UNIVERSITY OF WASHINGTON

Techniques for Fault Detection and Visualization of Telemetry Dependence Relationships for Root Cause Fault Analysis in Complex Systems

by

Nathaniel Guy

A thesis submitted in partial fulfillment for the
degree of Master of Science in Aeronautics & Astronautics

in the

William E. Boeing Department of Aeronautics & Astronautics
University of Washington College of Engineering

February 2016

Declaration of Authorship

I, NATHANIEL GUY, declare that this thesis titled, 'TECHNIQUES FOR FAULT DETECTION AND VISUALIZATION OF TELEMETRY DEPENDENCE RELATIONSHIPS FOR ROOT CAUSE FAULT ANALYSIS IN COMPLEX SYSTEMS' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Okay, 13. We've got lots and lots of people working on this; we'll give you some dope as soon as we have it, and you'll be the first one to know.'

Jack Lousma, Apollo 13 CapCom [22]

UNIVERSITY OF WASHINGTON

Abstract

William E. Boeing Department of Aeronautics & Astronautics
University of Washington College of Engineering

Master of Science in Aeronautics & Astronautics

by Nathaniel Guy

This thesis explores new ways of looking at telemetry data, from a time-correlative perspective, in order to see patterns within the data that may suggest root causes of system faults. It was thought initially that visualizing an animated Pearson Correlation Coefficient (PCC) matrix for telemetry channels would be sufficient to give new understanding; however, testing showed that the high dimensionality and inability to easily look at change over time in this approach impeded understanding. Principal component analysis (PCA) was used to reduce dimensionality, and the time curve visualization proposed by Bach et al (2015) was adapted to visualize both raw telemetry and telemetry data correlations. Subsequent testing revealed insights into understanding and an intuitive grasp of data families that suggests the viability of this approach to enhance root cause analysis for actual aerospace systems.

Acknowledgements

I would like to thank my advisor, Dr. Mehran Mesbahi, for his academic guidance, his patience, and his constant encouragement as I struggled with the process of determining my own interests in this field. Deep thanks go to Dr. Jeff Heer for his insights and suggestions into the best practices for visualizing the hidden patterns within correlation data. My sincere thanks to Dr. Chris Lum for his opinions and suggestions about aerospace fault detection systems. I'd also like thank my various internship teams for their guidance and encouragement: the JPL OpsLab team, specifically Scott Davidoff, Jeff Norris, and Garrett Johnson, for introducing me to teleoperation interface design and testing techniques with the Unity 3D game engine; the SpaceX Flight Software team, specifically Mike Soares, Dan Gelband and Derek Bronish, for introducing me to Fault Detection, Isolation and Recovery systems and teaching me much about aerospace ground software systems; and the HAKUTO Lunar XPRIZE team, specifically Dr. Nathan Britton, Louis Burtz, Kurai Shimizu, Toshiro Shimizu, Toshiki Tanaka, Dr. John Walker, and Dr. Kazuya Yoshida for letting me use their lunar rover as a testbed for new ground software design methodologies and fault diagnosis techniques. Special thanks go to Daisuke Kikuchi for countless bits of design advice and for helping to increase the aesthetic appeal of the HAKUTO rover ground station UI. My thanks to the team at Unity, for developing an excellent and adaptive game engine. Finally, I'd like to thank Steve Rabin at Nintendo of America for his advice about fault detection and valuable comparisons to runtime analysis and profiling of executable code. Without the aforementioned people and many others, I would be lost!

Contents

List of Figures

List of Tables

Abbreviations

FD	**F**ault **D**etection
FDIR	**F**ault **D**etection, **I**solation, and **R**ecovery
GSN	**G**round **S**statio**n**
PCC	**P**earson **C**orrelation **C**oefficient

Physical Constants

Speed of Light $c = 2.997\ 924\ 58 \times 10^8\ \mathrm{ms}^{-\mathrm{S}}$ (exact)

Symbols

σ_X standard deviation of vector X

For Carl, who showed me the way to space.

Chapter 1

Introduction

Complex, remote-operated systems present many problems to engineers and designers who are conscious of mission safety. Complex systems often have detailed and comprehensive rules for the detection of anomalous conditions, or "faults," but even the most complex cannot capture the full range of possible anomalies that can occur on a system, especially if false positives are a concern and if the user has not thought of all possible conditions. Visualization of fault conditions can be an even greater problem, owing to issues such as the impracticality of simultaneously displaying data from thousands of telemetry channels, organizing data in a logical and discoverable way, maintaining system reliability in the presence of performance constraints, and leaving screen space for other non-fault-related visualization components and control affordances.

Because of these difficulties, root cause analysis can be a very long and difficult task. There are several historical cases of major system anomalies that have required very long periods of concentrated, manual telemetry data analysis (and, in some of the most catastrophic cases, post-disassembly hardware analysis) in order to piece together the root cause for anomalies. Some examples include:

- On October 28th, 2014, the Orbital Sciences "Antares" rocket suffered a catastrophic failure 6 seconds after launch, experiencing a large explosion which destroyed its cargo, which had been bound for the International Space Station. Orbital Sciences immediately launched an investigation to determine the cause, but preliminary root cause data loosely linking the mishap to a failure of one of the AJ26 engines was not publicly indicated until November 5th, and a final root cause assessment has still not, to this date, been released [2]. An independent review team within NASA evaluated telemetry data, historical data and hardware samples, beginning in November 2014, and roughly a year later, issued a report that

was still unable to provide a clear root cause for the mishap, instead linking it to three likely causes, all involving the AJ26 engine which initially exploded [20].

- On July 28th, 2015, SpaceX's "Falcon 9" rocket, carrying the "CRS-7" payload delivery up to the International Space Station, experienced an overpressure event in the second stage liquid oxygen tank, causing rapid unscheduled disassembly and failure of the mission. SpaceX engineers, working with NASA and the Air Force, began intensively examining system telemetry from the event. Despite SpaceX's well-known history of transparency about anomalous events and engineering challenges, the root cause of flawed second-stage helium system strut was not publicly identified until nearly a month later, on July 20th [19].

- On December 4rd, 2015, the PROCYON interplanetary cubesat, developed by the University of Tokyo and JAXA, went completely "dark," ceasing to provide any telemetry data at all. As of February of 2016, attempts to analyze previously received telemetry data in order to gain insight into the cause of the anomaly, and possible ideas for recovery, continue, but no root cause has been able to be determined [15].

As is shown by the cases above, the root cause diagnosis process is very difficult, and can take weeks to months to complete. It involves intense scrutiny of potentially thousands of data channels, and often the only comprehensive understanding of how these data channels relate to each other is encoded in human "tribal knowledge." Although the examples above are extreme ones, root cause diagnosis can be extremely valuable even with trivial anomalies for gaining a better understanding of system properties and subsystem connections, and the tools to do so that are currently in use are inadequate for the task. Having seen this problem first-hand in the space industry, we set out to examine the space of possible tools that could begin to tackle this problem.

In this thesis, we examine some possible solutions to the long-standing problem of root cause analysis for complex, remotely monitored systems. The paper starts with identification of irregular telemetry via a typical fault detection models, and describes the necessity to characterize anomalous conditions in a more in-depth way than traditional fault detection algorithms allow, in order to identify root causes for anomalies, or to predict the occurrence of anomalous conditions ahead of time. We will point out some of the issues that commonly occur with time series data on multiple channels which can make it difficult to both analyze and visualize.

Next, we will examine traditional methods of analyzing connections between sets of time series data. We will see how solutions to some of the aforementioned issues are

provided by these analytical techniques. We will also examine a number of visualization techniques that have been used in the past to show connections within correlation data.

We will propose a data visualization technique, based on an animated adaptation of statistical correlation matrices. To assess the efficacy of this visualization, we will apply it to simulated data from a sample system, and will show test results when users are given an implementation of this visualization and asked to use it to gain insight into events during a simulated scenario. We will discuss some of the downsides of our techniques that were uncovered by testing.

Next, we will iterate and propose adaptations to address some of the issues in the first round of testing. We will look at analytical improvements as well as new visualizations, borrowing from recent research in the field of temporal data visualization. We will then run another set of human tests on this data to show the effect on user insight.

Finally, we will present our conclusions about the employed analysis and visualization techniques, and will propose avenues for further research.

Chapter 2

Background in Fault Detection, Isolation and Recovery

In this chapter, we will discuss the theoretic fundamentals of Fault Detection, Isolation and Recovery (hereafter, "FDIR"). We will look at how behavior of complex systems can be modeled in such a way that undesirable states can be clearly defined and detected. We will look at more modern, advanced techniques for this analysis. We'll also spend some time talking about analytical as well as human-centered issues with current techniques, in order to motivate the subsequent work within this paper.

2.1 Goals and Definitions

[18] [7]

fault levels [23]

2.2 Model-Based FDIR

[25]

2.2.1 System Model-Based FDIR

[11]

2.2.2 Rule-Based FDIR

[18]

2.2.3 Residual Generation

mention Thresholds

2.3 Fault Filtering and State Observation

2.3.1 Full-State Observer Fault Filters

2.3.2 Parity-Space Filtering

2.3.3 Kalman Filtering for FDIR

[13] [24]

Also, particle filters:

[7]

2.4 Redundancy and Voting Systems

2.5 Advanced Techniques for Fault Detection

2.5.1 Fault Syndromes

[9]

2.5.2 Machine Learning and Classification

SVMs: [14] HMMs for state learning/classification: [3] Bayes: [10] [16]

-talk about their downsides, like needing to have a lot of training data, and hardness to debug

2.5.3 Distributed Fault Control

[5]

2.5.4 Common FDIR Issues

[12]

-write about problems with root cause analysis

-lead into the higher-level analysis of correlation between faults, and my theories about how this can be effective -Human-Centered Considerations for FDIR

Chapter 3

Correlative Analysis

Purpose:

-briefly introduce why I think correlative analysis is valuable -talk about some basic correlative theory and statistic analysis -lay out various correlation score techniques -talk about their limitations and some alternatives -choose what I think is most appropriate for this problem –lay out argument for PCC, perhaps

We borrowed insight and several visualization techniques from the literature. Cancro et al. developed useful techniques for packing large numbers of channels into a dense rectangular space [**?**], and Yairi et al. demonstrated ways to show change correlation between data channels [**?**], which were inspirations for our Global Correlation Matrix and Channel Correlation Vector.

3.1 Correlation for Fault Analysis

3.1.1 Correlation Matrices

3.1.2 Cross-Correlative Leak Analysis

Isermann paper

Discussion of how this is using correlation between telemetry values to look for an understood fault state, rather than for data discovery (but it's still valuable!)

3.2 Common Correlation "Score" Techniques

When looking at two sets of data, it can often be valuable to reduce their interdependence (i.e., how much they change in sync with each other) into a single number, or "correlation coefficient." This coefficient can be used as straightforward metric to determine correlation between sets of times series data. It may even be used as input into visualization algorithms to affect shading or even positioning, as we will see in later chapters.

3.2.1 Pearson Correlation Coefficient

The Pearson Correlation Coefficient (PCC), also known as the Pearson Product-Moment Coefficient, is a metric of the linear relationship between two sets of data. It is essentially a scaled covariance, defined as

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{3.1}$$

where X and Y are two vectors of data (or, in the case of the telemetry data sets we examine in this paper, times series of values over time for two telemetry channels). The PCC gives a quantified measurement of the linear correlation between the two vectors, in the form of a value in the range of $[-1, 1]$, where 1 is a total positive correlation, -1 is a total negative correlation, and 0 is no correlation at all.

The Pearson Correlation Coefficient carries with it a few important assumptions:

- Samples have values that are interval or ratio variables (not ordinal or categorical)
- Sample pairs follow a bivariate normal distribution
- Sample pairs have a linear relationship (or, at least, these are the type of relationships you wish to see)

If the data doesn't fit the assumptions above, one of the two rank correlation coefficients discussed below may be more appropriate.

3.2.2 Spearman Rank Correlation Coefficient (ρ)

The Spearman Rank Correlation Coefficient, or Spearman's rho, is a type of correlation coefficient, which, like the PCC, seeks to quantify relationships between vectors of data,

but which looks the ranking of variables within an ordering, rather than their linear relationship. This allows the coefficient to express relationship *monotonicity*, in order to be less dependent on linearity of relationships. It actually makes uses of the PCC to do this, by calculating the PCC on the ranked data values.

3.2.3 Kendall Rank Correlation Coefficient (τ)

The Kendall Rank Correlation Coefficient, like Spearman's rho, seeks to capture non-linear dependence by using the ordered ranks of the argument variables as input to the algorithm.

A visual comparison of the Pearson Correlation Coefficient, Spearman Rank Correlation Coefficient, and Kendall Rank Correlation Coefficient is shown in Fig. 3.1.

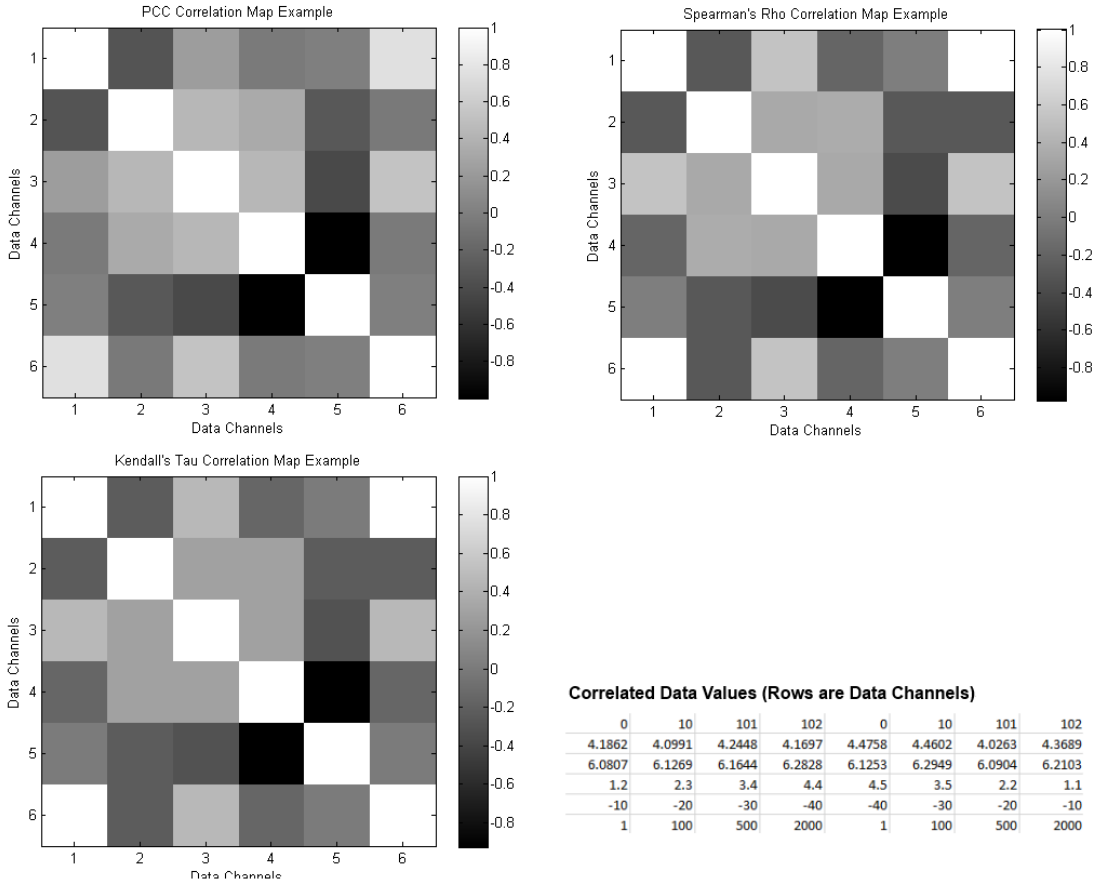

FIGURE 3.1: Three common correlation coefficient algorithms are compared on a sample data set. Note that 1 and 6 have a strong rank-based correlation, but the relationship is non-linear, resulting in a visibly lower correlation score within the PCC visualization. Also note the strong negative correlation between items 4 and 5.

3.3 Limitations of Traditional Correlative Techniques

There are many limitations to the three traditional correlative techniques above, and it's important to understand them, even if the ultimate decision will be to use one of these techniques. Some of the major limitations are decribed below.

3.3.1 Linearity Assumptions

The PCC algorithm, in particular

In contrast, the rank correlation methods

3.3.2 Multivariate Normal Distances vs. Elliptical Distances

3.3.3 Implied Causation

3.3.4 Alternatives to Correlation Score Techniques

3.3.5 Distance Correlation

3.3.6 Correlation Ratio

3.3.7 Brownian Covariance

3.3.8 Coefficient of Determination

3.3.9 Polychloric Correlation

Chapter 4

Visualization of Correlative Relationships

Purpose:

-talk about why we need to visualize correlative relationships in the first place -talk about challenges of visualization of correlation -talk about Z-scores for comparison and other traditional statistical methods -present corrgrams and their background and theory -talk about some popular applications of corrgrams and research -talk about extensions and the idea of animated corrgrams –don't get into the widgets/data reduction methods of corrgrams yet; I can do that later

4.1 Goals

4.2 Corrgrams

4.2.1 Background

4.2.2 Applications

4.2.3 Animated Corrgrams

Chapter 5

Case Study: Hakuto "Moonraker" Lunar Rover

In this section, we will discuss a motivating problem and platform on which to test some of the algorithms we've developed and assess their practicality.

5.1 HAKUTO Lunar XPRIZE Team

In the summer and autumn of 2015, research was performed at Tohoku University's Space Robotics Lab (hereafter "SRL"), under the guidance of Professor Kazuya Yoshida, among others. This laboratory focuses on the research and development of robotic systems for space exploration and science missions.

One of the major sub-groups within SRL is the Hakuto Lunar XPRIZE team, a group of engineers who, working together with their promotional counterparts in Tokyo, have been working for several years on the core mission of sending a lunar rover to the Moon, traveling at least 500 meters, and sending back high-resolution and photos. Completing this mission would satisfy the requirements of the Google Lunar XPRIZE, an international lunar rover competition with a combined purse of $30M USD [1].

As a secondary mission, Hakuto hopes to explore the interior of caves on the Moon, as precursor exploration to assess their feasibility as future human habitats. Recent high-resolution photography from JAXA's Kaguya spacecraft, and from NASA's Lunar Reconnaissance Orbiter, has confirmed that large "skylights" exist on the lunar surface leading into these caves [6], and Hakuto aims to land near enough to one of these skylights to make its exploration a possibility.

We decided that Hakuto's four-wheeled "Moonraker" rover would be an excellent testbed against which to develop advanced fault analysis algorithms and visualization. Moonraker is a state-of-the-art micro-rover, developed over the past 5 years by the Hakuto team. During normal operation, Moonraker sends back status reports on 100 to 150 channels of telemetry data to its ground station on Earth, at a rate of once per second. This telemetry covers everything from IMU attitude data and temperature sensors readings to motor rotations, solar charge voltage, and communication metadata such as packets errors detected and radio signal strength.

See Figure 5.1 for a photo of Moonraker.

FIGURE 5.1: Hakuto's Moonraker "Pre-Flight Model 2." Photo by George Thomas Mendel.

5.2 Ground Station Interface

In early July 2015, Team Hakuto began designing a new ground station software suite for the most recent version of the rover. The rover had been updated in several ways since the prior Pre-Flight Model, and many of its avionics, including its software, were updated and redesigned, breaking compatibility with the previous version of the ground

station software. We worked together to evaluate the current and future needs of the rover, and to redesign and reimplement software to optimally fit these needs.

Our discussions focused on optimizing performance, reliability, and maintainability of the software. The latter factor was of particular concern, given that the software was to be used and maintained in a university laboratory environment, with many students—some of them inexperienced in software engineering—potentially responsible for updating the software and adding new features. After evaluating a list of disparate choices, including C++ on Linux with the Qt framework and multi-platform JavaScript running in HTML5, we ultimately decided that the ideal choice would be the Unity Game Engine, for its high frequency of software updates, its active developer community, and its aerospace legacy within the NASA Jet Propulsion Laboratory [21].

5.3 Rover Data Path

In Hakuto's network configuration, Moonraker sends data packets over radio from the Moon, and they are intercepted on Earth and relayed to the ground station over the Internet. Subsequently, these packets are decoded and processed in order to be visualized by the ground station. The data is stored locally in memory by the ground station software for subsequent display. A flowchart of this data path is shown in Fig. 5.2.

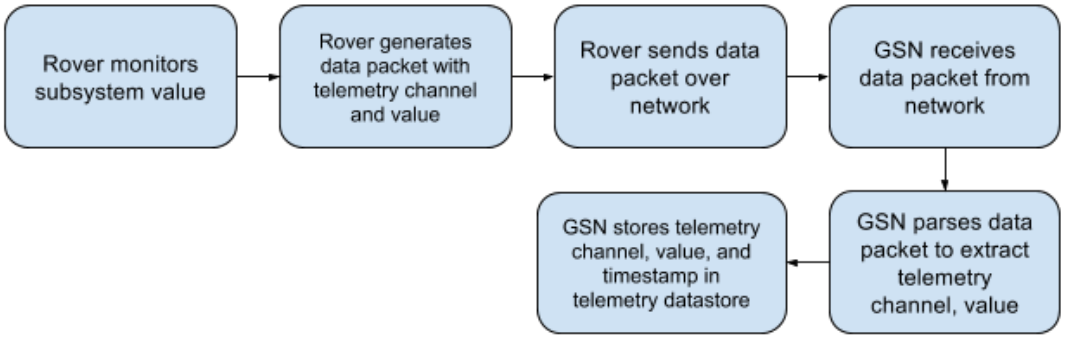

FIGURE 5.2: Moonraker's communication flow, from rover subsystem to ground station data storage, is shown.

5.3.1 Non-FDIR-Related Components

The focus of this paper is on the fault detective and correlative visualizations implemented within the ground station, so we will spend the bulk of the time focusing on this component. However, there were several non-FDIR-related components as well, briefly described below to provide context:

- *Numerical telemetry display*, to visually show the state of various sensors on subsystem boards (such as IMU accelerations, motor voltages, and board temperatures), as well as internal software metrics. Whenever possible, telemetry data was placed in semantically meaningful positions and groups, to improve discoverability.
- *Attitude display and visual tachometer*, to provide more intuitive visualizations of pitch, roll, and current wheel rotation rate (which mostly corresponded to vehicle speed).
- *Telemetry change indicators*, to point out data channels that have strong downward or upward trends over time.
- *Quad-camera display*, to display the most recent images and streaming video from the rover cameras.
- *Connectivity map*, to show the state of connectivity to various subsystems based on the elapsed time since packets from those subsystems had been received.
- *Immersive viewing*, allowing users to navigate the camera data in an embedded 3D mapping.
- *Map display*, showing the position of the rover with respect to the surrounding selenography, based on mobility subsystem telemetry and SLAM telemetry.
- *Audio alert cues*, to draw the user's attention to the UI in the event of faults.
- *Telemetry saving, loading, and playback*, to facilitate the review of mission events after the fact.

5.3.2 FDIR-Related Components

In comparison with traditional aerospace ground station software, particular attention was given in this implementation to FDIR-related components. The components below were implemented and used extensively.

5.3.2.1 Threshold-based fault detection

In order to build a threshold-based fault detection system for Moonraker, we worked with engineers on our team to define the "danger" thresholds that indicated points of severe jeopardy, as well as the "warning" thresholds that indicated points of concern. We implemented these thresholds in my ground station software as a general-purpose fault detection engine. Faults are detected constantly, and are displayed to the user via

color and detailed information (see the following sections for more details). All fault occurrence details and times are logged for future review as well. See Figure 5.3.

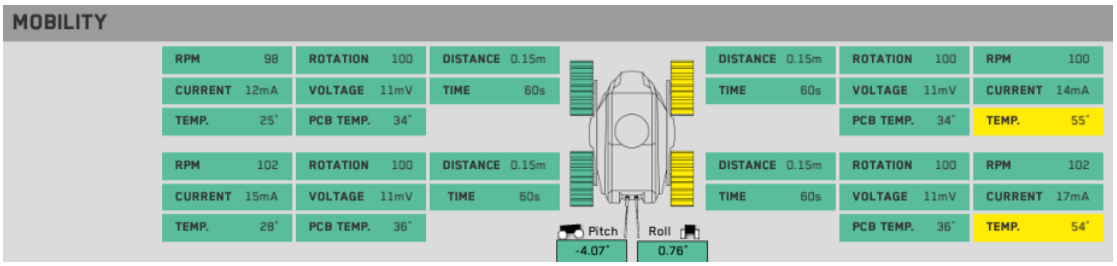

FIGURE 5.3: Various telemetry values are shown for the mobility subsystem. Colors indicate current fault detection levels, with green representing a nominal state.

5.3.2.2 Fault alert panel

For safety, faults that have occurred need to be easily visible and understood by human operators. Towards this end, a highly visible, brightly colored alert panel was placed at the top of the screen seen by human operators. Each panel cell corresponds to a subsystem or other type of data grouping, and any issues with that grouping (i.e., faults that occur on monitored channels) will trigger a color change on that cell. Interacting with the cell can give the user more information on the fault (see the next section for details). See Figure 5.4.

FIGURE 5.4: An alert panel monitors potential faults on various different subsystems, indicating any faults and their severity by color.

5.3.2.3 Expert fault information

Ensuring human understanding of fault data is an essential part of the fault diagnosis and recovery process. As such, it's important to design a system where detailed information can be provided about individual faults that have occurred, while maintaining a high-level understanding of which systems are behaving anomalously. The system designed allows for this hierarchical organization of information. When high-level fault information is indicated in the "fault alert panel," more concise information is provided in the "fault information panel," including which anomalous data channels are contributing to the problem. The fault information is highly extensible, allowing for any other additional fault-related notes that system designers or operators would like to include for

reference. This additional information, uncommon in traditional fault monitoring systems, accelerates the fault diagnosis problem by immediately pointing towards possible root causes. See Figure 5.5.

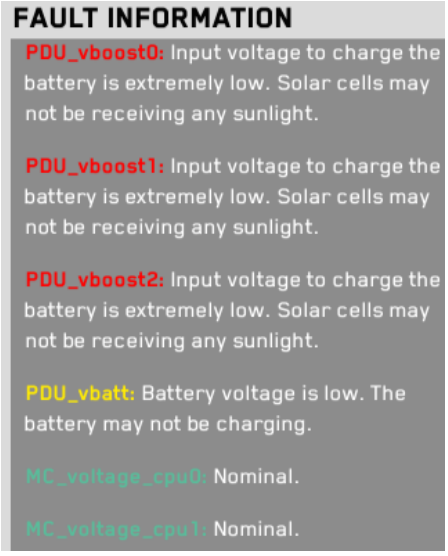

FIGURE 5.5: Information for monitored faults on a given subsystem.

5.3.2.4 Correlative Functionality

The Pearson Correlation Coefficient was chosen as an algorithm for calculating correlationbetween sets of data channels, in order to give a metric of their mutual "connectedness." This algorithm was chosen for its simple and efficient calculations, under the (perhaps premature) assumption of linear relationships.

To visualize this data, a two-dimensional corrgram was used, visualizing relationships between data channels as a matrix with cell shadings representative of the PCC score of each relationship. Hue of each cell shows positive/negative correlation, and intensity shows the strength of that correlation. This visualization allows an operator can see changing channel correlations, which may suggest possible interconnectedness or causation and aid with troubleshooting. See Figure 5.6.

To show the flow of this data, pseudocode for the algorithm used is shown in Algorithm 1. Assume that a pre-generated $n \times n$ grid of transparent blocks, **Corr**, is displayed on the screen. Also assume that **GetColor** refers to an arbitrary function which maps from a PCC score $\in [-1, 1]$ to an RGB color $(r, g, b) \in \mathbb{R}^3$, where $0 \leq r, g, b \leq 255$. We experimented with several variations of this function, and found that a linear mapping exaggerated the importance of low correlation scores, which led to the development of a hand-tuned exponential color mapping function to produce the correct scores. Certain work has explored the complexities of determining a proper mapping, which relate to the

non-linearity of human perception of color [8]. Other work has suggested the efficacy of pre-squaring the PCC score before display, which instead results in displaying the coefficient of determination (i.e., the shared portion of the variance) for the pair of data items, which intuitively corresponds to a more intuitive human understanding of data connectivity [17].

Algorithm 1 Animated Corrgram Generation Algorithm

procedure CORRGRAMGENERATOR(D) ▷ Takes data point matrix $D \in \mathbb{R}^{n \times t}$.
  $S \leftarrow \{U_{:,j}\}_{j=t-s}^{n}$ ▷ Reduce data to most recent s points.
  $P \leftarrow \mathbf{PCC}_s(D_r)$ ▷ Calculate a symmetric PCC matrix using last s points.
  for row $= 1$ **to** n **do**
    for col $= 1$ **to** n **do**
      color $=$ **GetColor**$(S_{row,col})$ ▷ Convert PCC score to a color.
      $\mathbf{Corr}_{row,col}$**.color** $\leftarrow$ **color** ▷ Assign color to cell in corrgram.
    end for
  end for
end procedure ▷ Algorithm is re-run on every graphical update.

A few additional modifications were applied in order to make this algorithm performant. For instance, the entire corrgram is not updated at once, due to the performance hit from calculating new PCC scores for thousands of corrgram cells on each graphical update. Instead, a subset of the corrgram cells are updated, resulting in a "rolling" effect wherein cells update gradually, with a full update of the corrgram being achieved on the order of once per second.

We also built functionality to adjust the sampling rate, the period of time over which samples are taken, and parameters related to averaging for smoothing out noise. Channel pairs may also be filtered via substring search.

5.4 Testing

Hakuto carried out a number of field tests on the ground station software, including FDIR-related and correlative components. The major tests we conducted are described in detail below.

5.4.1 Field Testing

Over the course of two weeks, we performed daily field tests in the rock quarry, to both assess the physical characteristics of our radio communication and to confirm our mobility in a simulated lunar environment. (See Fig. 5.8 for an image of the quarry terrain.) The ground station interface was in constant use throughout the course of the

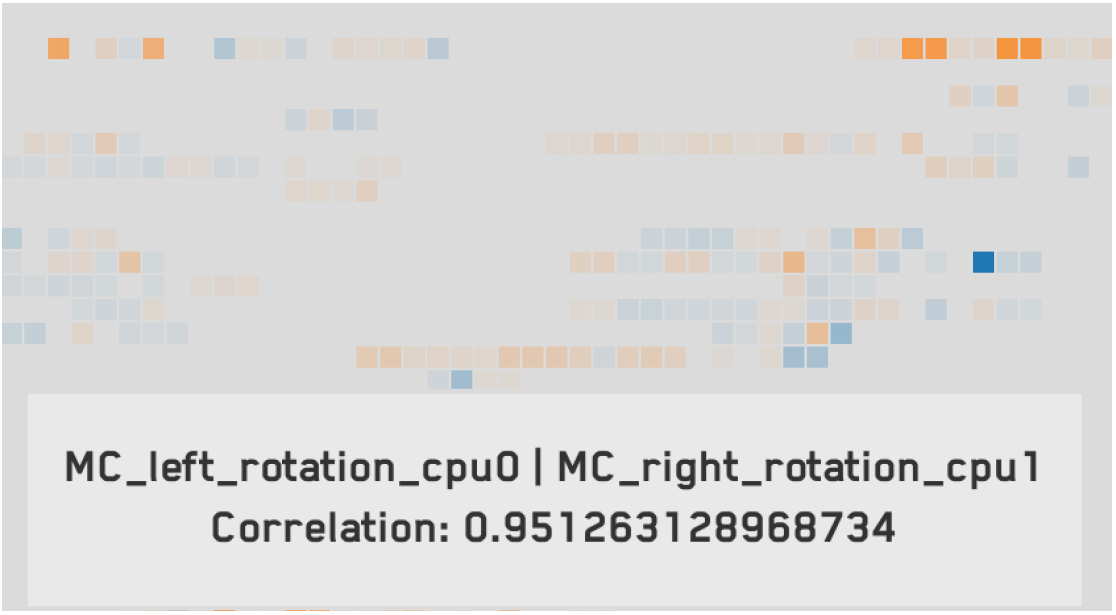

FIGURE 5.6: Correlation map visualization showing various different data channel pairs and their correlations. Bright orange signifies a high positive correlation, and bright blue is a high negative correlation.

tests, with alternating drivers at the helm. The setup usually involved one primary pilot and one copilot, who helped with troubleshooting. See Fig. 5.7 for an image of normal operation.

Our radio testing was multi-pronged. First, we endeavored to established that we could a) communicate from the ground station to Moonraker and vice-versa by going through Astrobotic's radio relay, which they set up to emulate their lunar lander which will function as a communication relay during the actual, planned lunar mission. Second, we tested the communication capabilities of Moonraker's radio antenna at two different operating frequencies (900 MHz and 2.4 GHz), in order to characterize performance over long distance and with line-of-sight blocked by rocky terrain. We looked for communication signal strength and commanding efficacy in the presence of packet loss, and also tested the effect of a signal strength amplifier in a poor connectivity situation. These results were promising and are currently being analyzed by our communications team.

Our mobility testing consisted of driving long distances over rocky, yet generally even, terrain, occasionally using only the near-real-time ("NRT") streaming video telemetry as visual feedback. We set up various challenges, such as large rock obstacles and inclines of increasing steepness, to test the rover's mobility capabilities as well as our operational capabilities using the ground station interface.

Other tasks performed during the field tests included coordinating with Hakuto crew and Astrobotic/Caltech engineers to fulfill mission tasks, scouting for field test locations,

setting up and testing the ground station and radio equipment, and driving the rover during all of the tests to accomplish all of our test goals.

Results of this field test, and other similar ones, will be discussed in the next chapter.

FIGURE 5.7: Lead software engineer Toshiro Shimizu and the author work on radio testing during a Pittsburgh field test at the Lafarge rock quarry.

5.4.2 Usability Testing

In November 2015, we performed a set of usability tests on a slimmed-down version of the Moonraker ground station interface, in order to evaluate the various data analysis affordances and to determine any usability issues in need of attention. Seven users participated in the test, mostly interns and students possessing some familiarity with the rover, but with limited to no experience operating it via the ground station interface. See Fig. 5.9 for an image of a user taking the test.

Participants were presented with a simplified form of the ground station, with the Pilot Screen, Copilot Screen, and the Correlation Map screen. As the correlation map is a relatively novel idea, and has not (to the best of the author's knowledge) ever been used in ground station software, this test also served to see if it could be immediately usable

FIGURE 5.8: Engineering members of Team Hakuto discuss terrain challenges during a nighttime field test.

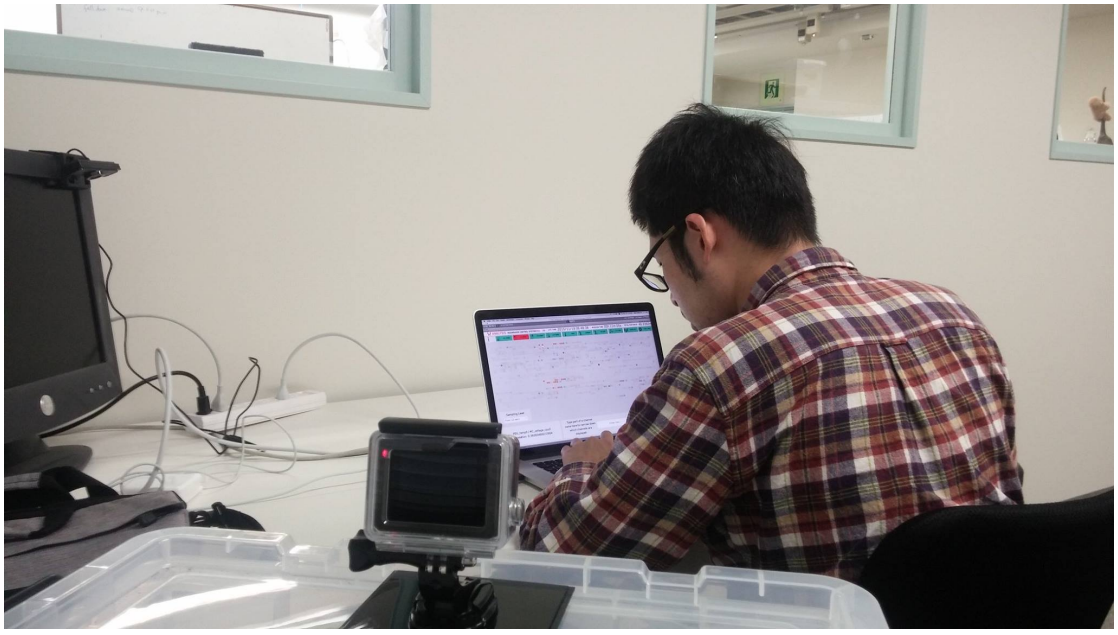

FIGURE 5.9: A usability test participant focuses on the incoming telemetry to try to ascertain patterns behind a faulted system.

in its present state, or if special training or modifications will be necessary before the correlation map is useful for correlation discovery and root cause analysis.

5.4.2.1 Test Tasks

During the test, participants monitored telemetry and looked for patterns in received data during a 10-minute simulation drive on the lunar surface. During the course of the drive, camera images were unavailable, but the rover telemetry was indicative of various events that occurred to the rover during the run. The scenario consisted of the rover descending down into a crater, where it loses direct sunlight, resulting in reduced temperatures and solar charge voltages. It also loses line of sight with the lunar lander, causing packet losses and a reduction in signal strength. The rover then climbs out of the crater, bringing about an all-around improvement in these conditions. It enters some rough terrain, and ultimately stops after a rock becomes stuck in the wheel.

It was the task of the usability test participants to analyze the telemetry as it was received in order to try to understand the details of the story above. After the 10-minute run was complete, users were then given up to an additional 10 minutes to review received telemetry until they were satisfied they had understood the "story" of the rover's trip.

Results of usability testing will be discussed in the next chapter.

Chapter 6

Intermediate Results and Reassessment

This chapter discusses some of the learnings from our intermediate testing, and how we used them to shape our next steps for correlative assessment techniques.

6.1 Field Test Results

Our field tests went swimmingly, and we returned to Japan with excellent results and ideas for improvement, but these tests were not without a small number of issues. We uncovered several minor bugs within the ground station, which were quickly fixed on-the-spot. The tests—especially the act of watching others operate the ground station—yielded useful data about which features were used the most, and which were all but ignored entirely.

We found the "immersive viewing" feature to be immensely helpful for gaining situational awareness, and proved its efficacy in testing where engineers placed the rover in a precarious situation, then asked us over radio to use the telemetry and visual data to give them a detailed description of the current situation. The "connectivity map" also proved extremely useful in quickly informing us of issues with the rover, as did the overall fault detection system. Some features were determined to be of limited utility and removed.

Fault information was shown to be somewhat sparse for encoding expert understanding of problems, and there were several instances of users accidentally ignoring unsafe rover conditions. In response to these observations, space made available from removing other features was used for providing more detailed fault information (this was an enhancement

of the "Expert fault information" feature discussed in the previous chapter). Also, fault detection and display rules were rethought and tweaked, to enhance visibility of anomalous states.

6.2 Usability Test Results

Usability testing yielded many results that were useful for setting subsequent development priorities. Overall, all participants were able to deduce the general course of events undergone by the rover, and at the end were able to narrate a story which resembled the intended one outlined above.

Users remarked that they found the Copilot Screen's telemetry display the most useful, and they spent the majority of their time monitoring telemetry data on this screen. Most participants used the data review feature heavily while on this screen, rewinding and fast-forwarding through time and watching displayed telemetry values change as they did so. Users remarked that this feature was easy to use and excellent for reviewing the flow of telemetry. However, multiple users expressed a desire for time series plots of data channels, to better see the history of data at a glance.

Faults were generally very visible, and users commented that they found the additional fault information very helpful when trying to understand the meanings of various fault states. However, it was observed that tunnel vision was occasionally a problem for users; too much attention on one specific UI component, such as the correlation map or displayed telemetry on the Copilot Screen, seemed to be responsible for delays of up to 30 seconds in reacting to critical fault events. This observation led to the addition of audio cues to redirect user attention, which has already shown to be effective in informal testing.

Multiple users commented on the difficulty of understanding telemetry for channels whose meaning they did not understand. (Many of the subsystems have very specific data channels whose meaning is not well understood to anyone except the designers of those systems.) Results indicate that the expert fault information feature mentioned above was effective in eliminating much of the confusion about specific faults; however, we believe that adding information that leads to a better understanding of individual telemetry channels would result in less user confusion and could improve the effectiveness of human telemetry monitoring. Knowledge capture efforts are currently underway to collect detailed information about data channels from subsystem engineers to incorporate this data into the interface.

The usability test uncovered many issues with the correlation map in its current form. Many of the users commented that they did not understand the proper way to use it to analyze data (although they understood the basic idea of the visualization). Users requested better, more intuitive spatial organization of data channel pairs, and the ability to more easily filter channels of interest and ones pertaining to faults. We received comments that additional training sessions might be beneficial. Nearly all users expressed an interest in using this feature, but the performance of those who endeavored to analyze faults with this tool showed evidence of a need for automated simplification to reduce stimuli, and to come up with better ways to train users and to lead them to useful conclusions.

A few other minor issues came up which we had not foreseen until performing the testing. Some users had difficulties with transition lag between screens (there is a lag of approximately a second between screen transitions due to a need to load resources). We am looking into performance enhancements and/or loading screens to fix this issue. We also received the feedback that a more visible speed/RPM indicator for the rover would be helpful, as speed is one of the most important aspects of the rover as it operates, and this data is easily overlooked in the midst of other types of telemetry. This led to the development of the visual tachometer RPM visualization discussed in the previous chapter.

6.3 Improvements and Additions

Looking at the rests of this intermediate testing, we were able to identify various targets for further research and improvements in the field of analysis of the fault-related, and correlative data.

Even with the small number of data channels available on the Moonraker rover (112), the dimensionality was very high for a full corrgram display, and seemed to stretch the visual and attentive capacities of the users who participated in the test. Even with the addition of data filtering features, without focused training of the use of these features, they didn't seem to provide a better experience for users looking for patterns in the correlative data. Though users were able to, from an integrative viewing of telemetry data as shown in the numerical and color-based fault displays, able to come to understand a timeline of the rover events, mental links between associated channels seemed to emerge due to sheer coincidence; remarks from users were along the lines of "the temperature is increasing... and I see that the voltage is increasing too... maybe they're related." The correlation map, as a way to call out these patterns, did not seem to be as effective as anticipated.

While this data was successfully captured by the analysis, we determined that it was a) buried in the noise of lots of unimportant correlations and b) shown on a larger visual display with too much data to easily visual process in a short amount of time. What's more, there was no affordance for users to display time across different temporal points at the same time.

As such, we determined that it would be of value to iterate on these two points, with the main goals of reducing correlative noise and data dimensionality and of providing a simplified visualization capable of displaying data from multiple time points simultaneously. The next chapter will discuss a few approaches towards this end and their results.

Chapter 7

Dimensional Reduction and Visualization Improvements

Purpose:

7.1 Dimensional Reduction

7.1.1 Principal Component Analysis

Tharrault paper

Russell paper

7.2 Meta-Analysis

7.2.1 Out-of-Family Telemetry

Two NASA papers

7.2.2 Out-of-Family Correlations

7.3 Corrgram Enhancements and Dimensional Reduction

7.3.1 Smoothing and Time Adjustments

7.3.2 Ranked Filtering

7.3.3 Fault Filtering

7.3.4 Substring Filtering

7.3.5 Cross-System Filtering

7.3.6 Timelines

7.4 Two-Dimensional Graph Embeddings

Since the vast majority of user interfaces in common usage are two-dimensional, and hardware limitations can easily result in 3D user interfaces being infeasible for users, it makes sense to look at ways that n-dimensional system state data can be embedded within a 2D visualization. Towards this purpose, I did a brief survey of state-of-the-art 2D graph embeddings, looking for implementation feasibility and the ability to give a user "insight" into the nature of a system fault. Preferably, a 2D embedding for system understanding will make major state transitions and patterns visibly obvious at a glance, and will spatially separate different system "modes" so that they can easily be mentally grouped by the viewer.

7.4.1 Undirected Dependency Graphs

The first type of two-dimensional graph embedding that we examined as an alternative for animated corrgrams was the "dependency graph." Dependency graphs are a type of 2D embedding in which a complex system is represented as a directed graph, where nodes are system components and edges represent dependencies; for example, if $\mathbf{node}_A \rightarrow \mathbf{node}_B$ and $\mathbf{node}_B \rightarrow \mathbf{node}_C$, this indicates that the value of $\mathbf{node}_A$ depends on the value of $\mathbf{node}_B$, which in turn depends on $\mathbf{node}_C$. A simple example of this type of visualization is shown in Fig. 7.1.

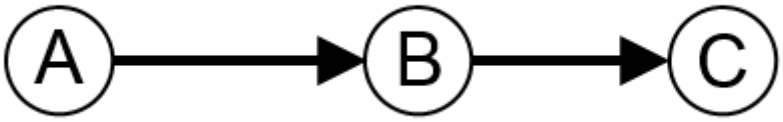

FIGURE 7.1: A simple example of a traditional dependency graph is shown. Here, node A's value depends on the value of node B, and node B's value depends on the value of node C.

As such, this type of visualization lends itself well to illustrating the effect of causation in a system. Though causation can be difficult to determine in a complex system, as we have already shown, correlation is calculable via the PCC and other metrics, and a correlation-generalized undirected correlation graph could be envisioned, wherein two nodes have an undirected edge if their mutual PCC score exceeds a certain threshold, and have no edge if their mutual PCC score fails to meet that threshold. The steps to generate such a graph are as follows:

1. At a given point in time, generate a PCC matrix for all of the possible data channel pairs.
2. Generate an unconnected graph in which there exists a degree-0 node for each data channel.
3. For node-node pair, add a connecting edge if there exists a PCC score above a reasonably high correlation threshold (e.g., $r^2_{PCC} > 0.8$). This edge can be colored to show correlation sign (e.g., blue for $r_{PCC} < 0$ and red for $r_{PCC} > 0$).
4. Finally, cull all nodes that are still of degree 0.

With the corrgram visualization, we needed to illustrate all possible pairs of channels as a separate cell, and thus ended up needing to visualize $\frac{n!}{2(n-2)!}$ different cells (where n is the number of data channels). However, with a dependency graph visualization, we can reduce the number of colored elements (nodes) to a count of n, by introducing connecting edges. (For systems that are not highly correlated, this will produce far less visual clutter than the corrgram visualization.) Furthermore, we can simplify the undirected dependency graph visualization by eliminating any components of degree 0, if the correlated components are the only ones we wish to see.

We experimented with actually creating this visualization for explorational purposes. First, we ran a system dynamics simulation for our lunar rover described in Chapter

5. We paused the telemetry analysis at a certain time step at which interesting correlated components were present, and examined the data channel correlations at that instant. The correlated components are illustrated in the correlation map visualization in Fig. 7.2. We then isolated the correlated components and illustrated them as an undirected dependency graph, using the steps outlined above. The resulting undirected dependency graph visualization, with positive and negative correlation edges both visible, is shown in Fig. 7.3. In addition, positive and negative correlated components have been isolated into separate graphs for readability and discoverability, as shown in Fig. 7.4 and Fig. 7.5, respectively.

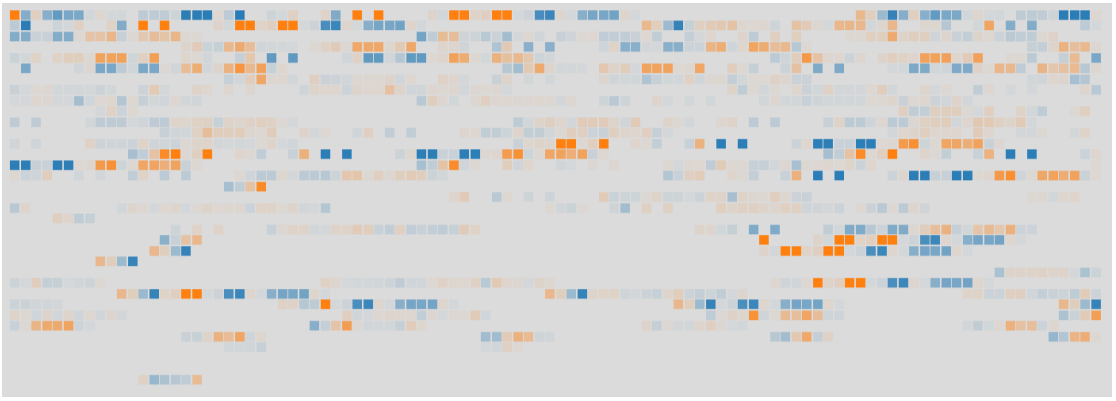

FIGURE 7.2: A snapshot of the correlation map display from a simulated run. Note the strong correlations illustrated by opaque orange (positive) and blue (negative) cells.

FIGURE 7.3: A snapshot of an undirected dependency graph display from a simulated run. Correlated components have been isolated, with edges drawn for all correlation relationships exceeding a certain value ($r^2_{PCC} > 0.8$). Both positive and negative correlation connections are shown. Self-correlations are not shown.

These visualizations present a very different way of viewing the correlative data. While the temporal dimension is still only captured as a snapshot (i.e., the correlative state

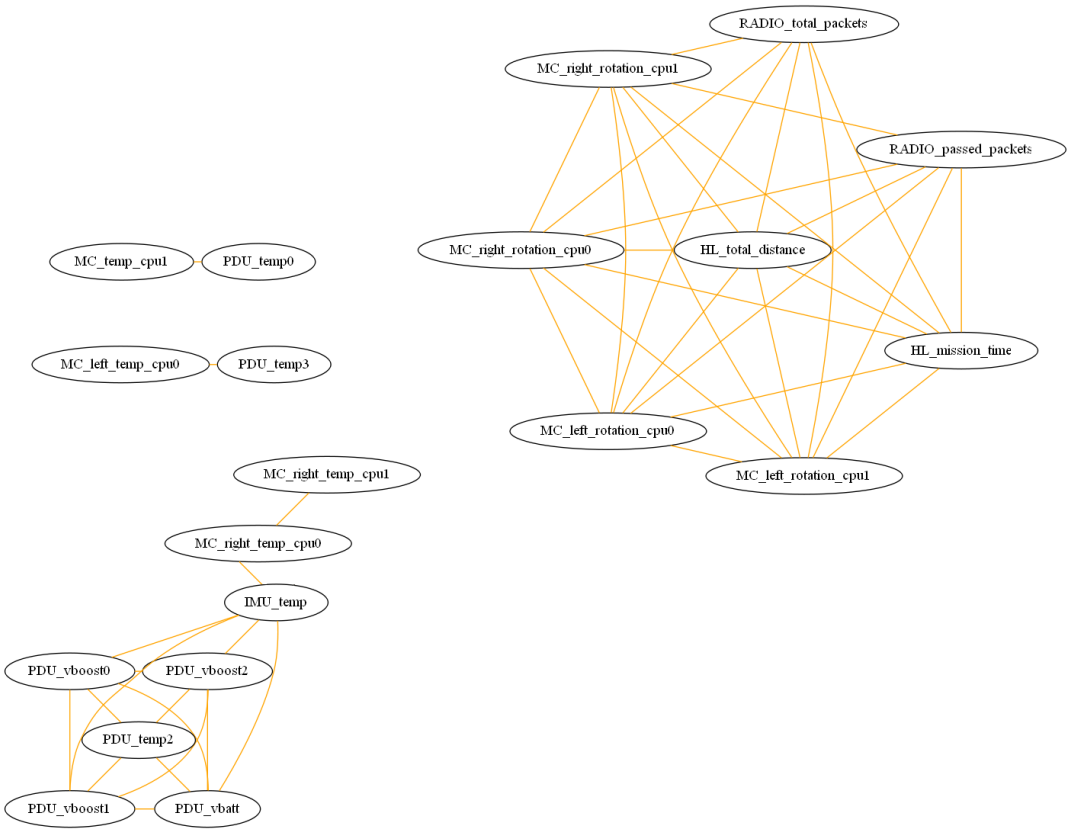

FIGURE 7.4: A snapshot of an undirected dependency graph display from a simulated run. Correlated components have been isolated, with edges drawn for all correlation relationships exceeding a certain value ($r^2_{PCC} > 0.8$). Only positive correlations are shown. Self-correlations are not shown.

at only one time point can be shown at a time), correlated components are shown very clearly, and can be understood at a glance. In particular, the intuition behind the positive correlated components in Fig. 7.4 seems clear; the most fully-connected, major clusters are exhibiting behavior which is very similar to each other. (In fact, the lower left cluster channels were all in a state of monotonic decrease, and the upper right cluster channels were in a state of monotonic increase.) The negative correlated components are less obvious, as they don't "cluster" in the same way; however, the negative correlation data can be overlaid onto the positive correlation graphs as a higher-level operation on the clusters. This, perhaps, produces the most informative type of graph; this application is shown in Fig. 7.6.

Another advantage of this technique is that it clearly isolates small groups of correlated components; low-degree subgraphs can point towards noisy data, or towards significant links, but if they are persistent, it seems they may suggest interesting correlations that deviate from the patterns exhibiting by the bulk of the data channels, which tend to correlate due to behavior exhibiting positive and negative monotonicity. However, towards the idea of exploring a visualization which can show the evolution of state and

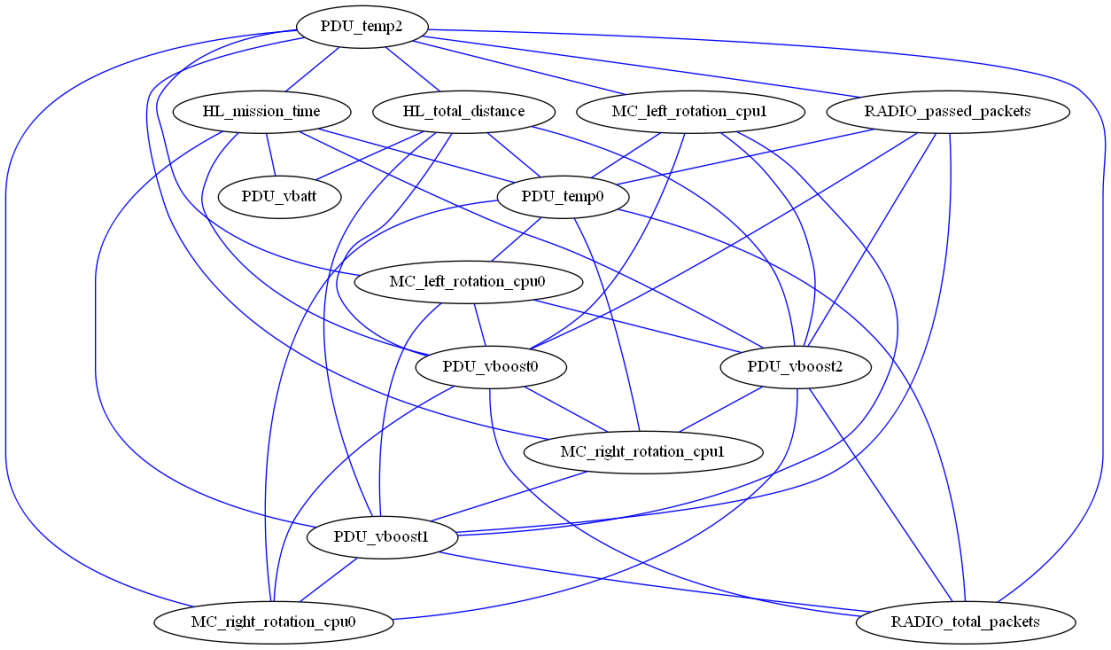

FIGURE 7.5: A snapshot of an undirected dependency graph display from a simulated run. Correlated components have been isolated, with edges drawn for all correlation relationships exceeding a certain value ($r^2_{PCC} > 0.8$). Only negative correlations are shown. Self-correlations are not shown.

correlative data over time, we will look at another technique in the following section.

7.5 Time Curves

In early 2016, Bach et al presented a powerful new type of visualization tool called "Time Curves" [4]. The time curve is a generic 2D embedding algorithm designed specifically for system state data which changes over time. Time curves visualize system states as a series of points, connected in temporal along curves within the 2D embedding. This allows the viewer to gain an understanding of system behavior by the shape and directions of the curves, and by the grouping of the data points. A visual example of how a time curve is "folded" from an initial linear timeline is shown in Fig. 7.7.

7.5.1

7.5.2

7.5.3

7.5.4

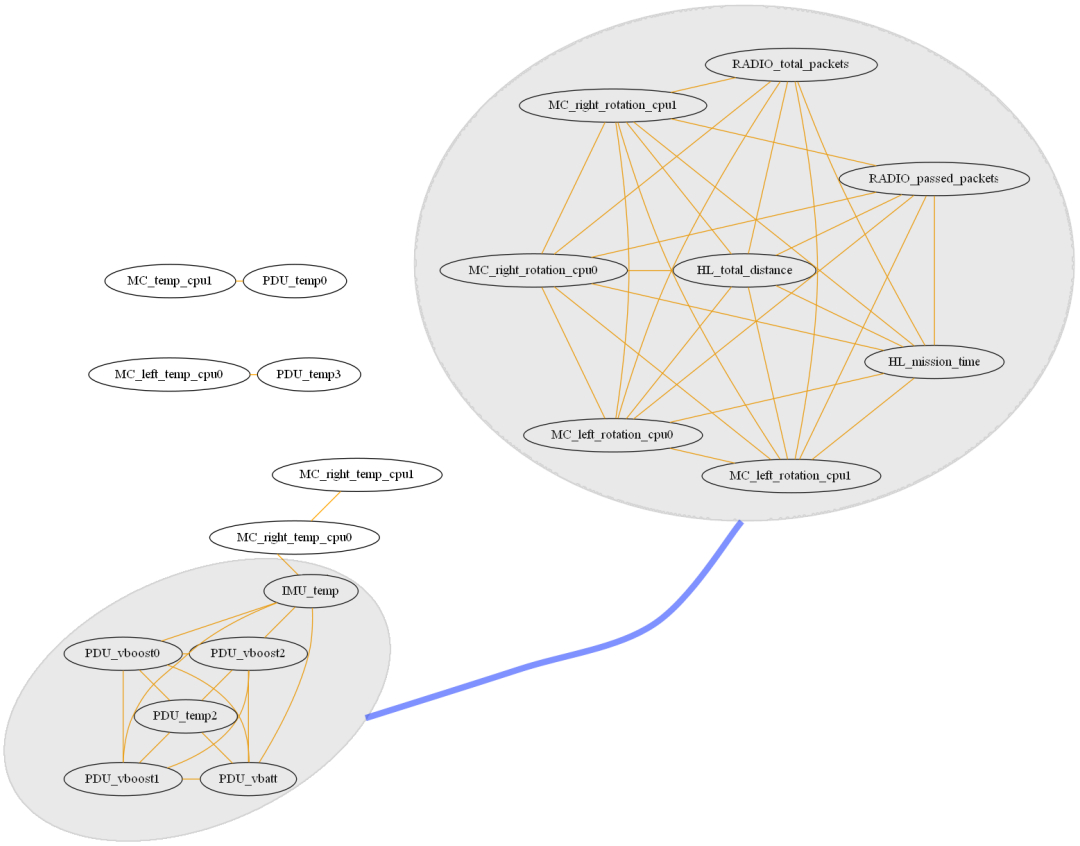

FIGURE 7.6: A snapshot of an undirected dependency graph display from a simulated run. Correlated components have been isolated, with edges drawn for all correlation relationships exceeding a certain value ($r^2_{PCC} > 0.8$). Positive correlations, and negatively correlated subgraphs, are shown. Self-correlations are not shown.

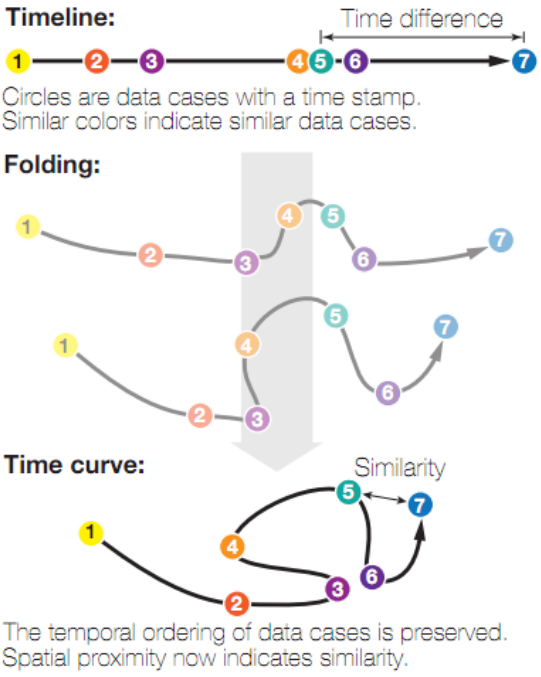

FIGURE 7.7: A time curve is "folded" from an initial linear timeline to bring similar data points close together in the 2D embedding. From [4].

Chapter 8

Results and Discussion

Purpose:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus at pulvinar nisi. Phasellus hendrerit, diam placerat interdum iaculis, mauris justo cursus risus, in viverra purus eros at ligula. Ut metus justo, consequat a tristique posuere, laoreet nec nibh. Etiam et scelerisque mauris. Phasellus vel massa magna. Ut non neque id tortor pharetra bibendum vitae sit amet nisi. Duis nec quam quam, sed euismod justo. Pellentesque eu tellus vitae ante tempus malesuada. Nunc accumsan, quam in congue consequat, lectus lectus dapibus erat, id aliquet urna neque at massa. Nulla facilisi. Morbi ullamcorper eleifend posuere. Donec libero leo, faucibus nec bibendum at, mattis et urna. Proin consectetur, nunc ut imperdiet lobortis, magna neque tincidunt lectus, id iaculis nisi justo id nibh. Pellentesque vel sem in erat vulputate faucibus molestie ut lorem.

8.1 A Section

Quisque tristique urna in lorem laoreet at laoreet quam congue. Donec dolor turpis, blandit non imperdiet aliquet, blandit et felis. In lorem nisi, pretium sit amet vestibulum sed, tempus et sem. Proin non ante turpis. Nulla imperdiet fringilla convallis. Vivamus vel bibendum nisl. Pellentesque justo lectus, molestie vel luctus sed, lobortis in libero. Nulla facilisi. Aliquam erat volutpat. Suspendisse vitae nunc nunc. Sed aliquet est suscipit sapien rhoncus non adipiscing nibh consequat. Aliquam metus urna, faucibus eu vulputate non, luctus eu justo.

Event Number	Timestamp	Event Description
0	0s	Rover begins traveling forward along smooth terrain.
1	188s	Rover enters crater; begins descending into crater.
2	287s	Rover enters shade, causing temp, comms, and power drops.
3	300s	Rover begins traversing smooth bottom of crater.
4	330s	Rover begins climbing out of crater.
5	343s	Rover exits shade; continues uphill.
6	534s	Rover emerges from crater and enters smooth terrain.
7	594s	Rover enters choppy terrain.
8	643s	Rover wheel has fault; rover stops moving.

TABLE 8.1: Events during a visualized user simulation are shown. Cross-reference "Event Numbers" with labels on on Fig. 8.1, 8.2, 8.3 and 8.4 to see correspondence.

8.1.1 System State Visualization with Time Curves

Set about to do time curve viz

describe data pipeline

describe four choices for comparison

maximum folding

no averaging

results shown

discuss results

8.2 Another Section

Phasellus nisi quam, volutpat non ullamcorper eget, congue fringilla leo. Cras et erat et nibh placerat commodo id ornare est. Nulla facilisi. Aenean pulvinar scelerisque eros eget interdum. Nunc pulvinar magna ut felis varius in hendrerit dolor accumsan. Nunc pellentesque magna quis magna bibendum non laoreet erat tincidunt. Nulla facilisi.

Duis eget massa sem, gravida interdum ipsum. Nulla nunc nisl, hendrerit sit amet commodo vel, varius id tellus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac dolor est. Suspendisse ultrices tincidunt metus eget accumsan. Nullam facilisis, justo vitae convallis sollicitudin, eros augue malesuada metus, nec sagittis diam nibh ut sapien. Duis blandit lectus vitae lorem aliquam nec euismod nisi volutpat. Vestibulum ornare dictum tortor, at faucibus justo tempor non. Nulla facilisi. Cras non massa nunc, eget euismod purus. Nunc metus ipsum, euismod a consectetur vel, hendrerit nec nunc.

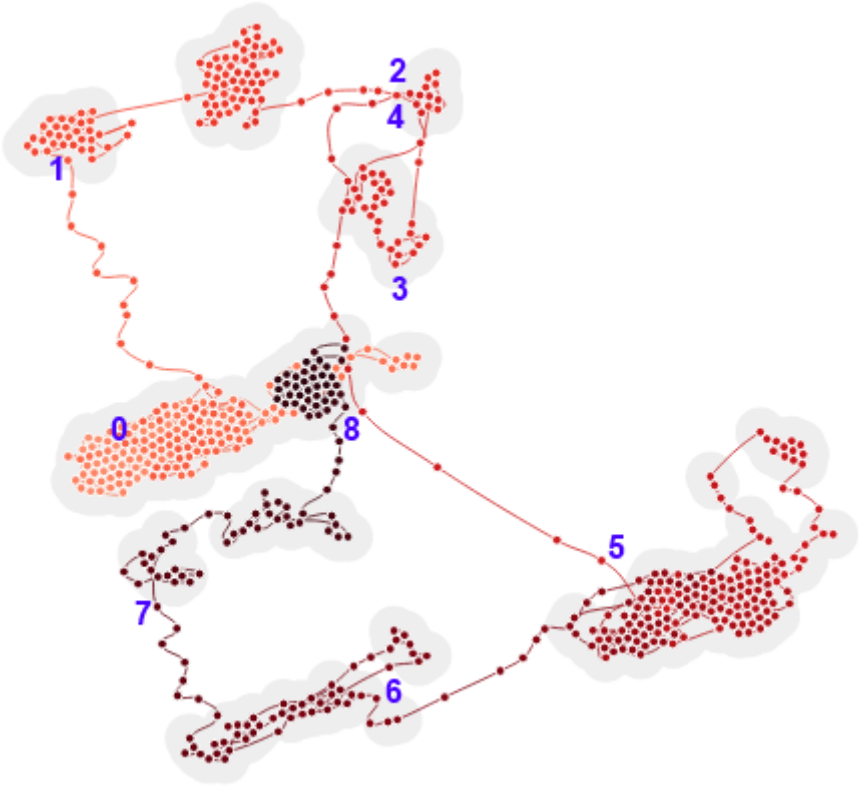

FIGURE 8.1: A time curve embedding visualizing mission events using PCC correlation state. Event annotations are described in Tbl. 8.1.

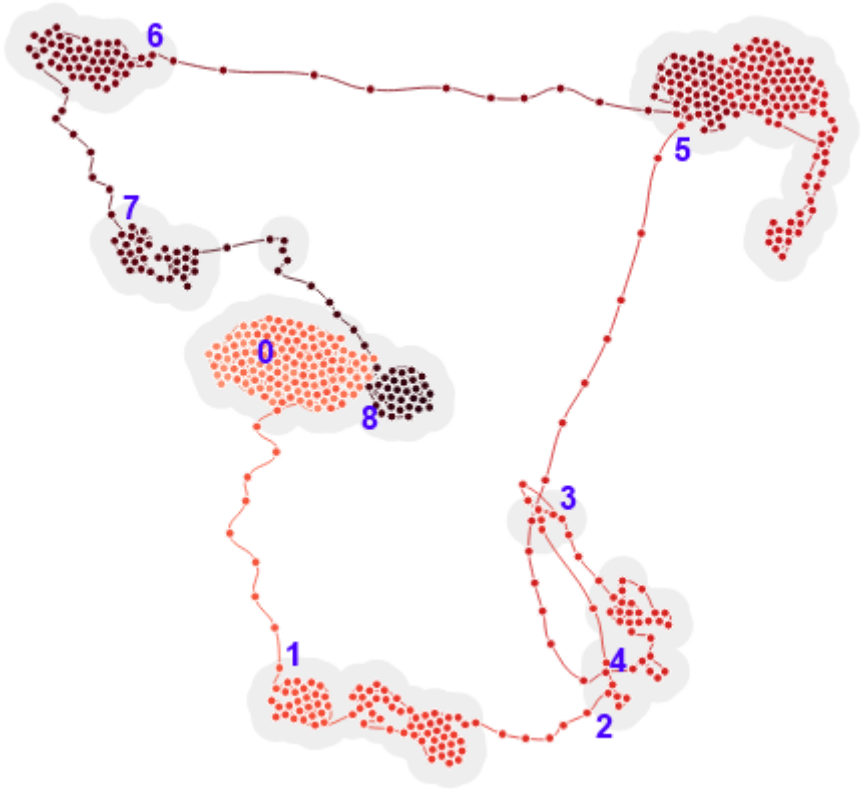

FIGURE 8.2: A time curve embedding visualizing mission events using Kendall's Tau correlation state. Event annotations are described in Tbl. 8.1.

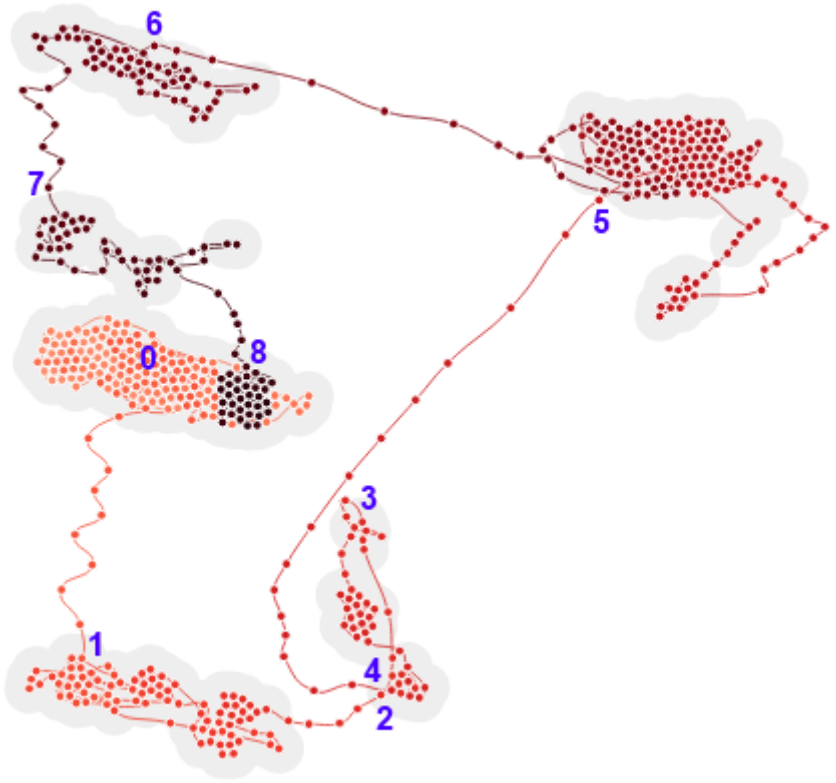

FIGURE 8.3: A time curve embedding visualizing mission events using Spearman's Rho correlation state. Event annotations are described in Tbl. 8.1.

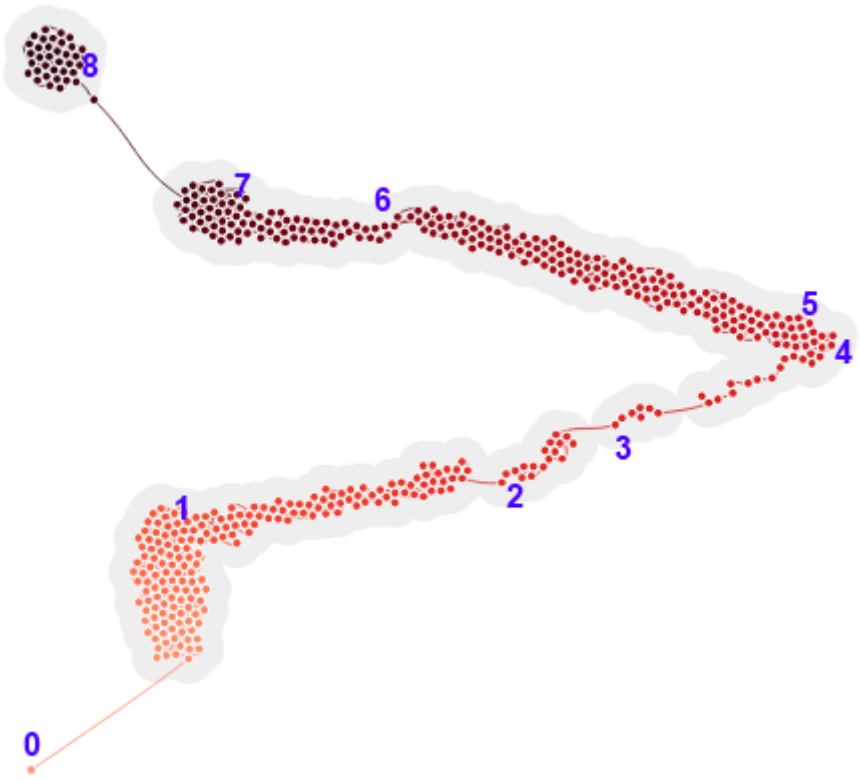

FIGURE 8.4: A time curve embedding visualizing mission events using raw telemetry state. Event annotations are described in Tbl. 8.1.

Chapter 9

Conclusion

Purpose:

recap of intro:

Purpose:

-to lay out the basic idea of the paper -explain motivating problem of paper -briefly mention what aerospace faults are -say why they're hard to troubleshoot -explain a little bit about root cause analysis -talk about what I'm trying to do in the paper -which is to look at data connections, correlation, etc. on a real aerospace system to try to figure out how data discovery can best be achieved and how patterns can be found with math methods -these aren't easy to show, so also look into viz -implement for actual system -evaluate -reassess -make modifications to both math and viz -test again -find conclusions

Appendix A

An Appendix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus at pulvinar nisi. Phasellus hendrerit, diam placerat interdum iaculis, mauris justo cursus risus, in viverra purus eros at ligula. Ut metus justo, consequat a tristique posuere, laoreet nec nibh. Etiam et scelerisque mauris. Phasellus vel massa magna. Ut non neque id tortor pharetra bibendum vitae sit amet nisi. Duis nec quam quam, sed euismod justo. Pellentesque eu tellus vitae ante tempus malesuada. Nunc accumsan, quam in congue consequat, lectus lectus dapibus erat, id aliquet urna neque at massa. Nulla facilisi. Morbi ullamcorper eleifend posuere. Donec libero leo, faucibus nec bibendum at, mattis et urna. Proin consectetur, nunc ut imperdiet lobortis, magna neque tincidunt lectus, id iaculis nisi justo id nibh. Pellentesque vel sem in erat vulputate faucibus molestie ut lorem.

Quisque tristique urna in lorem laoreet at laoreet quam congue. Donec dolor turpis, blandit non imperdiet aliquet, blandit et felis. In lorem nisi, pretium sit amet vestibulum sed, tempus et sem. Proin non ante turpis. Nulla imperdiet fringilla convallis. Vivamus vel bibendum nisl. Pellentesque justo lectus, molestie vel luctus sed, lobortis in libero. Nulla facilisi. Aliquam erat volutpat. Suspendisse vitae nunc nunc. Sed aliquet est suscipit sapien rhoncus non adipiscing nibh consequat. Aliquam metus urna, faucibus eu vulputate non, luctus eu justo.

Donec urna leo, vulputate vitae porta eu, vehicula blandit libero. Phasellus eget massa et leo condimentum mollis. Nullam molestie, justo at pellentesque vulputate, sapien velit ornare diam, nec gravida lacus augue non diam. Integer mattis lacus id libero ultrices sit amet mollis neque molestie. Integer ut leo eget mi volutpat congue. Vivamus sodales, turpis id venenatis placerat, tellus purus adipiscing magna, eu aliquam nibh dolor id nibh. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Sed cursus convallis quam nec vehicula. Sed vulputate neque eget odio fringilla ac sodales urna feugiat.

Phasellus nisi quam, volutpat non ullamcorper eget, congue fringilla leo. Cras et erat et nibh placerat commodo id ornare est. Nulla facilisi. Aenean pulvinar scelerisque eros eget interdum. Nunc pulvinar magna ut felis varius in hendrerit dolor accumsan. Nunc pellentesque magna quis magna bibendum non laoreet erat tincidunt. Nulla facilisi.

Duis eget massa sem, gravida interdum ipsum. Nulla nunc nisl, hendrerit sit amet commodo vel, varius id tellus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac dolor est. Suspendisse ultrices tincidunt metus eget accumsan. Nullam facilisis, justo vitae convallis sollicitudin, eros augue malesuada metus, nec sagittis diam nibh ut sapien. Duis blandit lectus vitae lorem aliquam nec euismod nisi volutpat. Vestibulum ornare dictum tortor, at faucibus justo tempor non. Nulla facilisi. Cras non massa nunc, eget euismod purus. Nunc metus ipsum, euismod a consectetur vel, hendrerit nec nunc.

Bibliography

[1] Google sponsors lunar x prize to create a space race for a new generation, 2007.

[2] Spaceflight 101. Nasa report on antares launch failure places blame on aj26 engines, 2015.

[3] Olivier Aycard and Richard Washington. State identification for planetary rovers: Learning and recognition. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1163–1168. IEEE, 2000.

[4] Benjamin Bach, Conglei Shi, Nicolas Heulot, Tara Madhyastha, Tom Grabowski, and Pierre Dragicevic. Time curves: Folding time to visualize patterns of temporal evolution in data. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):559–568, 2016.

[5] Charles Castel, Jean-François Gabard, Catherine Tessier, Bertrand Laborde, and Raymond Soumagne. Fdir strategies for autonomous satellite formations-a preliminary report. In *Symposium Spacecraft Autonomy: Using AI to Expand Human Space Exploration*, 2006.

[6] Dauna Coulter. Down the lunar rabbit-hole, 2010.

[7] Richard Dearden, Thomas Willeke, Reid Simmons, Vandi Verma, Frank Hutter, and Sebastian Thrun. Real-time fault detection and situational awareness for rovers: Report on the mars technology program task. In *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, volume 2, pages 826–840. IEEE, 2004.

[8] Michael Friendly. Corrgrams: Exploratory displays for correlation matrices. *The American Statistician*, 56(4):316–324, 2002.

[9] Robert Hammett and Robert Luppold. Application of syndrome pattern-matching approach to fault isolation in avionic systems. In *Digital Avionics Systems Conference, 1991. Proceedings., IEEE/AIAA 10th*, pages 56–61. IEEE, 1991.

[10] Niklas Holsti and Matti Paakko. Towards advanced fdir components. *Data Systems in Aerospace, DASIA 2001*, 2001.

[11] Inseok Hwang, Sungwan Kim, Youdan Kim, and Chze Eng Seah. A survey of fault detection, isolation, and reconfiguration methods. *Control Systems Technology, IEEE Transactions on*, 18(3):636–653, 2010.

[12] James Kurien and María DR Moreno. Intrinsic hurdles in applying automated diagnosis and recovery to spacecraft. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 40(5):945–958, 2010.

[13] Edward C Larson, Eugene B Parker Jr, and Brian R Clark. Model-based sensor and actuator fault detection and isolation. In *American Control Conference, 2002. Proceedings of the 2002*, volume 5, pages 4215–4219. IEEE, 2002.

[14] Jiliang Lin and Jingping Jiang. Fault detection and analysis of control software for a mobile robot. In *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*, volume 1, pages 862–866. IEEE, 2006.

[15] JAXA Institute of Space and Astronautical Science. On the status of the procyon nanosatellite probe, 2015.

[16] Matti Paakko, Petri Myllymäki, Niklas Holsti, and Henry Tirri. Bayesian networks for advanced fdir. In *Proceedings of the ESA Workshop on On-Board Autonomy*, pages 311–318, 2001.

[17] RJ Rummel. Understanding correlation. 1976.

[18] Mark Schwabacher and Robert Waterman. Pre-launch diagnostics for launch vehicles. In *Aerospace Conference, 2008 IEEE*, pages 1–8. IEEE, 2008.

[19] SpaceX. Crs-7 investigation update, 2015.

[20] NASA Independent Review Team. Orb3 accident investigation report. Technical report, National Aeronautics and Space Administration, October 2015.

[21] Unity Technologies. Unity powers nasa virtual mars rover experience, 2012.

[22] Apollo Spacecraft Program Office Test Division. *Apollo 13 Technical Air-to-Ground Voice Transcription*. National Aeronautics and Space Administration Manned Spacecraft Center, 1970.

[23] Massimo Tipaldi and Bernhard Bruenjes. Spacecraft health monitoring and management systems. In *Metrology for Aerospace (MetroAeroSpace), 2014 IEEE*, pages 68–72. IEEE, 2014.

[24] Richard Washington. On-board real-time state and fault identification for rovers. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1175–1181. IEEE, 2000.

[25] Alan S Willsky. A survey of design methods for failure detection in dynamic systems. *Automatica*, 12(6):601–611, 1976.