, with an initial processing time of 0.845ms for 10 mpf (4 and 3.57 times the performance of the AE and Kalman, respectively), and a 6.414 growth rate, where it was outperformed at 200mpf by Kalman and AE by 6.52 and 2.74, respectively. Surprisingly, this growth rate was lower than that of AE, which had a growth rate of 9.38. It should be noted that a benefit of causality and Kalman constructs is that they do not require training.

In future work, we plan on optimizing our algorithm in preparation for a live flight by reducing the matrix size rendered by the causality construct. Currently, for frame size n, our algorithm renders a $n \times n$ matrix of associations at diagnosis time. This can be optimized by requesting causality vectors as they are needed in the walk down traversal, which would provide a guaranteed optimization. The walkdown algorithm excludes queries on symptomatic mnemonics, which reduces the search to $(n-k) \times (n-k)$ for k symptom mnemonics. In order to trigger a diagnosis, at least one mnemonic must be faulting, and therefore $k \geq 1$. Note that query time is constant and would only contribute a linear growth factor.

CONCLUSION

We presented a novel and innovative method for spacecraft fault diagnosis using telemetry data. We tested our method with three data sets of varying complexity levels, and of various domain representation levels. We compared our results with alternative artificial intelligence methods in baseline testing, and performed an ablation-based experiment in a realistic hardware flight test bed. This fault diagnosis work contributes research toward CPS diagnosis. In the following chapter, we shift focus toward the reaction portion of the CPS continuum, presenting a generalized framework for action discovery in robots.

REFERENCES

[1] Tianshe Yang, Bin Chen, Yu Gao, Junhua Feng, Hailong Zhang, and Xiaole Wang. Data mining-based fault detection and prediction methods for in-orbit satellite. In *Proceedings of 2013 2nd International Conference on Measurement, Information and Control*, volume 2, pages 805–808. IEEE, 2013.

[2] Aboul Ella Hassanien, Ashraf Darwish, and Sara Abdelghafar. Machine learning in telemetry data mining of space mission: basics, challenging and future directions. *Artificial Intelligence Review*, 53(5):3201–3230, 2020.

[3] Yixing Wang, Meiqin Liu, and Zhejing Bao. Deep learning neural network for power system fault diagnosis. In *2016 35th Chinese control conference (CCC)*, pages 6678–6683. IEEE, 2016.

[4] Hongzheng Fang, Hui Shi, Yunfan Dong, Huanzhen Fan, and Shuai Ren. Spacecraft power system fault diagnosis based on dnn. In *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, pages 1–5. IEEE, 2017.

[5] Marc A Carbone, Jeffrey T Csank, Brian J Tomko, Jeffrey C Follo, and Matthew J Muscatello. A multiple model based approach for deep space power system fault diagnosis. 2019.

[6] Matthew Daigle, Anibal Bregon, and Indranil Roychoudhury. Qualitative event-based diagnosis with possible conflicts applied to spacecraft power distribution systems. *IFAC Proceedings Volumes*, 45(20):265–270, 2012.

[7] Hyojung Ahn, Dawoon Jung, and Han-Lim Choi. Deep generative models-based anomaly detection for spacecraft control systems. *Sensors*, 20(7):1991, 2020.

[8] Yu Gao, Tianshe Yang, Nan Xing, and Minqiang Xu. Fault detection and diagnosis for spacecraft using principal component analysis and support vector machines. In *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1984–1988. IEEE, 2012.

[9] Qin Zhang. Probabilistic reasoning based on dynamic causality trees/diagrams. *Reliability Engineering & System Safety*, 46(3):209–220, 1994.

[10] Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5(11):eaau4996, 2019.