JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 6, July 2009

Unsupervised Anomaly Detection for Liquid-Fueled Rocket Propulsion Health Monitoring

Mark Schwabacher* and Nikunj Oza†
NASA Ames Research Center, Moffett Field, CA 94035

and

Bryan Matthews‡
Stinger Ghaffarian Technolgies, Inc., Moffett Field, CA 94035

DOI: 10.2514/1.42783

This article describes the results of applying four unsupervised anomaly detection algorithms to data from two rocket propulsion testbeds. The first testbed uses historical data from the Space Shuttle Main Engine. The second testbed uses data from an experimental rocket engine test stand located at NASA Stennis Space Center. The article describes nine anomalies detected by the four algorithms. The four algorithms use four different definitions of anomalousness. Orca uses a nearest-neighbor approach, defining a point to be an anomaly if its nearest neighbors in the data space are far away from it. The Inductive Monitoring System clusters the training data, and then uses the distance to the nearest cluster as its measure of anomalousness. GritBot learns rules from the training data, and then classifies points as anomalous if they violate these rules. One-class support vector machines map the data into a high-dimensional space in which most of the normal points are on one side of a hyperplane, and then classify points on the other side of the hyperplane as anomalous. Because of these different definitions of anomalousness, different algorithms detect different anomalies. We therefore conclude that it is useful to use multiple algorithms.

I. Introduction

THE ability to detect anomalies in sensor data from a complex engineered system such as a spacecraft is important for at least three reasons. First, detecting anomalies in near-real time during flight can be helpful in making crucial decisions such as whether to abort the launch of a spacecraft prior to reaching the intended altitude. Second, for a reusable spacecraft such as the Space Shuttle, detecting anomalies in recorded sensor data after a flight can help to determine what maintenance is or is not needed before the next flight. Third, the detection of recurring anomalies in historical data covering a series of flights can produce engineering knowledge that can lead to design improvements.

The current approach to detecting anomalies in spacecraft sensor data is to use large numbers of human experts. Flight controllers watch the data in near-real time during each flight. Engineers study the data after each flight. These experts are aided by limit checks that signal when a particular variable goes outside of a predetermined range. The current approach is very labor intensive. Also, humans may not be able to recognize faults that involve relationships among large numbers of variables. Further, some potential faults could happen too quickly for humans to detect them

Received 16 December 2008; accepted for publication 24 April 2009.

* Computer Scientist, Intelligent Systems Division, MS 269-3, AIAA Member, mark.a.schwabacher@nasa.gov
† Computer Scientist, Intelligent Systems Division, MS 269-2, Nikunj.C.Oza@nasa.gov
‡ Electrical Engineer, Intelligent Systems Division, MS 269-2, Bryan.L.Matthews@nasa.gov

and react before they become catastrophic. On future missions to Mars, there will be a speed-of-light delay of up to 20 min in between when the data are sensed and when flight controllers on Earth first see them, furthering the need for automated anomaly detection.

One approach to automating anomaly detection is the model-based approach. This approach encodes human knowledge into a model, which is then used to automatically detect faults. Examples of systems that use the model-based approach include Livingstone [1,2], HyDE [3], Titan [4], TEAMS-RT [5], RODON [6], SHINE [7], and MEXEC [8]. Building the models is very labor intensive; it therefore may not be feasible to model every part of a highly complex system such as a spacecraft. It also may not be possible to model all possible failure modes. We therefore consider supplementing the model-based approach with the data-driven approach.

The data-driven approach seeks to build a model for detecting anomalies directly from data collected during system operation, rather than building it based on human expertise. In this article, we explore a particular data-driven approach, which is based on unsupervised anomaly detection algorithms.

This article presents nine anomalies that we detected in rocket propulsion data using four unsupervised anomaly detection algorithms. We previously presented these anomalies at two conferences [9,10]. This article consolidates the results from those two conference papers, adds more details about the anomaly detection algorithms we used, and adds some of the lessons that we learned about evaluating the performance of unsupervised anomaly detection algorithms when they are applied to real-world data.

II. Anomaly Detection

Anomaly detection algorithms, also known as outlier detection algorithms, seek to find portions of a data set that are somehow different from the rest of the data set. Some good surveys of anomaly detection algorithms can be found in Refs. [11–13]. A supervised anomaly detection algorithm requires training data consisting of a set of examples of anomalies, and a set of examples of non-anomalous (or nominal) data. From the data, the algorithm learns a model that distinguishes between the nominal and the anomalous data points. Supervised anomaly detection algorithms typically require tens or hundreds of labeled examples of anomalies, plus a similar number of labeled examples of nominal data points, in order to obtain adequate performance. An unsupervised or one-class anomaly detection algorithm is trained using a single set of unlabeled data, most or all of which is assumed to be nominal. It learns a model of the nominal data, which can be used to signal an anomaly when new data fails to match the model. We make a distinction between two different types of unsupervised anomaly detection algorithms. Some algorithms assume that all of the training data are nominal, and learn a model of the nominal data that is guaranteed to cover all of the training data. If such a model is applied to the training data, it will never signal an anomaly. Other algorithms are trained using a data set that consists mostly of nominal data but may or may not contain a small number of anomalies. These algorithms learn a model that covers the vast majority of the training data, and which can be used both to find anomalies in the training data and to find anomalies in previously unseen data. Unsupervised anomaly detection algorithms typically require hundreds of nominal data points in order to obtain adequate performance. In the work described in this article, each data point is a vector of all of the sensor values and commands at one point in time.

We have been using historical data from the Space Shuttle Main Engine (SSME) [14] and from a rocket engine test stand to develop and test algorithms that we hope will be useful for future launch systems such as Ares I [15] and Ares V [16]. For both of these testbeds, the number of examples of anomalies available in historical data is fairly small. Fortunately, the number of examples of anomalies available in real launch systems is also too low for effective use of supervised anomaly detection algorithms. We therefore decided to use unsupervised anomaly detection algorithms, since they do not require labeled examples of anomalies. This article presents nine anomalies detected by four unsupervised anomaly detection algorithms: Orca [17], GritBot [18] (a commercial product), the Inductive Monitoring System (IMS) [19,20] and a one-class support vector machine (SVM) [21]. The next four subsections describe these four algorithms.

A. Orca

Orca [17] uses a nearest-neighbor approach to unsupervised anomaly detection. It defines an anomaly to be a point whose nearest neighbors in data space are far away from it. To measure the distance between two points, it uses a weighted average of the Euclidean distance for the numerical variables and the Hamming distance for the discrete variables. Orca does not assume that all of the training data are nominal, and can be used to find anomalies in the training data as well as in other data sets. Orca reports the “contribution” of each variable to the anomaly score, which is defined to be the amount that the overall anomaly score would decrease if a given variable were given a weight of zero in the distance metric. The obvious algorithm for finding distance-based outliers compares every point with every other point, and therefore has quadratic run time. Orca uses a novel pruning rule to obtain near-linear-time performance, allowing it to scale to very large data sets. The pruning rule works as follows: When searching for the *n* strongest outliers in a data set, Orca keeps track of the distance score for the *n*th top outlier found so far. Then, when searching for the nearest neighbors of a point, it stops the search as soon as it has found enough neighbors close enough to the point to prove that it is not one of the top *n* outliers. In the experiments described in this article, we used Orca to find the 30 strongest outliers in each data set.

In the past we have used Orca to detect anomalies in Earth science data, aviation safety data, aviation security data, data from the International Space Station, and data from the sensors on the leading edges of the Space Shuttle’s wings.

B. GritBot

GritBot is a commercial product from Ross Quinlan’s RuleQuest Research [18]. Rather than just looking for points that are anomalous with respect to the entire data set, GritBot searches for subsets of the data set in which an anomaly is apparent. For each anomalous point, it reports a description of the relevant subset of the data set, based on values of discrete variables or ranges of continuous variables, in which the target variable usually has a particular value (if it is discrete) or range of values (if it is continuous). The point is considered to be an anomaly because the target variable at that point is significantly different from the value of the target variable at the vast majority of the other points in the subset. Like Orca, GritBot assumes that the training data could contain a small number of anomalies, and can be used to find anomalies in the training data. We used GritBot’s default parameters.

C. Inductive Monitoring System

The IMS [19,20] is similar to Orca in that it is distance-based. Like Orca, it uses Euclidean distance as its distance metric. Unlike Orca, it does not explicitly support discrete variables, so we did not include any discrete variables in our experiments with IMS. The major difference between Orca and IMS is that during the training step, IMS clusters the nominal training data into clusters representing different modes of the system. Each cluster is represented using the smallest bounding hyperbox containing the points in the cluster. At run time, IMS uses the distance to the bounding hyperbox of the nearest cluster as an anomaly measure. IMS assumes that all of the training data are guaranteed to be nominal, and will always return zero as the anomaly score when tested on the training data, since all of the training data are within the bounding hyperboxes found in the training data.

Before applying IMS to the SSME data, we pre-processed the data by first normalizing the data to have zero mean and unit variance, then clustering the variables into sets of variables that are highly correlated (such as would be expected with redundant sensors). The clustering was performed using the “clusterdata” function from the Matlab Statistics Toolbox [22], which produces an agglomerative hierarchical clustering tree. The clustering threshold was set at the first break from the one class node. The resulting variables in each cluster were averaged to produce a single representative variable for each cluster that was input to IMS. The clustering and normalization were both based on the training data.

IMS is also currently being used to detect anomalies in data from the International Space Station and in data from an electrical power system testbed, and in the past was used to detect anomalies in data from sensors on the leading edges of the Space Shuttle’s wings.

D. One-class SVM

One-class SVMs [21,23] seek to describe the range of normal training data in such a way as to enable the resulting model to distinguish normal data from abnormal data in the future. Like Orca and GritBot, they assume that the training data may contain a small number of anomalies, and learn a model that covers the vast majority of the training data. The name “one-class SVM” is due to the possibility that only one class of data (normal data) may be available during training (if abnormal training data are available, they can be used). One-class SVMs first map the training data from the original *data* space into a much higher-dimensional or possibly infinite-dimensional *feature* space and then find a linear model (hyperplane) in that feature space that allows almost all the normal data to be on one side (and to be separate from abnormal training data if available). The idea is that linear models in a higher-dimensional space correspond to more complicated nonlinear models in the original data space. The key to this technique working, even when the feature space is infinite dimensional is the "kernel trick." In particular, training and test cases are only represented using a kernel function that returns distances between pairs of examples. In the work described in this article, we chose the radial basis function (RBF) kernel, which returns the distance between examples x_i and x_j as

$$K(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|}{2\sigma^2}\right).$$

We chose the RBF kernel because it is the most frequently used kernel. This kernel function, since it satisfies Mercer's conditions [21,23] has the form

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j),$$

where Φ is a function that maps its inputs from the data space into an infinite-dimensional space. However, this infinite-dimensional space is induced by the kernel matrix and is never explicitly represented. The Φ's only appear in dot products, and the dot products of infinite-dimensional vectors are scalars. Therefore, one-class SVMs give us the power of mapping our data into a very high-dimensional or infinite-dimensional space while still being mathematically tractable. However, even though the metric used to represent distances in K is often interpretable in terms of the problem being solved, the induced feature space is rarely interpretable in this way.

The use of linear models allows SVMs to retain the benefit that the algorithm finds the globally optimal solution given the training set, while still effectively using nonlinear models. For each test point, the one-class SVM returns a measure of how strongly normal or anomalous the data point is. In particular, the SVM learning algorithm returns a function $f(x)$ that returns a positive number if x falls within the range of normal data in the feature space and a negative number otherwise, and the further $f(x)$ is from zero, the more normal or abnormal x is. For the experiments in this article, we used the freely available OSU-SVM MATLAB package [24] which uses the LIBSVM library [25]. One-class SVMs do not explicitly support discrete variables; therefore, we discarded them for our one-class SVM tests. One-class SVMs require the user to set the maximum fraction of training points that can be outliers, for which we choose 0.0025 in our experiments since it gave reasonable results. We then chose a value of σ for the RBF kernel using the procedure described in Ref. [26], which is to choose the lowest value such that our designated fraction 0.0025 of the training points are classified as abnormal.

III. Testbeds

We used two testbeds to test the anomaly detection algorithms: The SSME and a rocket engine test stand at NASA Stennis Space Center (SSC).

A. Space Shuttle Main Engine

The Space Shuttle propulsion system consists of two solid rocket boosters, and three SSMEs [14]. The SSMEs are located on the Orbiter. During the initial ascent, all five engines are fired together. Approximately 2 min after launch, the solid rocket boosters run out of fuel and are jettisoned, and the Shuttle continues its ascent using only the three SSMEs. The SSMEs are liquid-fueled rocket engines employing cryogenic liquid hydrogen and liquid oxygen as propellants. These propellants are stored in the external tank, which is jettisoned when the Shuttle reaches its intended orbit, about 9 min after launch.

The SSMEs are reusable. After each flight, they are removed from the Orbiter, inspected, serviced as necessary, and then installed onto an Orbiter (which is not necessarily the same orbiter). Each engine is periodically acceptance tested by firing it on the ground using test stands located at NASA SSC.

Each SSME has approximately 90 sensors measuring quantities such as temperature, pressure, fuel flow rate, rotational velocity, and vibration. Many of the sensors are redundant for reliability reasons. For example, four identical pressure sensors may be placed right next to each other with the expectation that they will all provide approximately the same value. When one of the sensors provides a substantially different value from the other three, it can be inferred that the sensor has failed. When the SSMEs are used on the Shuttle, additional relevant information is available from sensors located in the Shuttle's fuel feed system. When the SSMEs are fired on a test stand, the test stand acts as a fuel feed system, and provides additional relevant information from sensors located on the test stand. A small number of additional sensors are placed directly on the SSME during testing. The SSME results reported in this article use only the data from the sensors that are located on the SSME and available both in flight and on the test stand.

B. Rocket Engine Test Stands

NASA SSC in Mississippi operates several rocket engine test stands. Each test stand provides a structure strong enough to hold a rocket engine in place as it is fired, and a fuel feed system to provide fuel to the engine. Test stands A-1 and A-2 are large test stands used to test the SSMEs. Test stand E-1 is a much smaller test stand used to test experimental rocket engines [27]. Test stand E-1 is being used as a testbed for a variety of integrated systems health management (ISHM) technologies. It has numerous sensors on its fuel feed system, which are analogous to the sensors on a spacecraft's fuel feed system. In this article, we present the results of applying anomaly detection algorithms to these sensor data from Test Stand E-1. Note that in this analysis, unlike the SSME analysis discussed in the previous subsection, only facility support data was used — that is, we used data from Test Stand E-1, and not from the test article (the rocket engine being tested).

IV. Results

In our tests, the four algorithms successfully detected one major system failure, and several sensor failures. They also detected some other anomalies that are not considered to be failures. In this section, we review these results.

In each case described in this section, we used an anomaly detection algorithm to detect anomalies in data from one rocket engine test. In some cases, we used data from a different test as training data. Each SSME test had 129 continuous variables, 18 discrete variables, and about 13,000 time steps. The continuous variables came from a variety of sensors including pressure, temperature, and vibration sensors. The discrete variables were categorical fault codes and status codes provided by the SSME controller. In some cases, we also included the first and second derivatives of the continuous variables with respect to time in the training and test data, so that the algorithms could detect unusual rates of change in the sensor values in addition to detecting unusual sensor values. Because IMS and the one-class SVM do not explicitly support discrete variables, we ran these algorithms only on the continuous variables in the SSME data. The CPU time needed to run each algorithm on the SSME data is shown in Table 1. Because of software licensing issues, we were unable to run all of the algorithms on the same computer, so the table includes the CPU type and clock speed used to run each algorithm. Each SSME test lasted about 8 min, so all four algorithms were able to process the SSME data faster than real time.

For Test Stand E-1, each test had 73 continuous variables, 87 discrete variables, and about 39,000 time steps. The continuous variables were sensor values, similar to those for the SSME. The discrete variables were all Boolean values indicating whether valves in the test stand were open or closed. Since the Test Stand E-1 data had a much larger fraction of discrete variables than did the SSME data, and we believed that these discrete variables were important for anomaly detection, we chose not to use IMS or the one-class SVM for these data. CPU times for the test stand data are also shown in Table 1. Each E-1 test lasted about 2.5 min, so both algorithms processed that data somewhat slower than real time. The data from the E-1 tests had a higher sampling rate than the data from the SSME tests, which prevented the algorithms from being able to process the data in real time. (The sampling rate is the number of times per second that the sensor data is recorded.) Note that the processors we used were slow by today's standards; we believe that both algorithms could process the E-1 data in real time using faster processors.

Table 1 Approximate CPU times of the four algorithms for two data sets

Algorithm	CPU type	CPU time for SSME data	CPU time for Test Stand E-1 data
Orca	500 MHz Sun Blade 100	3 min	12 min
GritBot	1.5 GHz Intel Pentium M	5 min	7 min
IMS	2.6 GHz dual-core Intel Xeon	3 min	N/A
1-class SVM	2.16 GHz Intel Core Duo	0.5 s	N/A

The first anomaly that we consider is a failure in the SSME's High Pressure Fuel Turbopump (HPFTP), in which a turbine blade broke during SSME static test 853 in January 1996, approximately 130 s into the test [28]. We tested the ability of the algorithms to detect this known failure by training them on data from test 851, which was nominal, and then testing them on tests 852 (which was also nominal) and 853 (which had the failure). Figure 1 shows the IMS outputs for tests 852 and 853. Both graphs plot the IMS output against time for each test. The IMS output is the distance in the normalized multidimensional space from the new data point at each time step to the bounding hyperbox of the nearest cluster in the training data. Larger IMS outputs indicate that the data are more anomalous. Notice that in test 852, the IMS output remained well below 50 for the entire test. In test 853, the IMS output remained well below 50 until 130 s, and then became much larger. These results show that IMS successfully detected the failure. If a threshold of an IMS score of 50 (as shown in Fig. 1) were used to signal an alarm, then IMS would signal the alarm at the correct time in test 853, without signaling any false alarms in test 852. We believe that later spikes in the IMS output may correspond to additional failures in the SSME that occurred as the broken turbine blade traveled downstream in the engine. Orca, GritBot, and one-class SVM were also able to successfully detect the HPFTP failure in test 853.

Figure 2 shows the output of the one-class SVM when it was tested on tests 852 and 853 after being trained on test 851. The horizontal axis displays the time in seconds from the start of the test just as in Fig. 1. The vertical axis

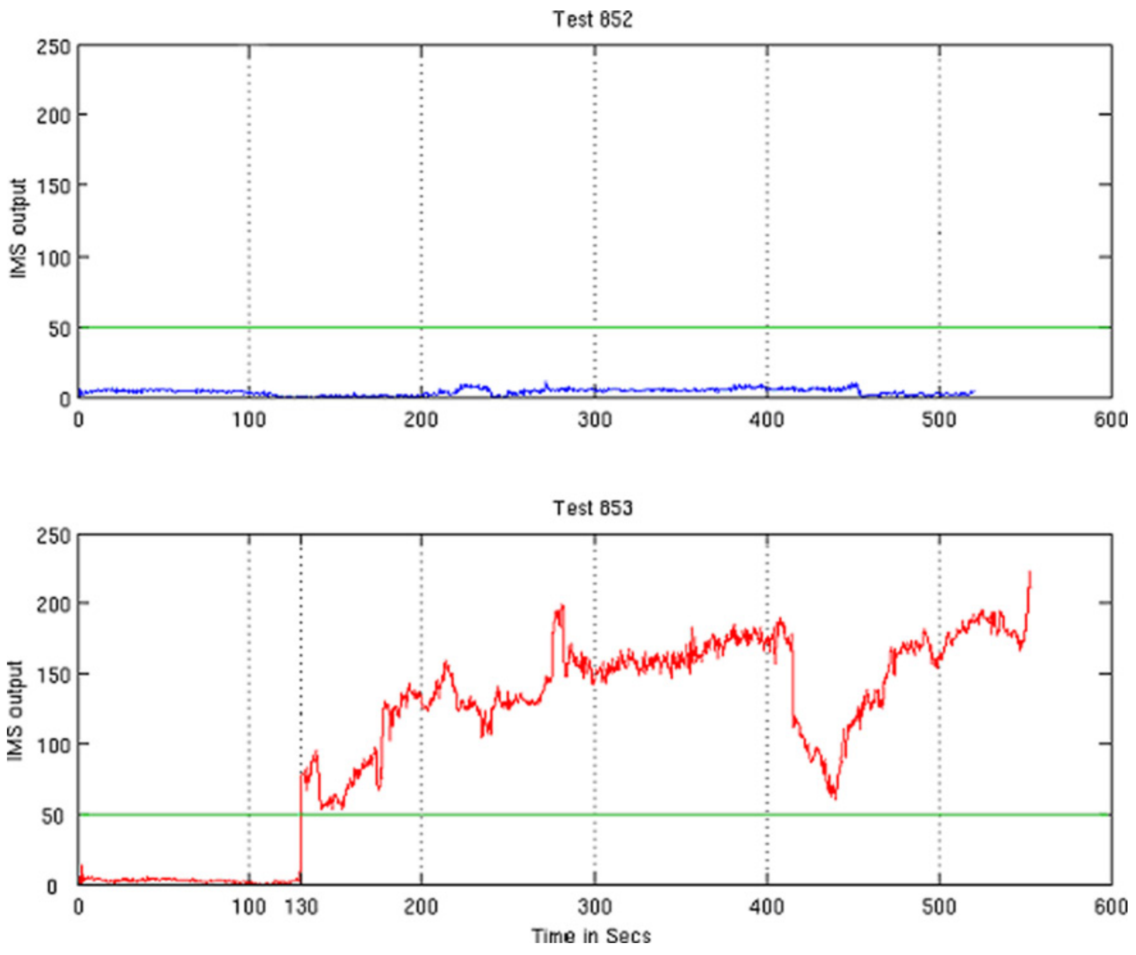

Fig. 1 IMS was trained on test 851 (which was nominal) and then tested on test 852 (top, also nominal) and test 853 (bottom, with an HPFTP failure at 130 s). The horizontal line at 50 output units represents a candidate threshold above which values are considered to be anomalous.

shows the output of the one-class SVM. Values greater than zero indicate that the one-class SVM thinks the system is behaving normally, with higher values indicating greater normality. Values less than zero indicate abnormal system operation, with lower values indicating more anomalous operation. At the start of both tests 852 and 853, the one-class SVM gives a very large range of values corresponding to startup transients. The one-class SVM correctly indicates that the remainder of test 852 demonstrated normal behavior. For test 853, the one-class SVM first indicates abnormal operation at 115 s, which is 15 s before the HPFTP failure occurred. At 130 s, the output goes lower, indicating a greater level of abnormality. The SVM output then returns to a normal level, gradually decreases, and then at 276 s returns to indicating abnormality. The one-class SVM only indicates whether the system as a whole is operating normally or abnormally, and does not indicate specific variables or systems that may be operating abnormally. Future work will need to determine the exact source of the possible precursor at 115 s, the reason for the return to normality at 140 s, and the reason for the return to abnormality at 276 s. We suspect that the return to abnormality at 276 s may have been caused by a secondary failure in the SSME, as the broken turbine blade traveled downstream in the engine and caused other failures.

Using Orca, we determined that it is possible to easily detect the HPFTP failure using a single sensor at a particular frequency. Figure 3 is a reproduction of a figure from Ref. [28], showing that the failure can be detected using a

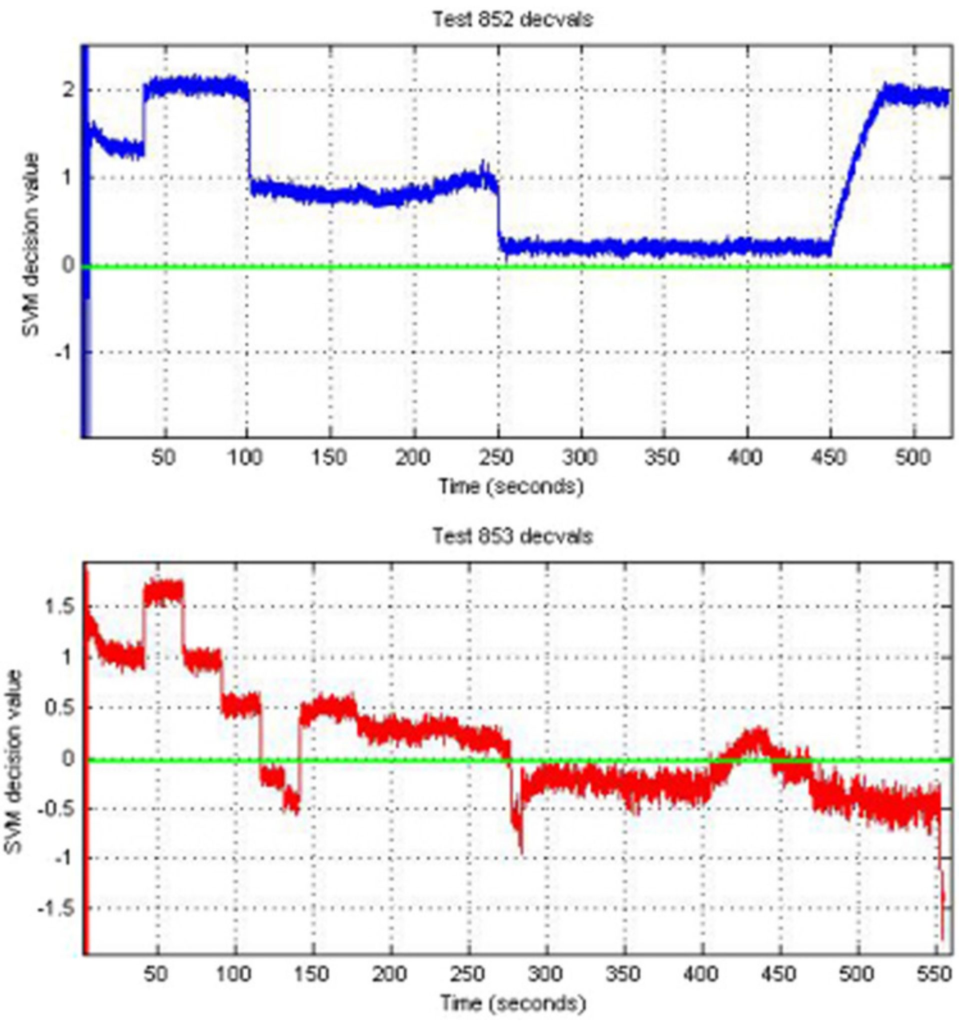

Fig. 2 The one-class SVM was trained on test 851 and then tested on tests 852 (top) and 853 (bottom). Negative values indicate abnormal system behavior.

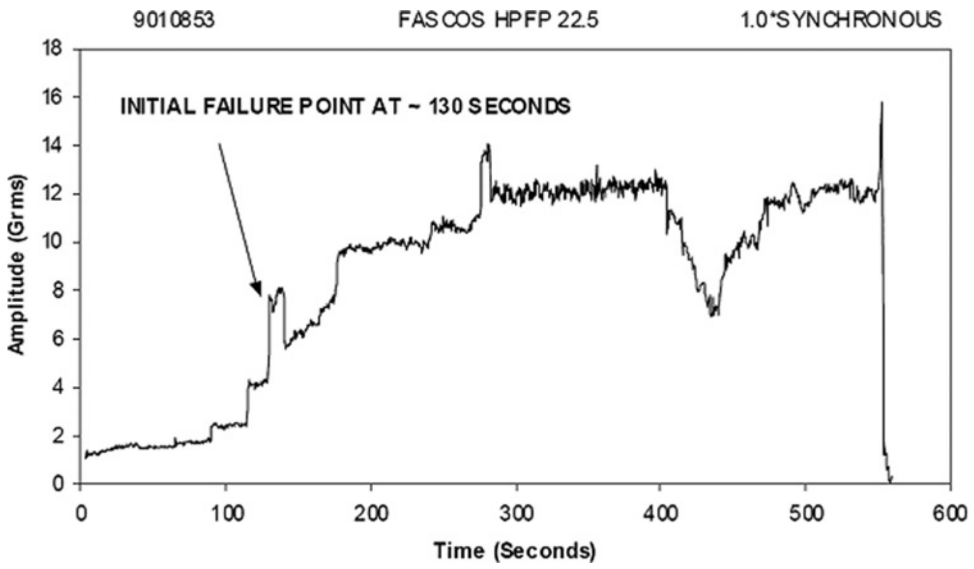

Fig. 3 The HPFTP failure can be detected using a particular vibration sensor at a particular frequency. This figure is reproduced from Ref. [28].

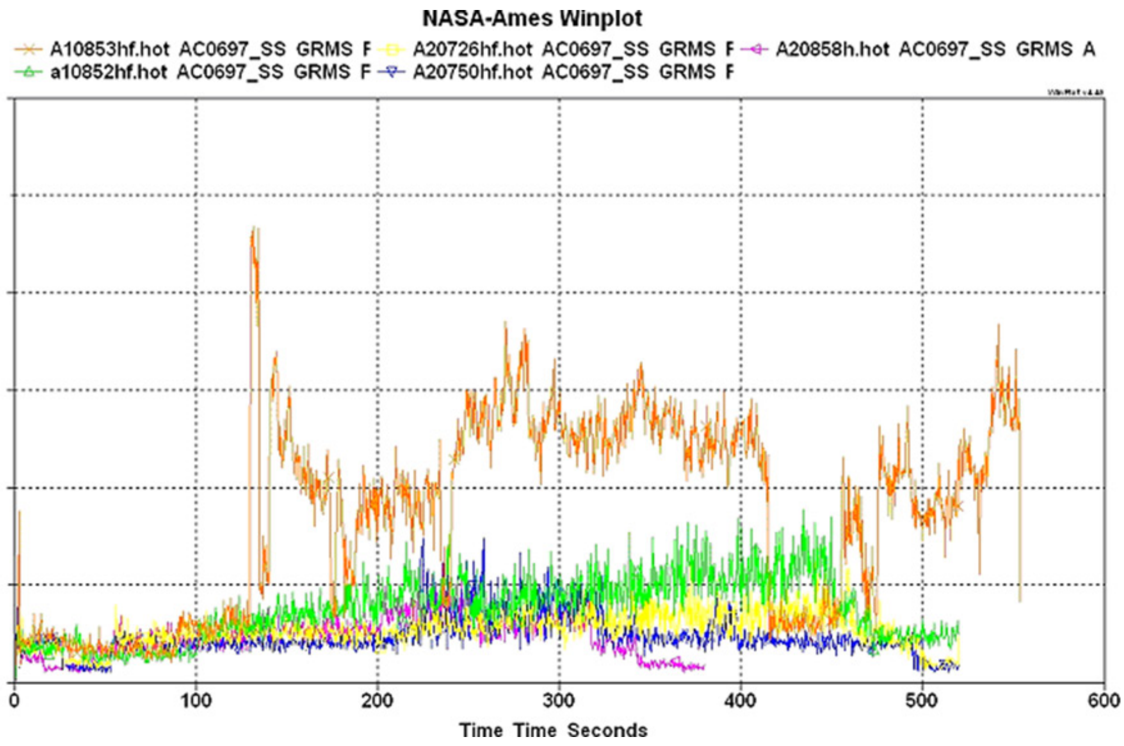

Fig. 4 The five curves represent the same vibration variable during five different SSME tests. The top curve, for test 853, shows the failure at 130 s.

particular vibration sensor at a particular frequency. When Orca discovered the HPFTP failure, it reported that the strongest contribution to the anomaly was a different variable representing a different vibration sensor at a different frequency. This variable is shown in Fig. 4, for five different SSME tests. (Note that in Fig. 4 and in all subsequent figures, the values have been removed from the Y-axis to protect the confidentiality of the data.) For test 853, there is a large increase in this vibration measure at 130 s, which is the time of the failure. By examining the other four tests, we see that the selected variable remains much lower during these other tests. We also note that in test 852—the previous test of the same engine on the same test stand, and the second highest curve in Fig. 4—there appears to be a gradual increase in vibration. Our experts disagreed with each other regarding whether or not this increase represents a precursor of the HPFTP failure.

One difference between GritBot and the other three algorithms is that while the other three algorithms can produce an anomaly score for every time step (which can then be plotted), GritBot only reports an anomaly score for the points that it considers to be outliers. It is therefore not possible to produce a graph (like Fig. 1 or Fig. 2) to illustrate how GritBot detected the HPFTP failure. GritBot, when run with its default parameters, detected 25 anomalies in Test 853, of which the HPFTP failure at 130 s was the second strongest anomaly. The strongest anomaly appears to be a false alarm during engine shutdown.

In addition to detecting the known failure in the HPFTP, the algorithms also discovered many sensor failures that were not previously known to us. We present four of them here.

Figure 5 shows the values from two redundant temperature sensors for the same test. The two sensors return approximately the same value as each other until just after 300 s, when one of the sensors apparently has a failure that causes it to return a constant high value. GritBot detected this sensor failure. In Fig. 5 and subsequent figures, the arrows each point to the approximate point in time at which the first anomaly occurred.

Figure 6 shows another temperature sensor failure that was detected by Orca. This failure, however, was detected not by comparing two redundant sensors during the same test, but by comparing two tests of two different engines. Orca was trained on one engine test and tested on the other engine test. It noticed that the temperature profiles from a particular sensor diverged sharply at around 400 s, indicating a sensor failure.

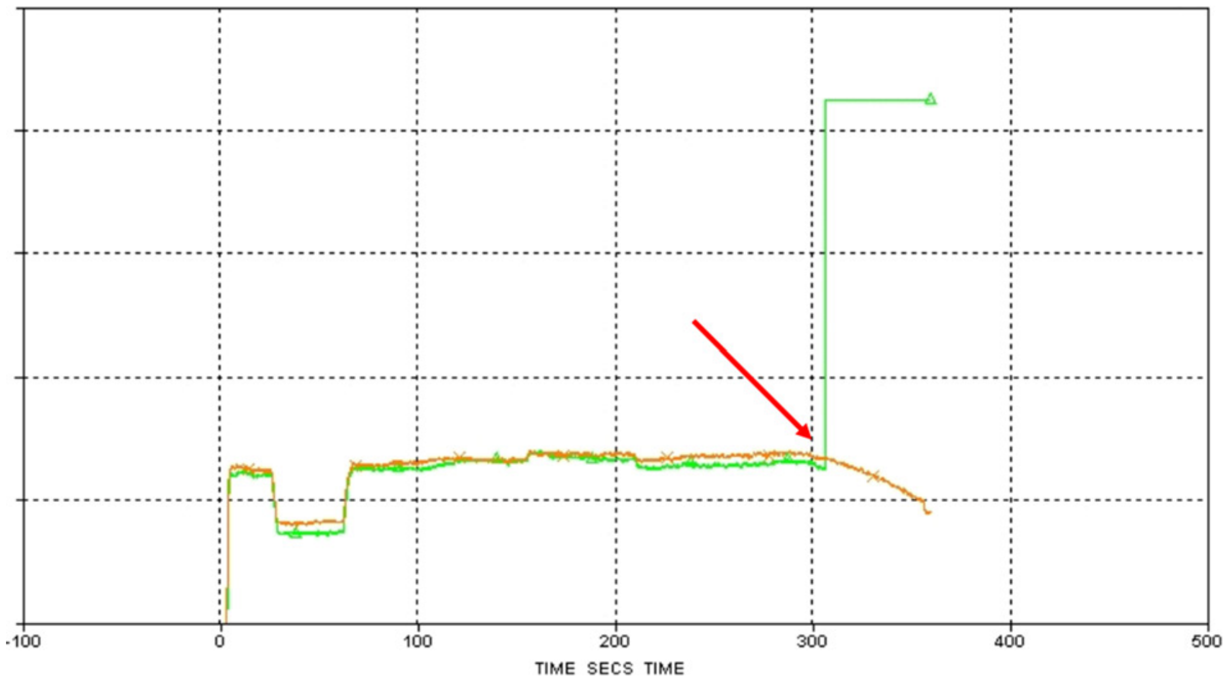

Fig. 5 GritBot detected a temperature sensor failure, shortly after 300 s. This graph shows two redundant temperature sensors for the same test of the same engine.

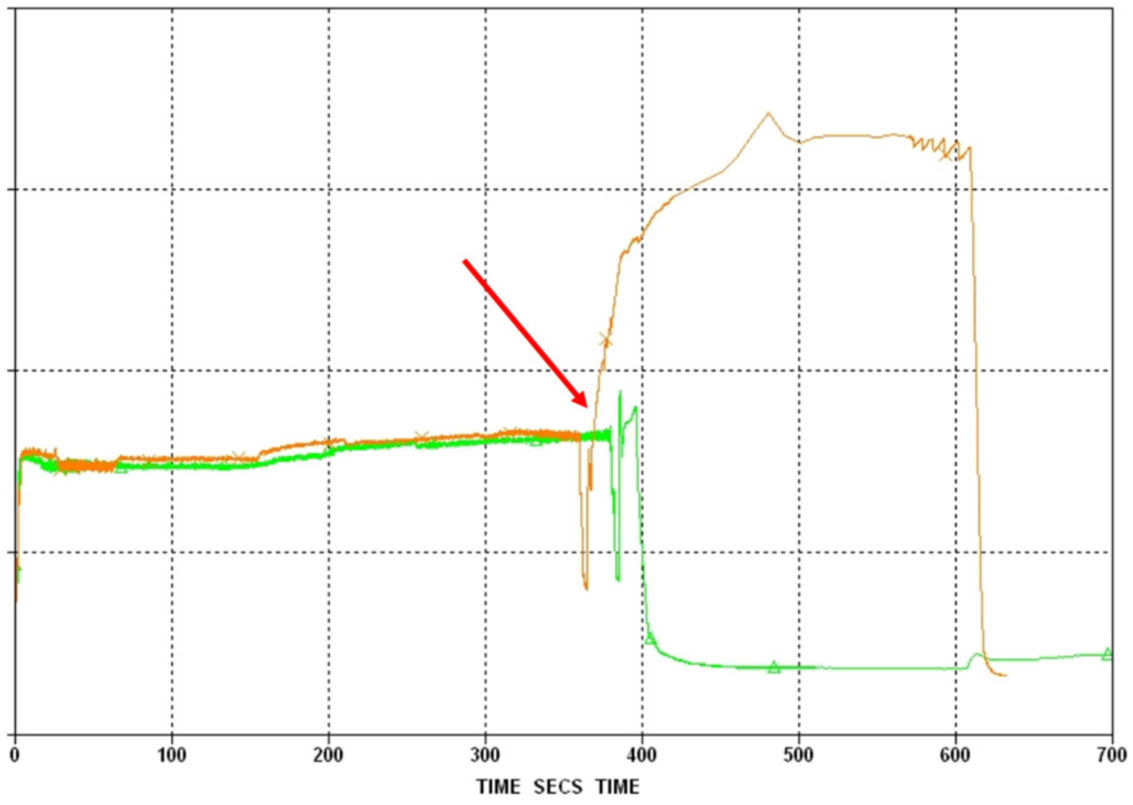

Fig. 6 Orca detected a temperature sensor failure. This graph shows the same temperature sensor across two different tests of two different engines.

Figure 7 shows a pressure sensor failure detected by Orca. This particular failure is more difficult to detect than the sensor failures in the previous figures, because the values from the failed sensor remain within the normal range. Orca detected the failure by noticing that the main combustion chamber controller reference pressure (shown in the bottom curve) and the main combustion chamber hot gas injection pressure sensor (shown in the top curve) are usually strongly correlated, but that the correlation is violated during this test. Our experts explained that this was a notorious sensor failure caused by moisture in the line freezing.

Figure 8 shows a vibration sensor failure detected by Orca. This figure shows the same vibration sensor as Fig. 4, but includes an additional test that was excluded from Fig. 4 because of the sensor failure. Figure 8 has a different scale from Fig. 4; the five curves from Fig. 4 can be seen at the bottom of Fig. 8, but they appear much smaller in Fig. 8 than they do in Fig. 4 because of the different scale. The HPFTP failure can still be seen in the second highest curve, but it is greatly overshadowed by the top curve. This particular vibration sensor returned values approximately ten times higher in the test corresponding to the top curve than it did in the other five tests. Our experts confirmed that it is a sensor failure.

The final four anomalies that we consider are not system failures or sensor failures, but instead are other types of anomalies that occur in the data. The first of these is shown in Fig. 9. The SSME has redundant fuel flow sensors. Figure 9 shows the values of two of those sensors during a short period of time (0.6 s). The values of the two sensors are usually almost exactly the same. However, at approximately 427.35 s, they differed substantially for one time step. GritBot considered this difference to be an anomaly. GritBot also detected 140 other similar anomalies involving differences between redundant fuel flow sensors; the one shown in Fig. 9 was the strongest anomaly. Our domain experts could not explain the difference between the two sensor values, but they noted that this type of small difference occurs fairly regularly, that the small errors occur in both directions, and that the values are averaged before being used to make a decision. They therefore felt that the sensor anomaly was not a reason for concern.

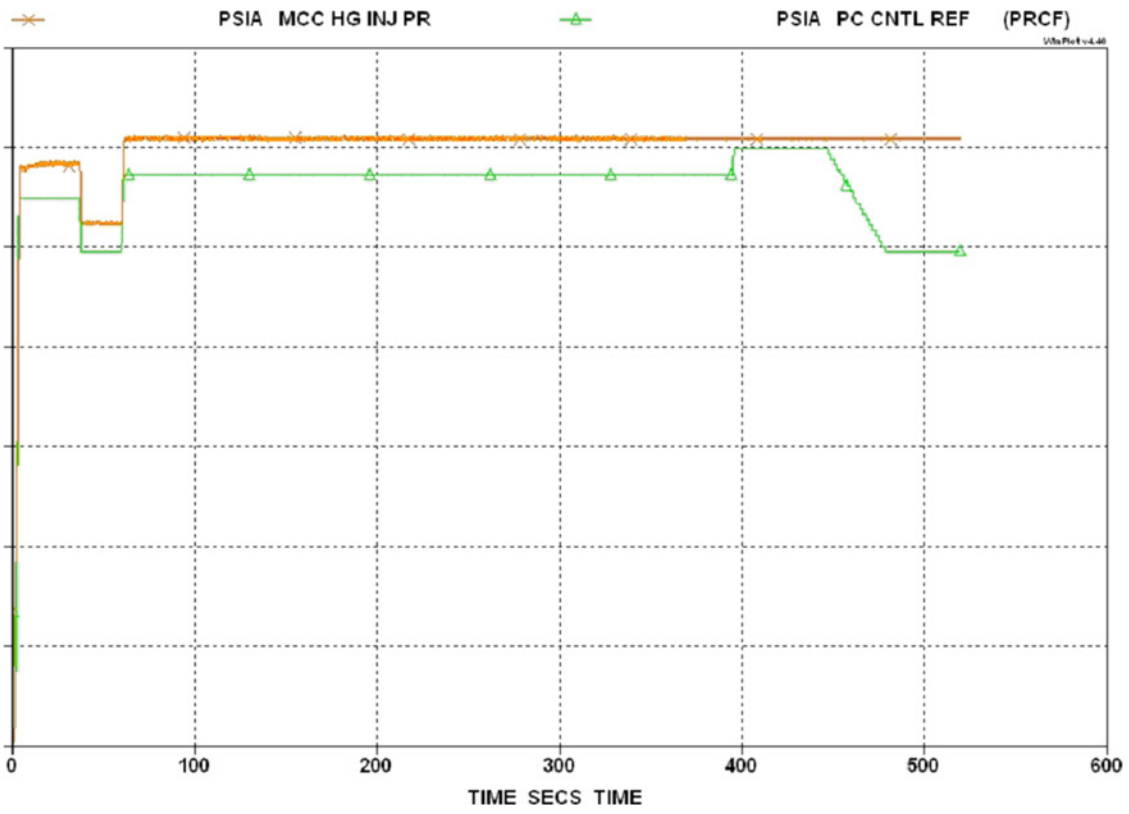

Fig. 7 Orca detected a pressure sensor failure. This graph shows the main combustion chamber controller reference pressure (bottom curve) and the main combustion chamber hot gas injection pressure sensor (top curve). The latter sensor apparently froze sometime before 400 s.

The next anomaly we consider is shown in Fig. 10. Mixture ratio is the oxidizer to fuel ratio of the SSME's main combustion chamber. By definition, it cannot be negative. GritBot observed that this variable was almost never negative in the training data, and then detected several points at which it was negative in the test data, all within the first 2 s of a test. The domain experts said that negative reported mixture ratio at the beginning of a test is a known and well-understood artifact of the way in which mixture ratio is calculated. The SSME does not have an oxidizer flow meter, so the ratio cannot be calculated directly. Instead it is calculated indirectly. The indirect calculation produces reasonably accurate values, except during startup.

The next candidate anomaly that we consider is shown in Fig. 11. The voltage on controller bus 2 usually stays within a fairly narrow range. Orca detected one point in time, approximately 2.5 s after the beginning of the test, at which the voltage dropped to an unusually low level. The domain experts said that the main igniters fire at that point in time, which causes the drop in voltage. They already knew about the drop in voltage, and did not consider it to be an anomaly.

The final candidate anomaly that we consider is from Rocket Engine Test Stand E-1. Some of the variables for E-1 are redline enablers — these are discrete control inputs that enable or disable a redline, which is a threshold on a sensor reading beyond which an automatic shutdown is triggered. There are some pairs of redline enablers that tend to be turned on or off at the same time. GritBot noticed that one particular pair of redline enablers had the same on or off value at every timestep in a test except for one timestep: one redline enabler was turned on one timestep later than the other one. GritBot considered this one point where the two redline enablers had opposite Boolean values to be an anomaly. Orca found some similar discrepancies between redline enablers. Our domain experts felt that these very small delays in between enabling one redline and another redline were not significant and should not be considered anomalies.

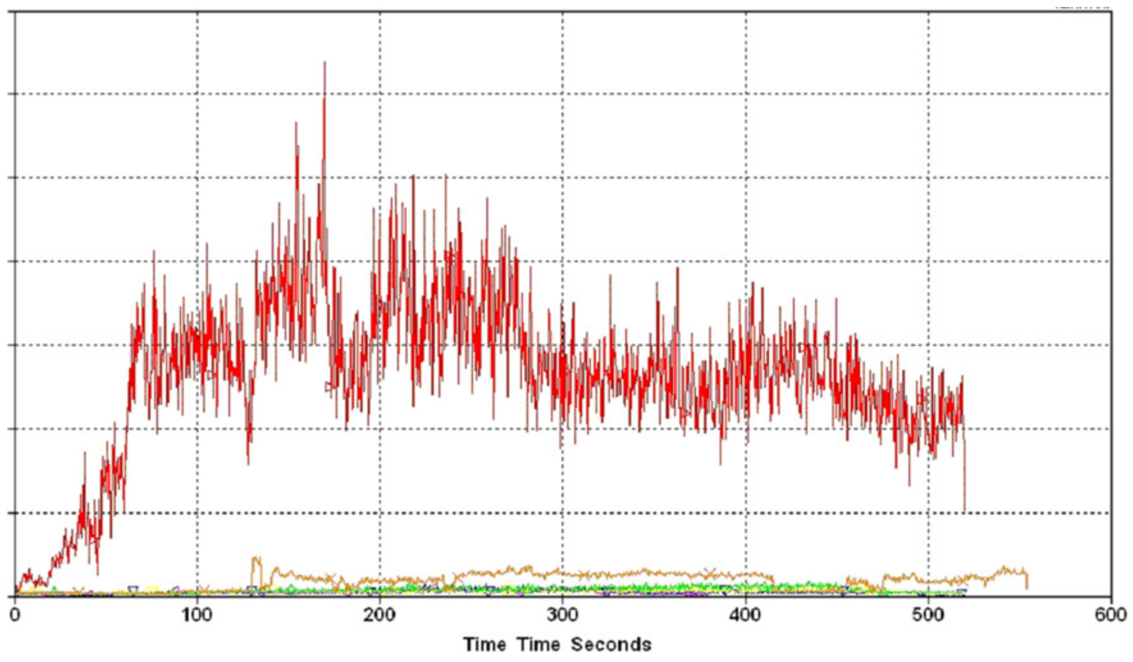

Fig. 8 Orca detected a vibration sensor failure. This graph shows the same sensor that is shown in Fig. 4, but with a sixth test added (top curve). The second curve from the top shows the sensor failure, as before, at 130 s. The top curve represents a sensor failure.

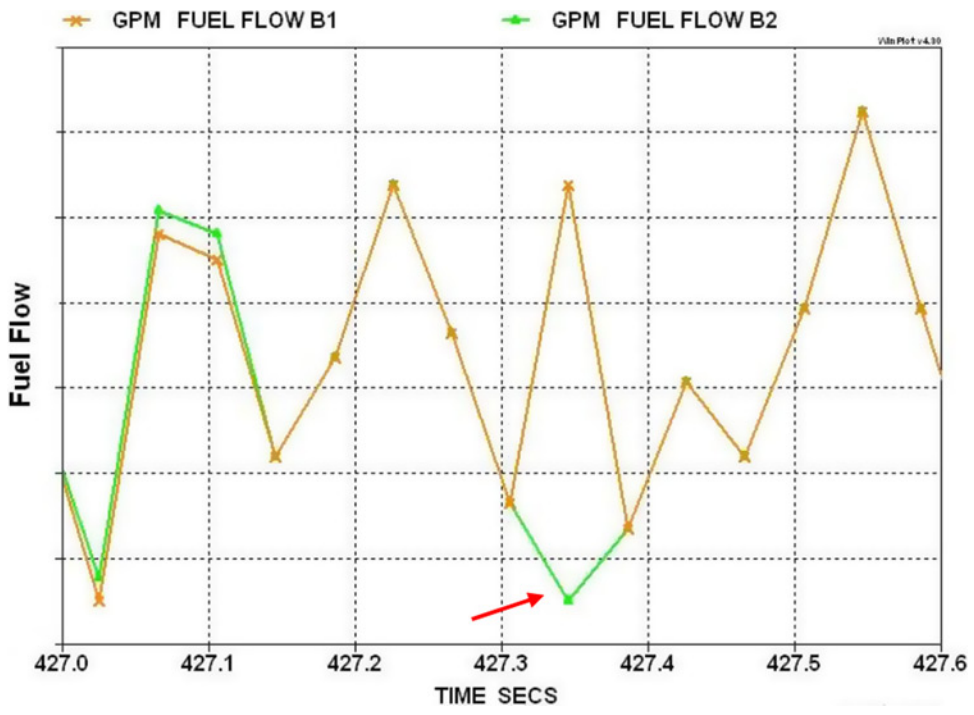

Fig. 9 SSME fuel flow anomaly. The two redundant fuel flow sensors differ significantly at approximately 427.35 s.

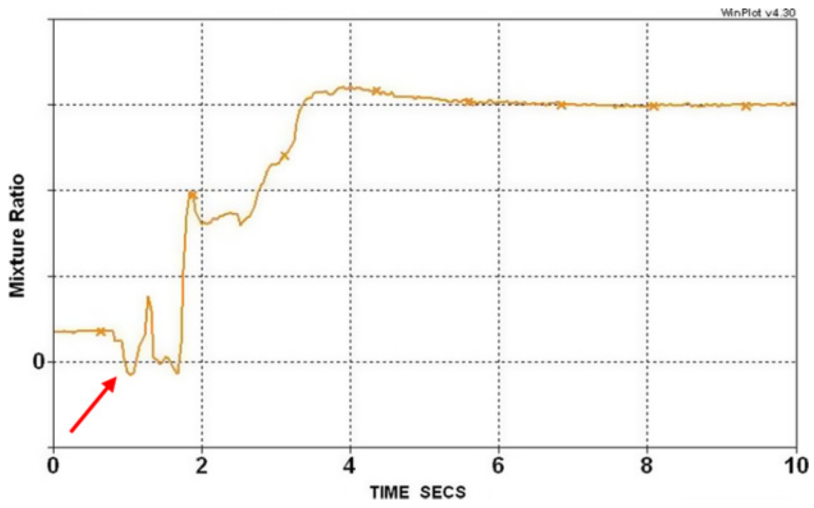

Fig. 10 SSME mixture ratio anomaly. Mixture ratio should always be positive, but it became negative at approximately 1 s.

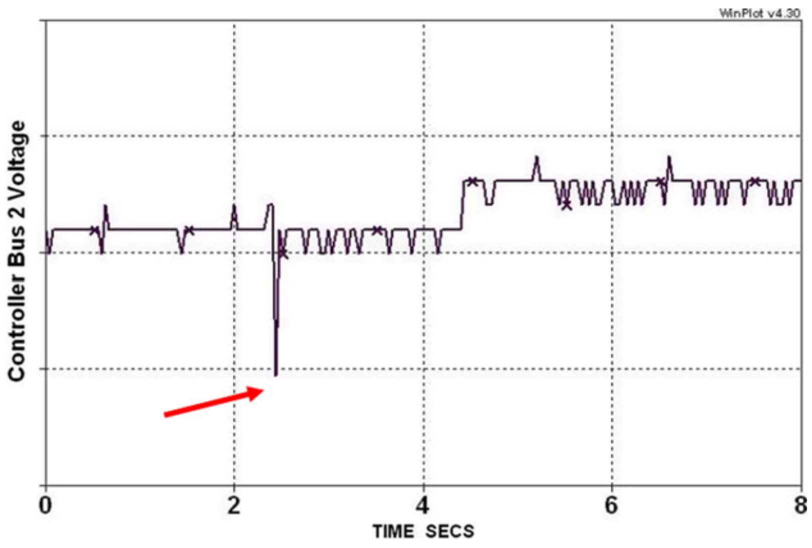

Fig. 11 SSME controller bus voltage anomaly. The bus voltage dropped for one timestep when the main igniters fired.

Table 2 Summary of which failures were detected by which algorithms

Anomaly	Orca	GritBot	IMS	SVM
HPFTP failure	Detected	Detected	Detected	Detected
Temperature sensor failure #1		Detected		
Temperature sensor failure #2	Detected			
Pressure sensor failure	Detected			
Vibration sensor failure	Detected			

Table 2 summarizes the five failures detected by the four algorithms. The one system failure was detected by all four algorithms, but the four sensor failures were each only detected by one algorithm.

V. Evaluating Unsupervised Anomaly Detection

Evaluating the accuracy of unsupervised anomaly detection algorithms when applied to real-world data is a challenge, since it is generally not known how many anomalies exist in the data. In the machine learning community, classification and regression algorithms are typically evaluated by using cross-validation on the training data. For any application of these supervised algorithms, there must be labeled training data, so it is always possible to use cross-validation on the training data to assess the accuracy of the algorithms on real-world data. For real-world applications of unsupervised learning algorithms, there often are not any labeled data available for evaluation purposes. For unsupervised anomaly detection, it is often not known how many anomalies exist in the data. One approach to evaluating the performance of the algorithms is to evaluate their performance on synthetic data in which the number of anomalies is known. For example, one can randomly sample a large number of points from a Gaussian distribution, then add a small number of points from a different distribution, and then test if the algorithms are able to detect the points from the different distribution as anomalies. We previously used this approach to evaluate the performance of Orca [17]. However, machine learning algorithms perform differently on different problems. An algorithm that performs well on synthetic data will not necessarily perform well in a particular real-world application.

When our experts provided us with the SSME data, they told us about one anomaly that was in the data — the HPFTP failure. All four algorithms successfully detected this anomaly. However, a sample size of one anomaly is far too small for us to draw any statistically significant conclusions about the ability of the algorithms to detect anomalies in the SSME data. Instead, we must rely on subjective evaluations of the algorithms' performance. We do this in two ways. First, we graph the anomaly scores output by two of the algorithms (IMS and one-class SVM) for the HPFTP failure in Figs. 1 and 2. Subjectively comparing Fig. 1 with Fig. 2, it appears that IMS did a somewhat better job of detecting the HPFTP failure than the one-class SVM did, because the IMS score stays very close to zero before the failure, and then stays at a much higher level after the failure, and because the one-class SVM had a false alarm at the very beginning of each test, while IMS did not. However, the one-class SVM detected a possible precursor about 15 s before the failure, which IMS did not detect. Therefore, the choice between the two algorithms may depend on whether it is considered more important to detect possible precursors, or more important to avoid false alarms. An additional factor in this choice is whether the errors that an algorithm makes are predictable. For example, if an algorithm always has false alarms at the very beginning of a test due to startup transients that are known to be acceptable, then the false alarms might not impact the algorithm's usability.

We performed a second type of subjective evaluation by using the algorithms to find previously unknown (at least to us) anomalies in the data, and then presenting these anomalies to the rocket propulsion experts and getting their feedback. We first presented these anomalies to the experts with whom we were collaborating. They told us that some of these candidate anomalies were insignificant (Fig. 11, for example), while others were sensor failures (Figs. 5–8). We later presented these discovered anomalies at aerospace conferences [9,10] so that other rocket propulsion experts could evaluate them. We argue that our application is a success because the algorithms detected several anomalies that were judged by the experts to be significant and that were not previously known to anyone on our team. Of course, we still do not know how many more anomalies exist in the data (if any), so we do not know what fraction of the anomalies we succeeded at detecting.

We could in theory perform a statistically significant comparison of the algorithms by having experts subjectively evaluate a large number of discovered anomalies. For example, we could use each of the four algorithms to find the top 100 anomalies in a data set. We would then have somewhere between 100 and 400 candidate anomalies, depending on how much overlap there was among the candidate anomalies discovered by the four algorithms. We would then ask a panel of experts to judge the importance of each of these candidate anomalies on a scale of one to ten. We would then be able to perform a statistically significant comparison of the importance of the anomalies detected by each algorithm. This analysis would allow us to calculate the false alarm rate (the percentage of detected anomalies that are judged by the experts not to be "real" anomalies), but would still not allow us to calculate the missed detection rate, since we would still not know how many more "real" anomalies exist in the data. Unfortunately, our domain experts are too busy to spend the time to evaluate such a large number of candidate anomalies. For some anomalies, the experts are able to immediately identify the cause of the anomaly, but for other anomalies the experts need to invest a significant amount of time studying the data, system and related documentation, and test procedures in order to explain each anomaly and judge its importance.

VI. Related Work

A previous conference paper written by two of the authors of this article and two other authors [29] presents a comparison of six unsupervised anomaly detection algorithms, including the four algorithms discussed in this article, plus a Gaussian mixture model and a linear dynamic system. They ran all six algorithms using SSME data from four Space Shuttle flights and two test stand firings for training, and eight Shuttle flights and four test stand firings for validation. Although they acknowledge that they do not have enough data to make statistically significant comparisons of the relative performance of the six algorithms, they conclude that the algorithm with the best accuracy appeared to be either Orca or the one-class SVM, depending on how they classify ground truth in the validation data.

Park et al. applied the BEAM (Beacon-based Exception Analysis for Multi-Missions) system to anomaly detection in SSME data [30]. BEAM has nine components that use nine different approaches to anomaly detection. The work reported in the referenced paper only used one of the nine components: The Dynamical Invariant Anomaly Detector (DIAD). DIAD is an unsupervised anomaly detection algorithm, like the algorithms described in this article. DIAD differs from the algorithms described in this article in that DIAD only considers one variable at a time, whereas the algorithms described in this article all consider all of the variables together and look for anomalies in the relationships among the variables, in addition to anomalies in individual variables. Park et al. trained DIAD using data from 16 nominal tests, and tested it using data from seven tests that contained known failures. It detected all of the major failures in these seven tests, although it missed some minor failures and had some false alarms.

Iverson describes the IMS and its use in another Space Shuttle application [19]. After the STS-107 Space Shuttle Columbia disaster, Iverson applied IMS to data from four temperature sensors inside the Shuttle's wings. He trained it using data from five previous Space Shuttle flights, and then tested it using STS-107 data. It detected an anomaly in data from the temperature sensors on the Shuttle's left wing shortly after the foam impact, suggesting in retrospect that with the aid of IMS, flight controllers might have been able to detect the damage to the wing much sooner than they did.

Keogh et al. describe an algorithm for finding time series *discords*, which are unusual subsequences of time series [31]. Their algorithm, called HOT SAX (heuristically ordered time series using symbolic aggregate approximation), differs from the algorithms used in this article in that it finds subsequences of the time series that are unusual, whereas the algorithms we used find single points in time that are unusual. HOT SAX also differs from the algorithms we used in that it looks at one variable at a time, whereas the algorithms we used are multivariate, and look for anomalous combinations of values of different variables. The authors applied HOT SAX to 82 different data sets, including one from a Space Shuttle Marotta Valve. Chan and Mahoney applied three other algorithms to Space Shuttle Marotta Valve data [32]. All three of these algorithms are similar to HOT SAX in that they find anomalous subsequences in univariate time series, but they differ in that they use multiple time series as training data.

Fujimaki et al. describe an innovative unsupervised anomaly detection algorithm that learns causal relationships among the variables, and then signals an anomaly when these causal relationships are violated [33]. Learning these causal relationships is another way to explicitly make use of time. They apply this algorithm to data from a simulation of an unmanned orbital transfer vehicle during the process of a rendezvous maneuver with the International Space Station, and demonstrate that their algorithm can detect three simulated anomalies.

Budalakoti et al. describe a system that uses an unsupervised approach to detect anomalies in sequences of discrete data [34]. The system uses envelope detection methods and dynamic Hidden Markov Models, and has been applied to data from the switches in an aircraft cockpit. In this application, it detects anomalies in the sequence of switch flips made by the pilot, including detecting if the switches are flipped in an unusual order. For example, it can detect if the pilot lowers and raises the landing gear multiple times, instead of just once (as is usually the case in the training data).

We have recently used a supervised learning approach to automatically detect and diagnose failures in the new J-2X rocket engine that will be used in the Ares I and Ares V launch vehicles [35]. Because the J-2X has not been built yet, no real sensor data are available. However, we do have access to a high-fidelity simulator that can simulate fault modes. The availability of this simulator enabled us to simulate a large number of faults, and to use these simulated faults to train a C4.5 [36] decision tree to detect the faults and to identify the fault mode. We demonstrated that the resulting decision tree had a very low false alarm rate and low missed detection and misisolation rates.

Many of the existing supervised approaches to data-driven systems health monitoring have used artificial neural networks to model the system. Artificial neural networks are a type of nonlinear model based loosely on the neural structure of the brain, in which the weights of the connections among neurons are automatically adjusted to maximize the fit of the model to the training data [37]. Guo and Musgrave applied neural networks to sensor validation for the SSME [38]. He and Shi found that SVMs produced better accuracy than artificial neural networks when applied to a pump diagnosis problem [39]. One disadvantage of neural network approaches is that most humans are unable to understand or interpret the neural network models. SVMs suffer from similar lack of comprehensibility, but models based on decision trees, decision rules, or nearest neighbors are generally easier to understand, and therefore more likely to be accepted by human experts.

VII. Conclusion

This article presents four sensor failures, one system failure, and four other anomalies that were detected by applying four unsupervised anomaly detection algorithms to data from the Space Shuttle main engine. Although all of these anomalies were either previously known or minor, we believe that they demonstrate that the algorithms have the ability to detect the kind of unusual phenomena in the data that correspond to significant anomalies. We have also demonstrated that these algorithms have the potential to process real rocket propulsion sensor data in real time.

An important point to make is that although some anomalies were detected by multiple algorithms, other anomalies were only detected by one algorithm out of the four. The reason for the difference is that the different algorithms use different definitions of an anomaly. Because of these differences, it can be useful to run multiple anomaly detection algorithms on a data set. In deciding which algorithms to use for a particular data set, one important consideration is the ability of the algorithm to handle discrete data and the presence and importance of discrete data in the data set. Orca and GritBot can handle discrete data explicitly, but IMS and one-class SVMs cannot. A second consideration is the ability of the algorithms to explain the anomalies. Orca, IMS, and GritBot all provide some sort of explanation of each anomaly in terms of the variables, but the one-class SVM algorithm does not (although it might be possible to add such a capability to it). A third consideration is that while some algorithms (such as IMS) learn a model that is guaranteed to cover all of the training data, others (including Orca, GritBot, and one-class SVMs) learn a model that may only cover the vast majority of the training data. If it is known that all of the training data is guaranteed to be nominal, then it may be better to use an algorithm such as IMS that assumes that all of the training data is nominal. However, if there is the possibility that the training data contain a small number of anomalies, then it may be better to use an algorithm that produces a model that can be used to find the anomalies in the training data. A final consideration, as mentioned in Sect. VI, is that some algorithms are easier to understand than other algorithms, and therefore more likely to be accepted by experts. Orca and IMS are relatively simple algorithms that are easy to understand, while the one-class SVM algorithm is more sophisticated and therefore more difficult to understand, and GritBot falls somewhere in between.

VIII. Future Work

This article presents early results; there are many directions for future research. The results presented in this article consist of nine examples of anomalies that were detected by four unsupervised anomaly detection algorithms. Unfortunately, we will probably never have enough data to calculate false positive and false negative error rates with any statistical significance, but testing the algorithms on a larger amount of data than we have used so far should shed additional light on the performance of the algorithms.

All four algorithms described in this article output anomaly scores that measure the degree of anomalousness of the data. Before the algorithms could be used in an automated fashion, alarm thresholds would need to be determined so as to minimize false positive and false negative error rates. Setting these thresholds is difficult because of the small number of available examples of verified anomalies.

In the research described in this article, we used the four algorithms individually and compared the results. As we note in the conclusion, different algorithms within the set of four did a better job of detecting different anomalies. An important direction for future research is the automatic combination of the outputs of the different algorithms in a way that produces better results than any of the algorithms individually.

Because of the small number of examples of anomalies available to us, we decided to first try unsupervised anomaly detection algorithms. We believe that the small number of examples of known anomalies will present a challenge for supervised anomaly detection algorithms. A direction for future research is to try to find ways to make supervised learning algorithms perform adequately given such a small number of examples of anomalies. One approach would be to use each time step from an anomaly that spans multiple time steps as an example of an anomaly. Another approach is the use of semi-supervised learning algorithms — those that are able to use a large amount of unlabeled data as hints to help in separating the small amount of available labeled data [40].

In the results presented in this article, the anomaly detection algorithms did not make use of any expert knowledge. We plan to explore ways of using background knowledge within the automated anomaly detection context. One possible source of knowledge is the determination made by the domain experts that certain anomalies detected by the algorithms are not significant. One way to make use of this knowledge would be to use semi-supervised learning algorithms, and to provide them with examples that are labeled as nominal for each candidate anomaly that the experts judged to be nominal. The algorithm would then avoid incorrectly signaling an anomaly in the future when similar patterns appear in the data.

The data sets used in this research are all time series. The algorithms described in this article, however, do not explicitly make use of time. Instead, time is either ignored or treated just like any other variable, and the algorithms treat each single point in time as a data point. (We do use time to a very limited extent when computing the first and second derivatives of the continuous variables and including them as additional variables.) We plan to try to detect anomalous subsequences in the data by using the same four algorithms together with a technique known as windowing [41]. In this technique, each data point provided to the algorithm includes all of the sensor and command data from a subsequence of the time steps, rather than from a single time step. The length of the subsequence is typically fixed, and a new subsequence is typically started at a fixed interval, resulting in a "sliding window" across the time series.

We also plan to explore algorithms that explicitly model time, such as Kalman filters and Hidden Markov Models. We also plan to try to find a way to extend HOT SAX [31] to find anomalous subsequences in multivariate data.

Acknowledgments

This work would not have been possible without the collaboration of our domain experts. For providing SSME data and expertise, we thank Matt Davidson, Al Daumann, and John Stephens of Pratt & Whitney Rocketdyne, and Anthony Kelley and John Butas of NASA Marshall Space Flight Center. For providing rocket engine test stand data and expertise, we thank Curtis Olive, Randy Holland, Mark Hughes, and Fernando Figueroa of NASA Stennis Space Center.

We also thank the other members of the liquid propulsion health management team at NASA Ames Research Center: Ashok Srivastava, Rodney Martin, and Richard Watson. We thank David Iverson of NASA Ames Research Center for providing the IMS software.

The research described in this article was funded by the NASA Exploration Systems Mission Directorate (ESMD) under the Technology Maturation Program as part of the ISHM Testbeds and Prototypes Project, by ESMD's Advanced Space Technology Program as part of the Collaborative Decision Systems Project, and by ESMD's Exploration Technology Development Program as part of the Integrated Systems Health Management Project.

References

[1] Williams, B. C., and Nayak, P. P., "A Model-based Approach to Reactive Self-Configuring Systems," *Proceedings of the National Conference on Artificial Intelligence,* American Association for Artificial Intelligence, Menlo Park, CA, 1996.

[2] Kurien, J., and Nayak, P. P., "Back to the Future for Consistency-based Trajectory Tracking," *Proceedings of the National Conference on Artificial Intelligence,* American Association for Artificial Intelligence, Menlo Park, CA, 2000.

[3] Narasimhan, S., and Brownston, L., "HyDE — A General Framework for Stochastic and Hybrid Model-based Diagnosis," *18th International Workshop on Principles of Diagnosis (DX-07)*, 2007.

[4] Williams, B. C., Ingham, M. D., Chung, S., Elliott, P., Hofbaur, M., and Sullivan, G. T., "Model-based Programming of Fault-Aware Systems," *AI Magazine*, Vol. 24, No. 4, 2003, pp. 61–75.

[5] TEAMS-RT Web page, http://www.teamqsi.com/RT.html [retrieved 26 April 2007].

[6] RODON Web page, http://www.uptimeworld.com/Rodon.aspx [retrieved 19 May 2009].

[7] Colgren, R., Abbott, R., Schaefer, P., Park, H., Mackey, R., James, M., et al., “Technologies for Reliable Autonomous Control (TRAC) of UAVs,” *19th Digital Avionics Systems Conference*, October 2000.

[8] Barrett, A., “Model Compilation for Real-Time Planning and Diagnosis with Feedback,” *Proceedings on the International Joint Conference on Artificial Intelligence*, 2005.

[9] Schwabacher, M., “Machine Learning for Rocket Propulsion Health Monitoring,” *Proceedings of the SAE World Aerospace Congress*, Vol. 114-1, Society of Automotive Engineers, Warrendale, PA, 2005.

[10] Schwabacher, M., Oza, N., and Matthews, B., “Unsupervised Anomaly Detection for Liquid-Fueled Rocket Propulsion Health Monitoring,” *AIAA Infotech@Aerospace Conference*, American Institute of Aeronautics and Astronautics, Washington, DC, 2007.

[11] Chandola, V., Banerjee, A., and Kumar, V., “Anomaly Detection: A Survey,” (to appear), *ACM Computing Surveys*, 2009.

[12] Markou, M., and Singh, S., “Novelty Detection: A Review. Part 1: Statistical Approaches,” *Signal Processing*, Vol. 83, 2003, pp. 2481–2497.
doi: 10.1016/j.sigpro.2003.07.018

[13] Markou, M., and Singh, S., “Novelty Detection: A Review. Part 2: Neural Network Based Approaches,” *Signal Processing*, Vol. 83, 2003, pp. 2499–2521.
doi: 10.1016/j.sigpro.2003.07.019

[14] Main Propulsion System. NASA Kennedy Space Center, http://science.ksc.nasa.gov/shuttle/technology/sts-newsref/sts-mps.html [retrieved 26 April 2007].

[15] Constellation Program: America’s Fleet of Next-Generation Launch Vehicles — The Ares I Crew Launch Vehicle. NASA Fact Sheet FS-2006-07-85-MSFC, 2006, http://www.nasa.gov/pdf/151419main_aresI_factsheet.pdf [retrieved 26 April 2007].

[16] Constellation Program: America’s Fleet of Next-Generation Launch Vehicles — The Ares V Cargo Launch Vehicle. NASA Fact Sheet FS-2006-07-84-MSFC, 2006, http://www.nasa.gov/pdf/151420main_aresV_factsheet.pdf [retrieved 26 April 2007].

[17] Bay, S. D., and Schwabacher, M., “Mining Distance-Based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule,” *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, New York, NY, 2003.

[18] GritBot. RuleQuest Research, http://www.rulequest.com [retrieved 26 April 2007].

[19] Iverson, D. L., “Inductive System Health Monitoring,” *Proceedings of the International Conference on Artificial Intelligence, IC-AI ’04*, Vol. 2 & *Proceedings of the International Conference on Machine Learning; Models, Technologies & Applications, MLMTA ’04*, Computer Science Research, Education, and Applications (CSREA) Press, Athens, GA, June 21–24, 2004.

[20] Iverson, D., Martin, R., Schwabacher, M., Spirkovska, L, Mackey, R., Castle, P., et al., “General Purpose Data- Driven System Monitoring for Space Operations,” *AIAA Infotech@Aerospace Conference*, AIAA, Washington, DC, 2009, AIAA paper 2009-1909.

[21] Tax, D. M. J., and Duin, R. P. W., “Support Vector Domain Description,” *Pattern Recognition Letters*, Vol. 20, No. 1113, 1999, pp. 1191–1199.

[22] Statistics Toolbox. The MathWorks Web site, http://www.mathworks.com/products/statistics/ [retrieved 23 March 2009].

[23] Scholkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C., “Estimating the Support of a High-Dimensional Distribution”, *Neural Computation*, Vol. 13, No. 7, 2001, pp. 1443–1471.
doi: 10.1162/089976601750264965

[24] OSU SVM for MATLAB Web site, http://svm.sourceforge.net/download.shtml [retrieved 30 November 2007].

[25] Chang, C., and Lin, C., “LIBSVM: a Library for Support Vector Machines,” http://www.csie.ntu.edu.tw/∼cjlin/libsvm/ [retrieved June 27, 2007].

[26] Runarsson R. T., Unnthorsson, R., and Johnson, T. M., “Model Selection in One Class Nu-SVMS using RBF Kernels,” *16th Conference on Condition Monitoring and Diagnostic Engineering Management*, 2003.

[27] E-1 Test Facility. NASA Stennis Space Center Web site http://sscfreedom.ssc.nasa.gov/esd/ESDTestFacilitiesE1.asp [retrieved 15 June 2007].

[28] Fiorucci, T. R., Larkin II, D. R., and Reynolds, T. D., “Advanced Engine Health Management Applications of The SSME Real-Time Vibration Monitoring System,” *36th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, AIAA, Washington, DC, 2000, AIAA paper 2000-3622.

[29] Martin, R. A., Schwabacher, M., Oza, N., and Srivastava, A., “Comparison of Unsupervised Anomaly Detection Methods for Systems Health Management Using Space Shuttle Main Engine Data,” *Proceedings of the Joint Army Navy NASA Air Force Conference on Propulsion*, Joint Army-Navy-NASA-Air Force (JANNAF) Interagency Propulsion Committee, Baltimore, MD, 2007.

[30] Park, H., Mackey, R., James, M., Zak, M., Kynard, M., Sebghati, J., et al., "Analysis of Space Shuttle Main Engine Data Using Beacon-based Exception Analysis for Multi-Missions," *Aerospace Conference Proceedings*, Vol. 6, IEEE, Washington, DC, 2002, pp. 6-2835–6-2844.
doi: 10.1109/AERO.2002.1036123

[31] Keogh, E., Lin, J., and Fu, A., "HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence," *Proceedings of the 5th IEEE International Conference on Data Mining*, IEEE Computer Society, Washington, DC, 2005, pp. 226–233.

[32] Chan, P. K., and Mahoney, M. V., "Modeling Multiple Time Series for Anomaly Detection," *Proceedings of the Fifth IEEE International Conference on Data Mining*, IEEE Computer Society, Washington, DC, 2005, pp. 90–97.

[33] Fujimaki, R., Yairi, T., and Machida, K., "An Approach to Spacecraft Anomaly Detection Problem Using Kernel Feature Space," *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery (ACM), New York, NY, 2005, pp. 401–410.

[34] Budalakoti, S., Srivastava, A. N., and Otey, M. E., "Anomaly Detection and Diagnosis Algorithms for Discrete Symbol Sequences with Applications to Airline Safety," *IEEE Transactions on Systems, Man, and Cybernetics, Part C,* Vol. 39, No. 1, 2009, pp. 101–113.
doi: 10.1109/TSMCC.2008.2007248

[35] Schwabacher, M., Aguilar, R., and Figueroa, F., "Using Decision Trees to Detect and Isolate Simulated Leaks in the J-2X Rocket Engine," *IEEE Aerospace Conference*, IEEE, New York, NY, 2009.

[36] Quinlan, J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993.

[37] Bishop, C. M., *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.

[38] Guo, T.-H., and Musgrave, J., "Neural Network Based Sensor Validation for Reusable Rocket Engines," *Proceedings of the American Control Conference*, Vol. 2, American Automatic Control Council, Dayton, OH, 1995, pp. 1367–1372.

[39] He, F., and Shi, W., "WPT-SVMs Based Approach for Fault Detection of Valves in Reciprocating Pumps," *Proceedings of the American Control Conference*, American Automatic Control Council (AACC), Dayton, OH, 2002, pp. 4566–4570.

[40] Belkin, M., Niyogi, P., and Sindhwani, V., "Manifold Regularization: A Geometric Framework for Learning from Examples," Technical Report TR-2004-06, University of Chicago, Department of Computer Science, 2004. http://www.cs.uchicago.edu/research/publications/techreports/TR-2004-06, [retrieved 27 June 2007].

[41] Das, G., Lin, K. I., Mannila, H., and Renganathan, G., "Rule discovery from Time Series," *Proceedings of the 3rd International Conference of Knowledge Discovery and Data Mining*, Association for Computing Machinery (ACM), New York, NY, 1998, pp. 16–22.

Christopher Rouff
Associate Editor