8

Evaluation of Anomaly Detection Capability for Ground-Based Pre-Launch Shuttle Operations

Rodney A. Martin, Ph.D.
NASA Ames Research Center
Mail Stop 269-1
Moffett Field, CA 94035
U.S.A.

1. Introduction

This chapter will provide a thorough end-to-end description of the process for evaluation of three different data-driven algorithms for anomaly detection to select the best candidate for deployment as part of a suite of IVHM (Integrated Vehicle Health Management) technologies. These algorithms will be evaluated based upon their capability for robustly detecting incipient faults or failures in the ground-based phase of pre-launch space shuttle operations, rather than based on system certifiability as performed in previous studies (Schwabacher & Waterman, 2008). Robust detection will allow for the achievement of pre-specified minimum false alarm and/or missed detection rates in the selection of alert thresholds. All algorithms will be optimized with respect to an aggregation of these same criteria. The final results will also include a formal cross-validation procedure, which will be used to perform optimization and alert threshold selection.

The data-driven algorithms to be evaluated in this study were deemed to be sufficiently mature for consideration as viable candidates for demonstration during the launch of Ares I-X. This launch represents the first test flight of Ares I, which will be the successor to the Space Shuttle for NASA's Constellation program. Our study relies upon the use of Shuttle data to act as a proxy for and in preparation for application to Ares I-X data, which uses a very similar hardware platform for the subsystems that are being targeted (TVC – Thrust Vector Control subsystem for the SRB (Solid Rocket Booster)).

Data-driven algorithms are just one of three different types being deployed, the details of which were presented in previous work (Iverson *et al.*, 2009); (Schwabacher & Waterman, 2008). The other two types of algorithms being deployed include a "rule-based" expert system, and a "model-based" system. Within these two categories, the deployable candidates have already been selected based upon non-quantitative factors such as flight heritage and system certifiability. For the rule-based system, SHINE (Spacecraft Health Inference Engine) has been selected for deployment, which is one of many components of BEAM (Beacon-based Exception Analysis for Multimissions) as described in its debut article (Mackey *et al.*, 2001). Other components of BEAM include various data-driven algorithms. BEAM is a patented technology developed at NASA's JPL (Jet Propulsion Laboratory) and

Source: Aerospace Technologies Advancements, Book edited by: Dr. Thawar T. Arif,
ISBN 978-953-7619-96-1, pp. 492, January 2010, INTECH, Croatia, downloaded from SCIYO.COM

SHINE serves to aid in the management and identification of operational modes. For the "model-based" system, a commercially available package developed by QSI (Qualtech Systems, Inc.), TEAMS (Testability Engineering and Maintenance System) was highlighted in work subsequent to its debut (Cavanaugh, 2001), and has been selected for deployment to aid in diagnosis. In the context of this particular deployment, distinctions among the use of the terms "data-driven," "rule-based," and "model-based," can be found in the previously cited paper by Schwabacher and Waterman.

Although there are three different categories of algorithms that have been selected for deployment, our main focus in this chapter will be on the evaluation of three candidates for *data-driven* anomaly detection. The main thrust of the chapter is to provide instructive coverage on the topics of algorithmic optimization and alert threshold selection for the candidate data-driven algorithms. These algorithms will be optimized and compared using the AUC (Area under the ROC (Receiver Operating Characteristic) curve), which represents overall classification discriminability. The resulting optimized algorithmic parameters can then be used to perform threshold or alert selection based upon the corresponding ROC curve. This allows for the demonstration of a robust anomaly detection capability when performing alert threshold selection, which is based upon specified minimum false alarm and/or missed detection rates. As a practical measure we will also present a performance comparison among the candidate data-driven algorithms by evaluation of their computational complexity as well as the AUC. The final results will also include a formal cross-validation procedure, which will be used to provide uncertainty bounds for the optimization and alert threshold selection processes.

2. Motivation & background

Some insightful previous studies (Fragola, 1996); (Paté-Cornell and Dillon, 2001) discuss statistical analyses concerning general Space Shuttle risk management for the pre-Columbia era, and detail both lapses and advancements in the use of probabilistic risk analyses (PRA). We note here that probabilistic risk analysis covers the quantification of failure probabilities in the absence of large volumes of data for supporting analyses, but is not directly related to the post-design real-time monitoring phase for the detection of anomalous events. The primary author of the previously cited study (Paté-Cornell & Dillon, 2001) has long been considered a respected expert in the PRA field, and is often sought by NASA for opinions, judgment, and analyses on such issues. In a study by this same author (Paté-Cornell and Fischbeck, 1994) on the risk analyses of tiles on the space shuttle which predated the Columbia tragedy by almost a decade, it was found that the use of PRA admitted a conclusion that almost foretold the events of the eventual demise of the Space Shuttle Columbia.

Noted for the advocacy on the use of Bayesian statistics as opposed to classical frequentist statistics within PRA, it is often shown in Paté-Cornell's studies that a quantitative model-based formulation requires data gathered from a variety of sources. In the case of frequentist statistics, this requires a sufficient volume of data in order to generate a reasonably accurate estimate of risk and confidence. The use of Bayesian analysis is more apropos to the situation of developing risk estimates for novel engineered systems such as the Space Shuttle, and even more so for Ares I-X where there is an inherent scarcity of data. As such, there is the need to develop a well-informed prior distribution.

Furthermore, Paté-Cornell strongly cautions engineers against the use of a mixed approach involving both Bayesian and frequentist statistics in order to provide for consistency, and

possibly due to previous lessons learned from experience with the application of conservative risk estimates in tandem with probabilities. Although a pure Bayesian approach is not used in our study, nor we do not concern ourselves with the computation and quantification of risks and potential failures for PRA, we are motivated by another of Paté-Cornell's cautions that all too often qualitative measures are used for decision making rather than quantitative ones.

We do not present these ideas as a preamble for the investigation of the development of such PRA's in this chapter, but rather for the monitoring of systems ostensibly that PRA served to inform at the design and post-design certification stages. However, areas where these previous studies may relate to ours are in the establishment of reasonably accurate failure probabilities to inform evaluation of monitoring algorithms, and the use of a model-based approach in tandem with a data-driven approach for monitoring. A good overview of the need to establish a formal relationship between design and monitoring for space applications has been discussed thoroughly in previous work (Tumer, 2005).

We note that PRA relates to design and risk assessment, whereas the use of data driven statistical analyses performed in this study assumes that appropriate measures have already been taken in advance in order to compute and quantify probabilities of failure. As such, complementary maximum allowable probabilities of false alarm and missed detection are assumed to have already been established and available. These two metrics are the essence of the analysis we present in this chapter.

We also note as described previously that a model-based approach (TEAMS) is used in the same architecture as a data-driven approach. This is still not applicable nor indicative of an attempt to develop a PRA. The framework presented here relates to the monitoring of the designed systems rather than the assessment of risk. There is evidence that the use of a Bayesian approach has been embedded within this model-based tool (Tu *et al.*, 2006), although these aspects of the system are not currently being used for monitoring in this particular demonstration or deployment. However, the availability of such functionality is a general step in the right direction nonetheless. Paté-Cornell alludes to the use of a Bayesian approach as being of paramount importance for inclusion in PRA, but it arguably has similar merit for monitoring. Even though the full Bayesian updating aspects of TEAMS are not currently being deployed for monitoring purposes, the option to include MTTF (Mean-Time-To-Failure) estimates based upon the results of PRA exists in TEAMS-RT, the real-time counterpart to our chosen model-based approach. These estimates can be used to calculate the probability of each suspect failure mode.

Our primary concern in this chapter, however, is with an optimization, evaluation, and performance assessment of candidate data-driven monitoring algorithms for the relevant platform. As such, we note that there have been other related studies *e.g.* (Hart, 1990) on the statistical analyses of data related to the exact subsystem that we are considering here. However, we have found very little indication that the results of these studies were adopted and implemented in any fashion. The mathematical sophistication of these studies was most likely well beyond the capability of practical implementation at the time, as they were performed nearly two decades ago. However, advancements in the use of ROC analysis that have recently gained traction or application in the aerospace IVHM community have motivated its current use for space-based propulsion systems of the exact type analyzed in previous work of the current author (Martin, 2007); (Martin *et al.*, 2007). Furthermore, ROC analysis is a much more straightforward and broadly applicable tool used for statistical analysis than in the previously cited work (Hart, 1990), and more useful for the design of alarm and monitoring systems.

3. Approach

Ostensibly, operating procedures for the thrust vector control (TVC) subsystem on Ares I-X used to support gimbaling of the solid rocket booster are also patterned after operations of the Space Shuttle. As such, for the purposes of our analysis here we consider two primary phases of operation of the TVC subsystem based upon Space Shuttle operations. Testing of the TVC in the VAB (Vehicle Assembly Building) often occurs anywhere from several weeks to a few months prior to the point at which the vehicle is fully assembled and ready to be rolled out on the MLP (Mobile Launch Platform) to the pad for launch. Launch countdown begins while the assembled vehicle is already rolled out to the launch platform; however, occasionally certain tests that normally occur in the VAB are also performed at the pad.

To simplify the distinction between the two main phases of operation, we will only consider tests of the TVC subsystem when the vehicle is in the VAB, and when the fully assembled vehicle is at the pad after launch countdown has commenced. Data from previous Space Shuttle launches has been gathered from both phases, and as such both will be considered viable candidates for analysis using the methods described previously. The availability of the data during each phase and its ability to meet certain requirements will determine the choice of a particular phase of operation as the primary test phase for evaluation of data-driven algorithms. Specifically, the requirement to be met is the availability of a sufficient amount of data to undergo a formal cross-validation procedure that can be processed in a reasonable amount of time. *A priori* knowledge on computational complexity of each data-driven algorithm will aid in determining if they can be processed in a reasonable amount of time, given a fixed volume of data that sufficiently characterizes nominal behavior spanning multiple launches.

Finally, all data must be pre-processed and transformed into a standard format fit for utilization of all data-driven algorithms. This pre-processing can be decomposed into two basic steps. The first step requires resampling of data that is recorded only when parameter values change or that are derived from different data sources. This resampling technique is performed by holding the last known value, resulting in a new data file where each row corresponds to a single time step, and each column corresponds to a distinct parameter, regardless of its origin. Thus, the data is merged into one aggregate "matrix" representation. The second step necessitates normalization of all continuous-valued parameters so that any potential algorithmic biases resulting from parameters that have drastically different ranges are effectively mitigated. If any discrete-valued parameters or continuous parameters that have fixed values exist in the dataset, they will bypass this normalization step in order to prevent potential singularities in algorithmic processing. For the purposes of this chapter, unless otherwise stated it will be assumed henceforth that all data has been processed and transformed into this standard format.

3.1 IMS (Inductive Monitoring System)

The Inductive Monitoring System (IMS) is a distance-based anomaly detection tool that uses a data driven technique called clustering to extract models of normal system operation from archived data. IMS works with vectors of data values as described in the previous section. During the learning process, IMS analyzes data collected during periods of normal system operation to build a system model. It characterizes how the parameters relate to one another during normal operation by finding areas in the vector space where nominal data tends to fall. These areas are called nominal operating regions and correspond to clusters of nearby,

similar points found by the IMS clustering algorithm. IMS represents these nominal operating regions as hyper-boxes in the vector space, providing a minimum and maximum value limit for each parameter of a vector contained in a particular hyper-box. These hyper-box cluster specifications are stored in a knowledge base that IMS uses for real-time telemetry monitoring or archived data analysis. Figure 1a shows an overview of the IMS method.

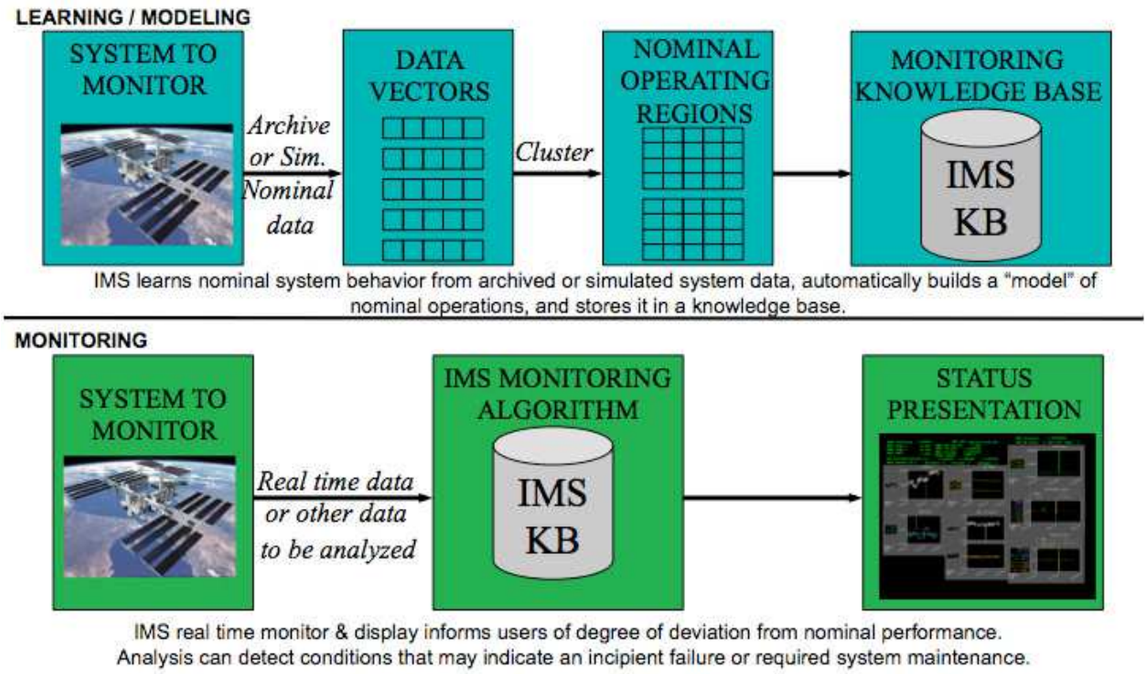

Fig. 1a. Inductive Monitoring System (IMS) Overview

3.1.1 IMS learning process

In general, the number and extent of nominal operating regions created during the IMS learning process is determined by three learning parameters: the "maximum cluster radius" can be used to adjust the size and number of clusters derived from a fixed number of training data points, the "initial cluster size" is used to adjust the tolerance of newly created nominal operating regions, and the "cluster growth percent" is used to adjust the percent increase in size of a nominal operating region when incorporating new training data vectors. More specifically, the learning algorithm builds a knowledge base of clusters from successively processed vectors of training data. As such, the clustering approach is incremental in nature, which distinguishes it from well-known methods such as k-means clustering where the resulting clusters are independent of the ordering of the vectors. With the processing of each new training data vector, the distance from this new vector to the centroid of the nearest cluster in the knowledge base is computed. If this distance is below a pre-specified value, the "maximum cluster radius," the new vector is summarily incorporated into that cluster. The upper or lower limits for each affected dimension of the cluster are expanded respectively according to the "cluster growth percent" parameter to reflect the inclusion of the new vector. This incremental, inductive process gives IMS an advantage over other clustering methods such as k-means, since it tends to group temporally related points during the learning process. The grouping of temporally related

points may also aid in discovering distinct system operations, which makes IMS more amenable to the specific goal of monitoring time series data for system operations.
The "cluster growth percent" parameter is used to adjust the learning rate. It establishes a fixed "growth" percentage difference for expansion of each dimension when updating previously formed clusters. This "cluster growth percent" learning parameter is therefore clearly proportional to the learning rate, due to the increased number of training data points that will be assigned to each new cluster per iteration for higher values of the "cluster growth percent" parameter. Naturally, the number of clusters in the knowledge base for a given training data set will increase as the "maximum cluster radius" and "cluster growth percent" values are decreased. Therefore, an inverse relationship between the maximum cluster radius and the number of clusters in the knowledge base exists. This dependence can be exploited to regulate the final size of the knowledge base in order to accommodate resource limitations in the computers running IMS, and to optimize selected metrics.
If the distance between a newly processed vector and the centroid of the nearest cluster in the knowledge base is above the pre-specified "maximum cluster radius" value, a new cluster is created. The formation of a new cluster is accomplished by creating a hyper-box whose dimensions are based upon forming a window around each element of the new training data vector. The window is defined by introducing the "initial cluster size" parameter which is used to adjust the learning tolerance. This "initial cluster size" learning parameter represents a fixed percentage of the value for each dimension of the new training vector and as such relates directly to the size of newly established clusters. The "initial cluster size" and "cluster growth percent" learning parameters also act as buffers which enable a provisional allowance for manufacturing sensor tolerances and for sensors that may have suffered from deterioration due to wear. Furthermore, these learning parameters provide increased coverage to compensate for training data that may not fully characterize the nominal performance envelope.

3.1.2 IMS monitoring process

During the monitoring operation, IMS reads and normalizes real-time or archived data values, formats them into the predefined vector structure, and searches the knowledge base of nominal operating regions to see how well the new data vector fits the nominal system characterization. After each search, IMS returns the distance from the new vector to the nearest nominal operating region, called the composite distance. Data that matches the normal training data well will have a composite distance of zero. If one or more of the data parameters is slightly outside of expected values, a small non-zero result is returned. As incoming data deviates further from the normal system data, indicating a possible malfunction, IMS will return a higher composite distance value to alert users to the anomaly. IMS also calculates the contribution of each individual parameter to the composite deviation, and the index of the nearest cluster to the monitored point, which can help identify and isolate the cause of the anomaly.

3.1.3 IMS score computations

This section describes the computation of the IMS composite score and individual contributing scores. The IMS scores are computed on a monitored data stream, and are functions of distances to the nearest clusters which have been computed from training data. Table 1 lists the notation used in the score calculation equations:

y	Raw input data in physical measurement units
s	Sample index
p	Parameter index
c	Cluster index
n_p	Number of parameters
S	Scale factor for IMS internal computations; as of this writing S = 10,000
k_z	Standard deviation multiplier for normalization constant
$\mathcal{O}$	Offset to make IMS internal numbers positive
$U_{p,c}$	Upper cluster bound for cluster c in dimension p
$L_{p,c}$	Lower cluster bound for cluster c in dimension p

Table 1. Notation

1. During training, data is converted to a multiple k_z of the Z-score over the training data. The mean and standard deviation of each parameter in the training data is computed. The normalized data is then multiplied by a scale factor S and an offset $\mathcal{O}$ is added to make all of the numbers positive.

   The modified Z-score calculation is shown in Eqn. 1.

$$z(y_{p,s}) = \frac{y_{p,s} - \overline{y_p}}{k_z \hat{\sigma}(y_p)} \tag{1}$$

   where $\hat{\sigma}$ is the sample standard deviation. The coordinates of each point, in IMS internal units, is calculated as shown in Eqn. 2.

$$r_{p,s} = z(y_{p,s})S + \mathcal{O} \tag{2}$$

2. The distance component between a monitor point $r_{p,s}$ for parameter p, sample s to cluster c is as shown in Eqn. 3.

$$\Delta_{p,s,c} = \text{dist}(r_{p,s}, c) = \begin{cases} r_{p,s} - U_{p,c}, & r_{p,s} > U_{p,c} \\ r_{p,s} - L_{p,c}, & r_{p,s} < L_{p,c} \\ 0, & L_{p,c} < r_{p,s} < U_{p,c} \end{cases} \tag{3}$$

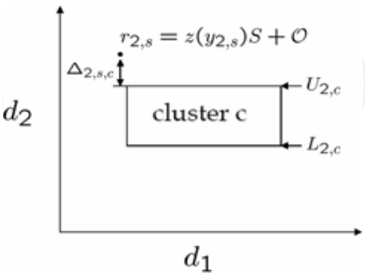

Fig. 1b. Distance from point $r_{2,s}$ to cluster c in dimension p=2

3. The squared Euclidean distance $\beta_{s,c}$ from each sample to each cluster is as shown in Eqn. 4.

$$\beta_{s,c} = \sum_p \Delta^2_{p,s,c} \tag{4}$$

4. The Euclidean distance γ_s to the nearest cluster for each point is as shown in Eqn. 5.

$$\gamma_s = \sqrt{\min_c(\beta_{s,c})} \tag{5}$$

Eqn. 6 provides a formula for the nearest cluster, denoted as $\tilde{c}_s$.

$$\tilde{c}_s = \arg\min_c(\beta_{s,c}) \tag{6}$$

5. The composite score shown in Eqn. 7 is computed in a way which is intended to have a rough correspondance with a percentage of the size of the vector space. This correspondance is closer when the data was normalized over the range of the training data in each dimension. In the IMS implementation where the normalizing constant in each dimension is some multiple of the standard deviation for that dimension, the meaning is less clear.

$$\theta_s = \left(\frac{\gamma_s}{\sqrt{n_p S^2}}\right)100 \tag{7}$$

The factor of $\sqrt{n_p}$ in the denominator of Eqn. 7 normalizes the IMS score across data sets with differing numbers of parameters.

6. The contributing channel score for an individual channel is computed as follows. Let $\tilde{d}(r_{p,s}, \tilde{c}_s) = \text{dist}(r_{p,s}, \tilde{c}_s)$. The contributing score for $\tilde{D}_{s,p}$ is shown in Eqn. 8.

$$\tilde{D}_{s,p} = \left(\frac{\tilde{d}_{p,s}}{S}\right)100 \tag{8}$$

3.1.4 IMS complexity

Figs. 2 and 3 illustrate empirically generated timing tests for IMS runs that demonstrate its computational complexity for datasets of increasing size. Fig. 2 illustrates only the time that it takes to train IMS, while Fig. 3 shows the time that it takes to normalize, train, test, and parse the results for IMS. In both figures, results are shown for data sets that have different numbers of parameters, which aids in gaining a sense of scalability as well as complexity. In Fig. 2, the size of the dataset containing 21 parameters was increased from a minimal size of 338 kB to its full size of 32 MB (blue dashed line). This data set was also noted to have been based upon resampled data, as distinct from "change only" data. This distinction is notable due to the fact that the resampled counterpart to a "change-only" data set is always of a much larger size, and a change only dataset is sufficiently representative for use as a training data set, both for obvious reasons. The resampled version is only necessary when performing monitoring and subsequent analyses.

In Fig. 2, both resampled and change only versions of a dataset containing 164 parameters are also shown. They are varied in size from a minimal ~ 6 MB to 218 MB (just over one-third the full size of the resampled dataset) and from 2.64 MB to 250 MB (the full size of the change only dataset), which are shown with solid blue dots and the solid blue line, respectively. The length of time spanned by the actual dataset (real-time) is shown in green,

and appears nearly as a vertical line due to the scale used for the graph, even though the slope is finite. As such, it is evident that for all timing tests shown in both Figs. 2 and 3 (in blue), IMS runs nearly in linear time and far faster than real-time, even when accounting for the time that it takes to normalize, train, test, and parse the results for IMS (In Fig. 3). It is also evident that the number of parameters in the dataset processed by IMS scales in a multiplicative fashion (as well as the size of the dataset, implicitly, due to aforementioned linear time characterization). However, for the purposes of performing a thorough analysis that involves optimization and an n-fold cross validation, even a linear time complexity needs to be considered carefully.

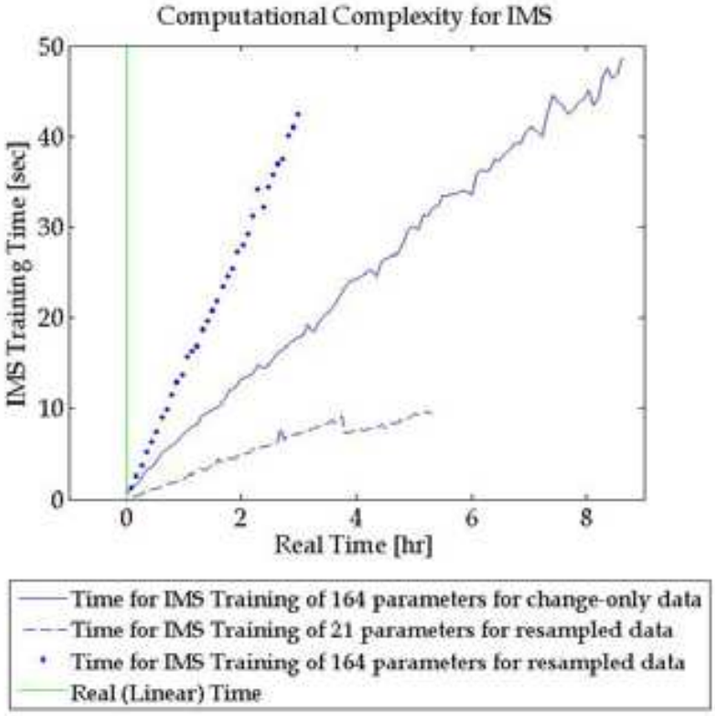

Fig. 2. IMS Training Complexity

To see how quickly time can add up, take as an example a 10-fold cross validation and optimization performed on 5-hr training and validation datasets with 21 parameters. If we base the total processing time for each point within a 150 point optimization grid on Fig. 3, it becomes clear that it would exceed ~ 30 sec x 10 cases X 150 optimization points (>12 hrs), in addition to the time for monitoring auxiliary fault data. If the validation dataset used to monitor auxiliary fault data consists of 2 nominal and 2 fault scenarios, and each scenario spans 5 hrs, then the time for monitoring fault data for 2 nominal and 2 fault scenarios exceeds ~ 4 sec X 10 cases X150 optimization points X 4 scenarios, based upon Fig. 2 (using ½ training time as a rough estimate for monitoring time). The total estimated time amounts to almost 20 hrs. of computational effort. And even so, these rough computations do not account for the fact that for file sizes above the max 250 MB used to create these plots, the number of clusters found by IMS may increase substantially, also adding significantly to the computational cost, possibly running in near real-time or even above real-time for very large files.

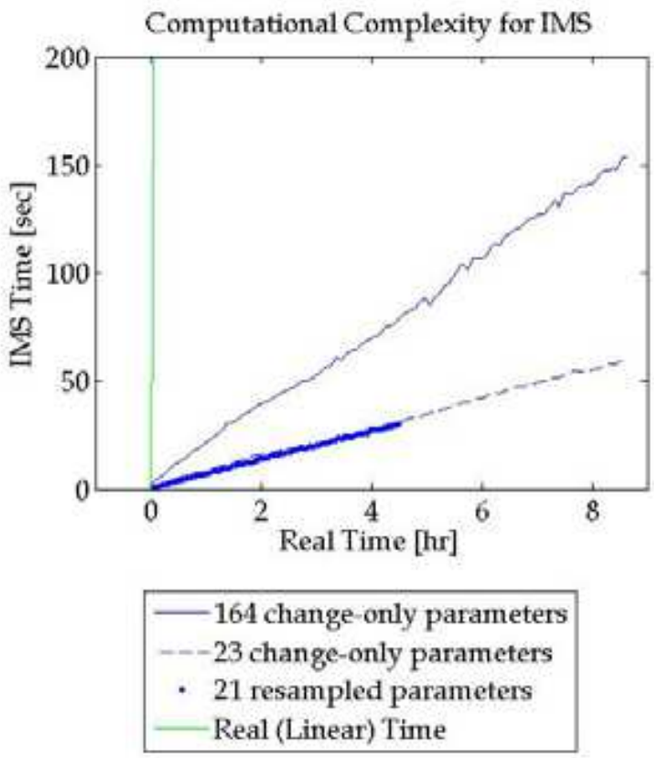

Fig. 3. IMS Normalization, Training, Testing, and Results Parsing Complexity

Evidence of this variation with number of clusters is shown in Fig. 4. Here, training /validation data file sizes are fixed for the entire experiment. The experiment implicitly varies the number of clusters with each subsequent training/validation run via adjustment of IMS's "maximum cluster radius" parameter. The x axis represents the number of sequential IMS training/validation set completions, the y axis on the left represents the date of completion, and the y axis on the right represents the number of clusters. As in Fig. 3, this involves IMS training, testing, and the overhead involved in wrapper code written for generation of statistics for optimization, which includes parsing of the results. The reason sequentially processing training/validation runs appears sublinear is due to the fact that the "maximum cluster radius" value is being adjusted from a very small value to 1. Recall that there is always an inverse relationship between the maximum cluster radius and the number of clusters. As such, we implicitly decrease the number of clusters per IMS knowledge base for each subsequent run from 976 to 18, causing the time for completion to be shortened decrementally.

For files of an increasing size which seeded the experimental results shown in Figs. 2 and 3, the effect seen in Fig. 4 may bias the attempt to control the experiment by holding the maximum cluster radius parameter fixed. When the file size is increased for the experiments yielding the results shown in Figs. 2 and 3, the number of clusters naturally increases due to the greater chance of visiting areas in the hyperspace that contain vectors previously unseen by the current IMS cluster population. As such, a similar sublinear effect as seen in Fig. 4 may result for an experiment in which the file size is decreased but the maximum cluster radius parameter is held fixed. Conversely, a limiting effect is apparent when the number of

clusters increases as a side effect of increasing the file size, implying that IMS will run in near linear time as file size increases (as is evident in Figs. 2 and 3). However, this limiting linear time effect does not imply that IMS will run in *real* time for an arbitrarily large number of clusters. In fact, as mentioned previously, it is possible for IMS to run above real-time for very large files, since we know that IMS scales in a multiplicative fashion with the size of the dataset. As such, an increase in the number of clusters resulting from an increase in file size will yield a substantially greater computational burden. In the previous example, rough estimates were provided that did not account for these effects.

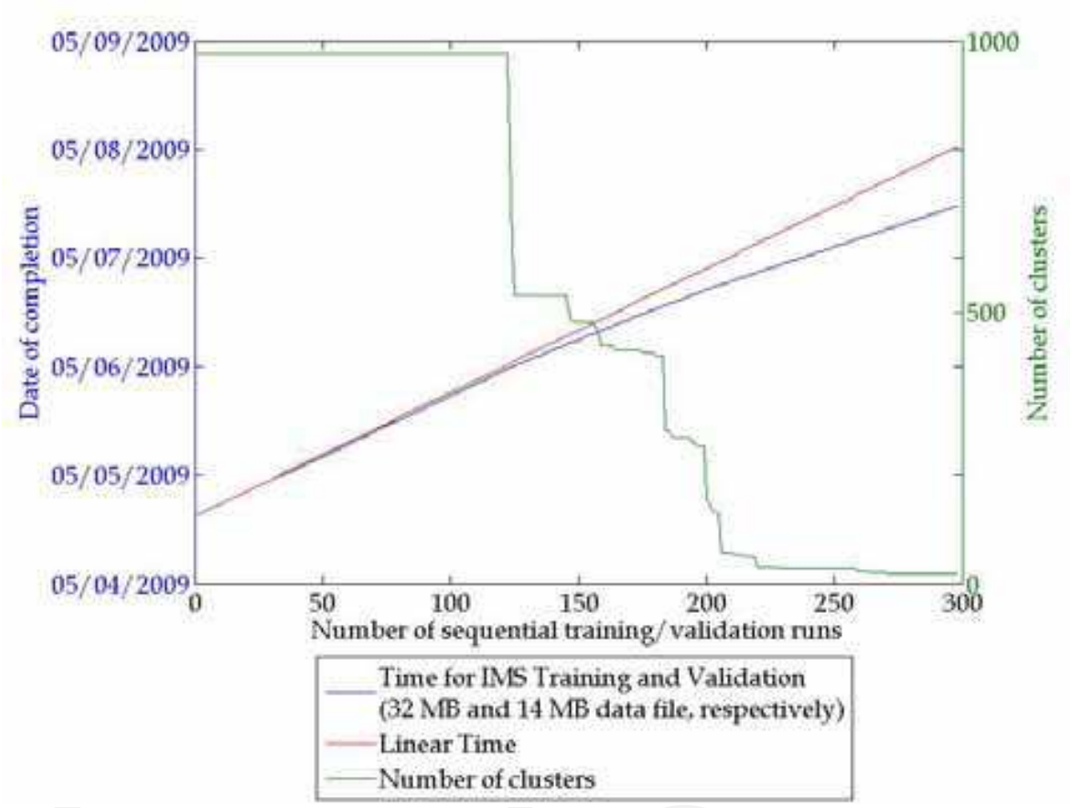

Fig. 4. IMS Complexity and corresponding number of clusters

3.2 Orca

Orca is a software tool that uses a nearest neighbor based approach to outlier detection which is based upon the Euclidean distance metric. It uses a modified pruning rule that allows for increased computational efficiency, running in near linear time as the number of "top score" outlier points selected for evaluation decreases. More information on this algorithm and some of its applications can be found in previous work (Bay and Schwabacher, 2003); (Schwabacher, 2005). This algorithm outputs a total score which represents the average distance to the nearest k neighbors in the multi-dimensional feature space containing all of the variables. It also outputs the contribution of each variable to this score in order to show which variables cause each outlier to be classified as such.

Unlike IMS, Orca requires no training due the inherent comparative nature of the algorithm. Ostensibly, this should reduce the total computational burden. However, the greatest

computational advantages of using Orca are reaped only when applying a modified pruning rule for a reduced number of outlier points based upon having the highest score magnitude selected for evaluation. An implicit requirement to use all available points exists, to aid in the construction of an unbiased ROC curve representative of all score magnitudes and with full resolution. Therefore, Orca loses is computational advantage and defaults to a standard *k*-nn (nearest neighbor) algorithm that scales quadratically. A two-dimensional example is provided in Fig. 5 in order to illustrate the basic idea behind the *k*-nn algorithm. The red circle represents the point being monitored, and the average Euclidean distance to a fixed number of nearest neighbors, *k*, is used to compute a composite anomaly score. Isolation of the outlier point becomes possible by using this technique as the monitored point is indexed across the entire population, as is evident in the diagram.

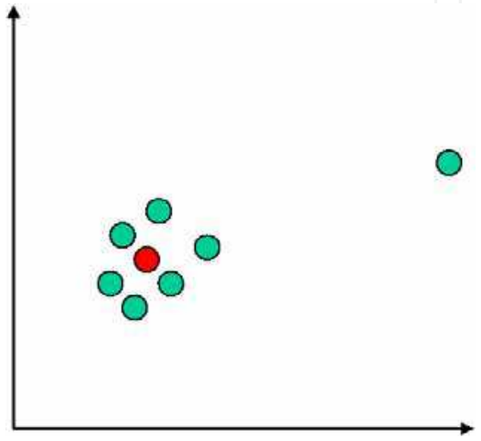

Fig. 5. Nearest Neighbor Approach to Anomaly Detection

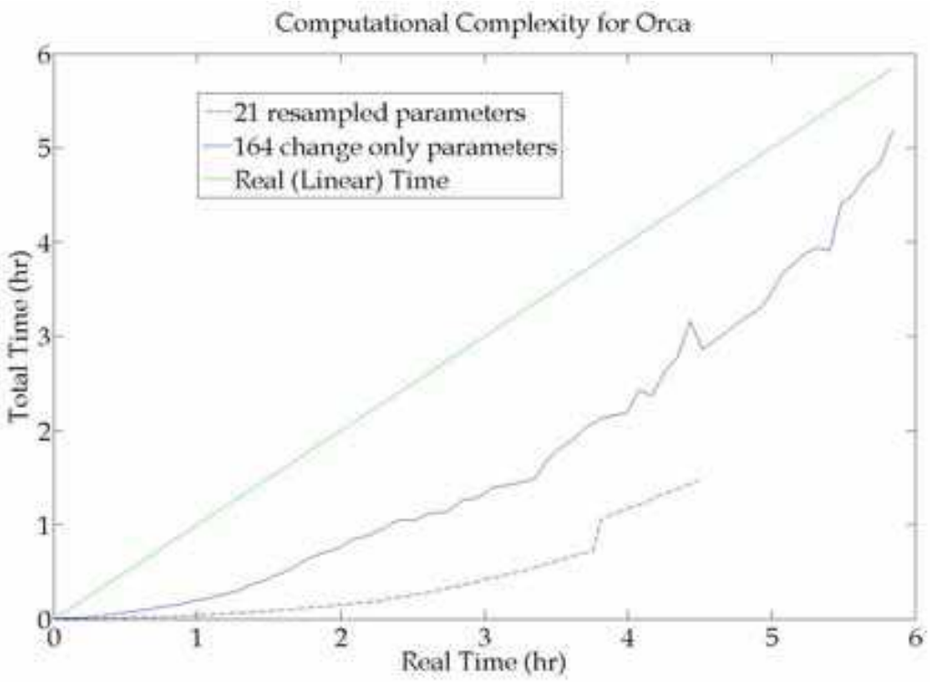

Fig. 6. Orca Normalization, Testing, and Results Parsing Complexity

Empirical evidence of quadratic complexity is shown in Fig. 6, which also illustrates the time required for normalization, application of Orca, and results parsing. This graph merits direct comparison to Fig. 3 showing complexity results for IMS, due to use of the same datasets and ranges in size of the datasets. Clearly, the computational complexity of Orca is far greater than that of using IMS. Furthermore, for datasets spanning a large period of time, it is readily apparent that the algorithm will eventually no longer be able to run faster than real time.

3.3 One-Class Support Vector Machine (SVM)

The one-class support vector machine is a very specific instance of a support vector machine which is geared for anomaly detection. The generic support vector machine (SVM) can be used to classify data in multiple dimensions by finding an appropriate decision boundary. Like neural networks, support vector machines perform classification using nonlinear boundaries, however they go one step beyond neural networks by finding the boundaries that provide the maximum margin between different classes of data. Additionally, using the support vector machine one can map data from a lower dimensional space that is not linearly separable to a higher (even infinite-dimensional) space where the data are linearly separable by a hyperplane. This is performed by using what is commonly known in machine learning as the "kernel trick," when using SVM's. A kernel function is chosen to map the data from the lower-dimensional space to the higher-dimensional space. It can be chosen arbitrarily so as to best suit the data and at the same time reduce the computational burden involved with generating the mapped values by direct evaluation. However, for our purposes, we choose the Gaussian radial basis function or kernel, given by Eqn. 9, which is the most widely used in the application of one-class SVM's.

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}} \tag{9}$$

K is the kernel function, and the training data vectors are given by $\mathbf{x}_i$ and $\mathbf{x}_j$. The kernel width is given by σ, and is used to control the distribution of the kernel function around the training data point. The magnitude of this value also often has a direct effect on the algorithm's speed and complexity. "Support vectors" correspond to those data points that lie along the margin or closest to it. The maximum margin between classes is found by solving a quadratic optimization problem.

The one-class SVM differs from the generic version of the SVM in that the resulting quadratic optimization problem includes an allowance for a certain small predefined percentage of outliers, ν, making it suitable for anomaly detection. As shown in Fig. 7, these outliers lie between the origin and the optimal separating hyperplane. All the remaining data fall on the opposite side of the optimal separating hyperplane, belonging to a single, nominal class, hence the terminology "one-class" SVM. The SVM outputs a score that represents the distance from the data point being tested to the optimal hyperplane. Positive values for the one-class SVM output represent normal behavior (with higher values representing greater normality) and negative values represent abnormal behavior (with lower values representing greater abnormality). For subsequent analyses and in the results section, the *negative* of this score will be used in order to establish a frame of reference commensurate with ROC analysis and for comparison to other algorithms. More technical details on the one-class SVM are available in previously published studies (Das *et al.*, 2007); (Cohen *et al.*, 2004).

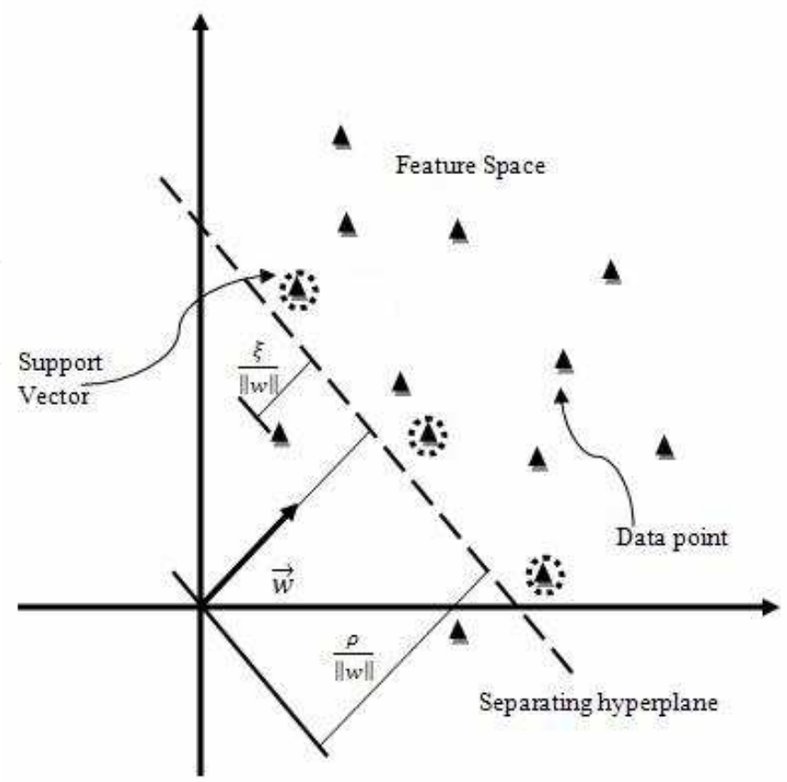

Fig. 7. Illustration of One-Class Support Vector Machine Framework

The one-class SVM differs from the other methods discussed in this paper because it determines whether or not a point is an outlier based on the distance of the point to a separating hyperplane in a feature space induced by a kernel operator, whereas most of the other methods rely on an analysis of the data in the original data space. For the one-class SVM, a single hyperplane separates the nominal data from the origin. Thus, for a system which undergoes nominal mode changes during its operation, all such changes will be characterized as nominal with a single hyperplane. Orca and IMS, on the other hand, characterize the anomalousness of a point based on local characteristics within the data space. This quality can make those algorithms more robust to significant mode changes compared with the one-class SVM.

The small predefined percentage of outliers, ν, is typically selected *a priori* and used to inform optimal selection of a value for the kernel width, σ, as is shown in Fig. 8. This is performed by finding the correct training classification rate as a function of the kernel width, and finding the value of σ that corresponds to 1-ν. It is often easy to find this value within a reasonable level of tolerance. In these cases, the kernel width, σ, corresponding to the first exceedance of 1-ν is used as the optimal value. However, when using a very small value of ν, it may be infeasible to find a training misclassification rate that will achieve this value (1-ν). This puts a lower bound on the predefined percentage of outliers that can be selected. As such, various candidates can be tested for feasibility in order to determine the lower bound on ν. For each of these candidates, a corresponding relevant metric can be computed, in addition to the corresponding optimal kernel width. In this case, the relevant test metric of interest is the AUC value. A value of ν that has been determined to be feasible,

and corresponds to the highest AUC value will thus be used for selection of the optimal kernel width.

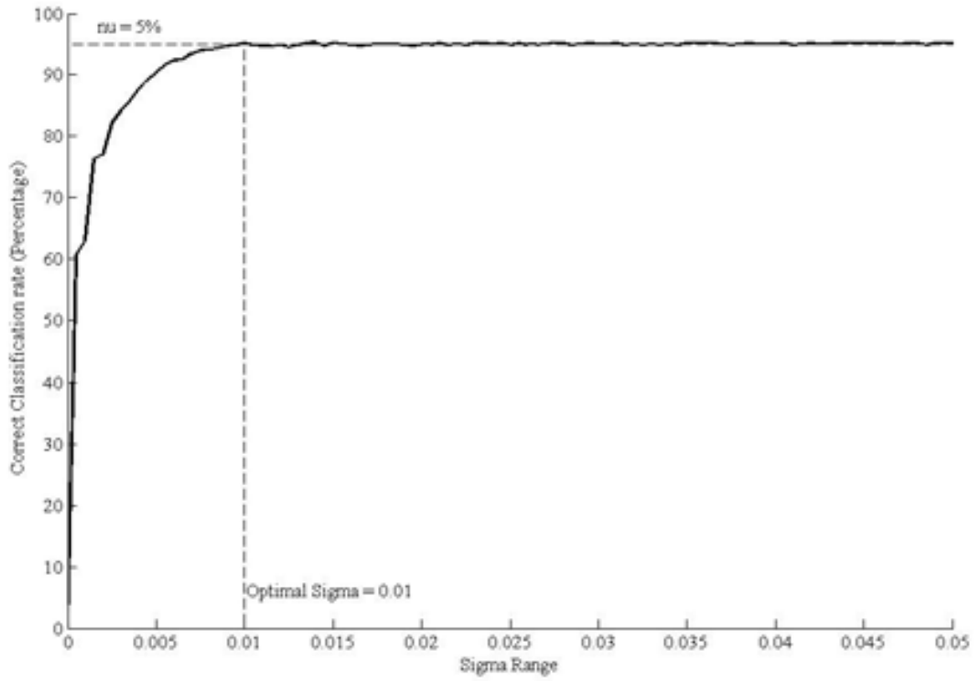

Fig. 8. Illustration of Selection of Optimal Kernel Width

The complexity of the one-class SVM algorithm is also of interest. As such, we provide an empirical complexity analysis shown in Fig. 9 to complement Figs. 2-4 and 6, which illustrate the complexity for IMS and Orca. Similar to Orca in Fig. 6, it is evident that the one-class support vector machine algorithm yields a superlinear complexity shown in Fig. 9. Furthermore, for the smaller dataset containing 21 resampled parameters, we compare the time for training only (shown as a dotted line), to the time required for normalization, training and validation (shown as a dash-dot line), which adds negligible time. It is also interesting to note that there is no noticeable multiplicative increase in complexity with the processing of additional parameters as opposed to Orca and IMS (cf. 21 resampled parameters to 164 change-only parameters), which is ostensibly due to the use of the kernel trick. However, regardless of this slight advantage, again the computational complexity of the one-class SVM algorithm is far greater than that of IMS, and on par with that of Orca. As seen in Fig. 9 , for datasets spanning a period of time greater than 7 hrs., it is readily apparent that the OCSVM algorithm is no longer able to run faster than real time.

4. Validation and evaluation criteria

In order to motivate a detailed discussion of the evaluation criteria, we will first need to introduce the means by which we will validate the algorithms discussed thus far. Specifically, we need to outline the type and volume of data required for validation. Recall

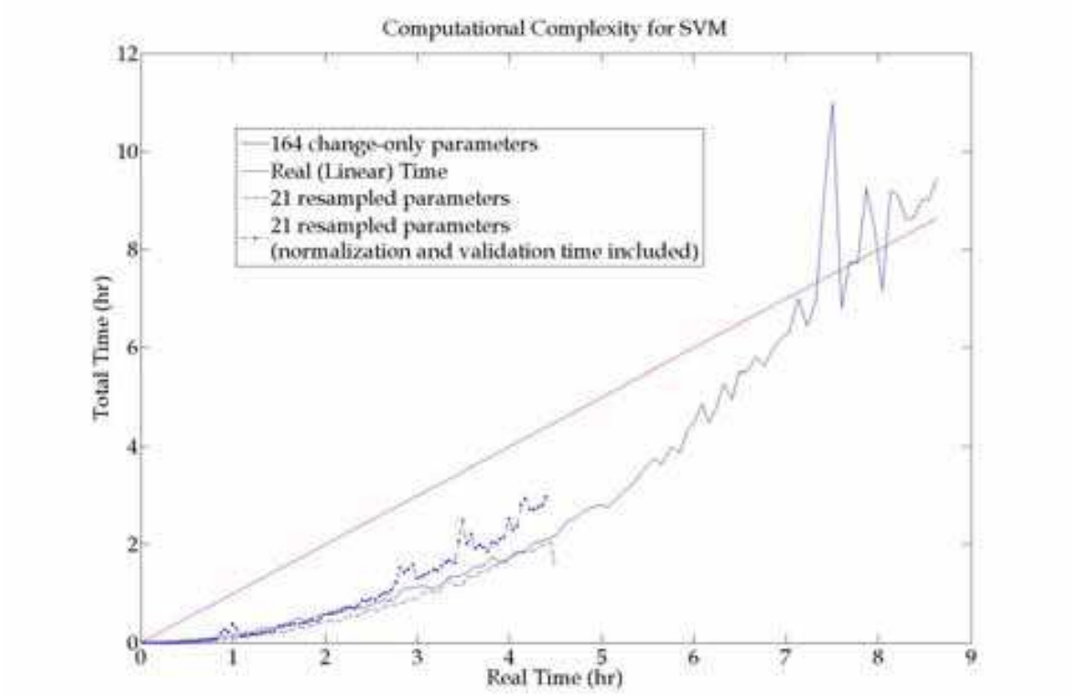

Fig. 9. One-Class Support Vector Machine Computational Complexity

the requirement to process a sufficient amount of data to undergo a formal cross-validation procedure that can be processed in a reasonable amount of time. The computational complexity of each data-driven algorithm is now well established, which facilitates the selection of various candidate cross-validation experiments. Again, for the purposes of our analysis here we will only consider two primary phases of operation of the TVC subsystem based upon meeting these specified data requirements. The two phases of interest involve both testing of the TVC in the VAB, and when the fully assembled vehicle is at the pad after launch countdown has commenced. For each of these phases, there are specific periods of time in which respective tests of interest or important elements of the operational sequence are conducted. Certain pre-specified failure modes from a FMECA (Failure Modes, Effects and Criticality Analysis) have been identified as potential adverse events that could occur during these critical periods of time.

Table 2 provides a summary of each failure mode under consideration for each phase of operation. The periods of time spanned by tests corresponding to each of these failure modes are also provided, which aid important considerations of computational complexity.

For the last three row entries in Table 2, the period of time spanned by the tests are largely dependent upon the specific flight, and are also driven by acquisition of a sufficient amount of data to accurately characterize nominal behavior in order to prevent false alarms. In light of this fact, the use of a 25 Hz sampling rate, and the example of how quickly time complexity for even the least burdensome algorithm adds up, we will use the first two row entries in Table 2 as the basis of experiments and results to be subsequently presented. Another advantage of using the two failure scenarios at the pad is due to the fact that they occur during the same time period within T – 1 min prior to launch, thus simplifying the experimental construction and allowing for fewer scenarios to investigate independently.

Operational Phase	Failure Mode	Period of Time Spanned
Pad	FSM (Fuel Supply Module) Pressure Drop due to N_2H_4 (Hydrazine) Leak	Within T – 1 min prior to launch
Pad	Hydraulic Fluid Reservoir Level Drop due to hydraulic fluid leak	Within T – 1 min prior to launch
VAB	Actuator Stuck during Actuator positioning test	2.5 min test in VAB
VAB	FSM (Fuel Supply Module) Pressure Drop due to N_2H_4 (Hydrazine) Leak	Within ≥34 minute period after calibration test in VAB
VAB	Hydraulic Fluid Reservoir Level Drop due to hydraulic fluid leak	Within ≥10 min period during TVC actuator tests in VAB
VAB	Hydraulic pumping unit overtemperature failure	Within ≥25 min period during tests in VAB

Table 2. Failure Mode Summary

Due to the absence of real data to use for validation and testing for the two candidate failure modes under consideration in Table 2, they must be simulated using existing means. Increasing the range of fidelity of available failure simulations is an ongoing process. In previous work, (Schwabacher *et al.*, 2009) a high fidelity physics-based simulation of a leak in a liquid propulsion-based J-2X engine was used to validate data-driven algorithms. However, such a simulator does not currently exist for relevant components of the TVC subsystem. In future work, we plan to develop high fidelity physics-based simulations of a leak in the fuel supply module (FSM) of the TVC subsystem, which is a spherical tank containing liquid hydrazine (N_2H_4) pressurized by GN_2 (gaseous nitrogen). However, for our purposes here we have chosen to use a rather low fidelity simulation for the two scenarios. The simple use of linearly decreasing ramps from nominal operating conditions to off-nominal values for parameters specific to each failure mode will be used, given predefined rates of degradation and preselected off-nominal values.

The time of failure injection within the T- 1 min period prior to launch is selected to leave a significant nominal fraction of the period at from T -1 min until injection of the failure, which occurs at T-28.12 sec for the hydrazine leak at the pad, levelling off at its off-nominal value at T-15 sec. For the hydraulic fluid reservoir leak at the pad, the failure is injected at T-13 sec and levels off at its nominal value at T-12 sec. However, it is often useful to supplement our validation data with additional cases of purely nominal behavior as well. In our case, we will use a concatenation of nominal data from T-1 min to launch, and two additional failure datasets representing the scenarios described above. Now that we have established the basis for constructing the validation data sets, it is of interest to describe how the algorithms will be optimized and evaluated with the use of ROC curve analysis.

The ROC curve essentially plots the true positive rate against the false alarm rate for all possible threshold values, as shown in Fig. 10. It therefore can be used as a design tool in order to select an alert threshold according to pre-established requirements for minimum

missed detection and/or false alarm rates. However, in order to compute true positive and false alarm rates, it is necessary to obtain a "ground truth" representation for each monitored example. In this case, an "example" can represent an individual validation flight, or a single time point. Here we will compute false alarm and missed detection rates based upon the classification of each individual time point, rather than on a flight-by-flight basis, as was performed in previous studies (Martin *et al.*, 2007).

In order to compute true positive and false alarm rates with a reasonable level of accuracy, there is a need to obtain a statistically significant number of labeled examples, both nominal and anomalous. The availability of nominally classified examples often far exceeds the availability of anomalously classified examples. The deficit of the latter drove our efforts to simulate faults, and hence to use classified time points as monitored examples in lieu of individual validation flights for construction of the ROC curve. For our purposes here, the classified time points represent the "ground truth," and will be based upon the times of failure injection specified previously, for each failure scenario.

One advantage of using ROC curve analysis is in its inherent robustness against the use of skewed distributions between the populations of nominally and anomalously categorized examples. Thus, as long as the population of failure examples is statistically significant, it does not need to be on par with the population of nominal examples. Furthermore, unlike other alert threshold selection techniques, the ROC curve may also be used for optimized selection of algorithmic parameters. This can be performed by using the area under the ROC curve, which represents overall classification discriminability. Therefore, the AUC can be maximized with respect to free design parameters used for algorithmic tuning to optimize and control performance during the validation stage. As such, we can ensure that the resulting model used to perform threshold or alert selection has the best anomaly detection capability possible. Furthermore, the AUC analysis can be used to compare the performance of all candidate algorithms.

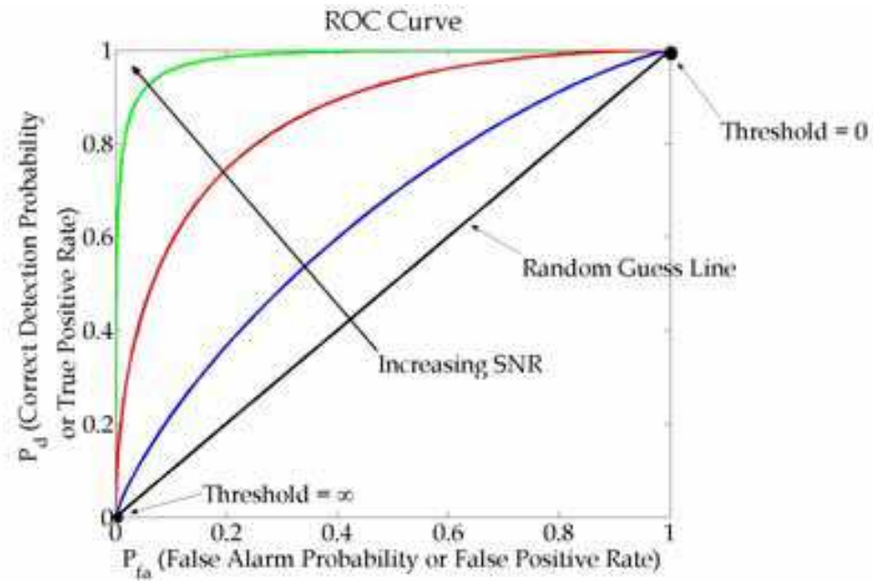

Fig. 10. Sample ROC Curve

As seen in Fig. 10, the area under the ROC curve has a classic increase in relation to the SNR (Signal-to-Noise ratio). This relationship is well established, and is derived from the origins

of the ROC curve for use in radar applications dating back to WWII. Unlike the SNR, it is often the case that algorithmic design parameters do not have a similar straightforward relationship to the AUC. However, this relationship is implicit in Fig. 10 due to the fact that optimization of algorithmic design parameters is performed by using the area under the ROC curve. In a sense, the ROC curve is "tuned" by attempting to maximize the AUC (area under the ROC curve) and choosing the appropriate algorithmic design parameter(s) to allow for maximum predictive capability. This can be thought of as choosing the ROC curve with the highest SNR based upon such parameters. In this case theoretically the "signal" might loosely be thought of as the *variability* of anomalous behavior, and the "noise" might be thought of as the *variability* of nominal operations.

5. Results and discussion

As previously discussed, a formal cross-validation procedure will be used to perform optimization and alert threshold selection. Doing so will help to prevent overfitting, as is often the case when using a single training and validation dataset. A k-fold cross-validation will be used, in which k-1 training examples are used, and the remaining example is used for validation. Each of the k examples is rotated and used one time for use as a validation set, so that there are k sets of statistics rather than a single one. The candidate training/validation dataset partitions will span 7 flights (STS-117, STS-107, STS-112, STS-113, STS-114, STS-120, and STS-122) containing 30 continuous-valued parameters, and a 7-fold cross-validation procedure will be used. Resulting AUC and ROC curves will be averaged over the results of the 7-fold cross-validation. The optimization parameters used and the respective optimized values derived from the average AUC across all 7 folds for each of the candidate data-driven algorithms are summarized in Table 3.

Algorithm	Optimization Parameter	Optimized Value	Max. of averaged AUC Value Achieved	"Hyperparameter" Values
IMS	Maximum cluster radius (MAX_INTERP)	.01894 (403 clusters)	.9807	Initial cluster size (INIT_TOLERANCE) = 0.01 Cluster growth percent (EXTRAP_PERCENT) = 0.05
Orca	k (number of nearest neighbors)	2	0.91077	N/A
OCSVM	σ (kernel width)	10.8081	0.92956	ν = 0.2289
OCSVM	σ (kernel width)	2.121	0.93072	ν = 0.2
OCSVM	σ (kernel width)	10.8081	0.90416	ν = 0.1
OCSVM	σ (kernel width)	2.121	0.9051	ν = 0.08
OCSVM	σ (kernel width)	0.74531	0.94666	ν = 0.075
OCSVM	σ (kernel width)	2.121	0.90664	ν = 0.07
OCSVM	σ (kernel width)	5.9636	0.90529	ν = 0.05
OCSVM	σ (kernel width)	5.9636	0.90525	ν = 0.01
OCSVM	σ (kernel width)	3.5565	0.89681	ν = 0.001
OCSVM	σ (kernel width)	infeasible	infeasible	ν = 0.0001

Table 3. Optimization Parameters

Additionally, the optimized maximum AUC values achieved by averaging across all 7 folds, and values of any supplementary "hyperparameters" that were used for each algorithm are provided in Table 3. Fig. 11 illustrates the average AUC values across all 7 folds for each of the candidate data-driven algorithms as a function of each of their design parameters. In addition, quasi-confidence bounds representing the maximum and minimum AUC values across all 7 folds are provided. In the lower right panel of Fig. 11, the maximum of the averaged AUC value achieved that lies in a feasible range is demarcated, even though it is clearly not the maximum over the entire range of kernel width values shown. The range of feasible kernel width values for the row in Table 3 highlighted in red represents any value greater than the optimized value shown, and corresponds to the lower right panel in Fig. 11. Both the highlighted row and the lower right panel of Table 3 and Fig. 11 respectively represent the maximum achievable AUC averaged across all 7 folds for the OCSVM algorithm. Thus, the optimized kernel width lies on the very border of the feasible range. Note that the last row of Table 3 indicates that a lower bound on the predefined percentage of outliers, ν_{min}, has been identified for any kernel width.

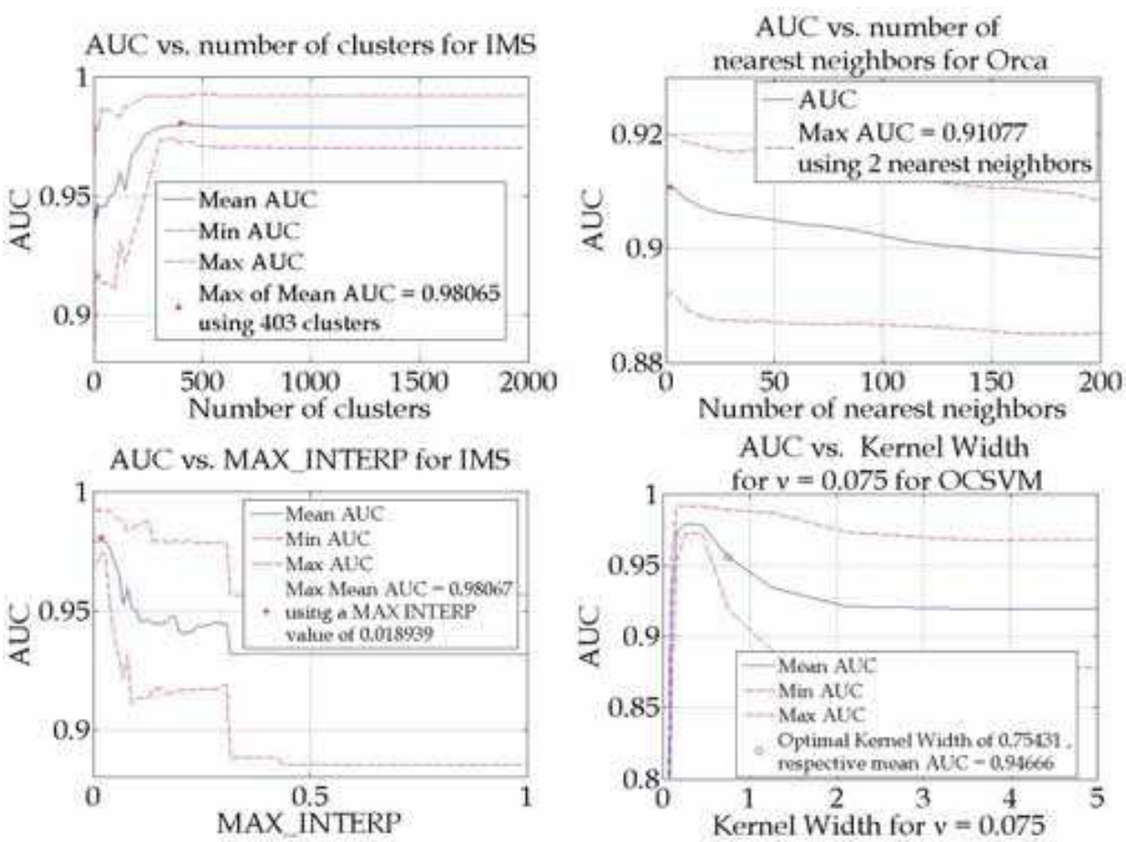

Fig. 11. AUC optimization with confidence intervals

Conflating the values presented in Table 3 with the graphs shown in Fig. 11, it becomes clear that IMS outperforms the other two algorithms, even when the hyperparameters for IMS are held fixed and not implicitly used to inform the optimization problem, as was performed with the one-class SVM. However, what may not be as clear from Fig. 11 is that some overlap between the confidence intervals among the algorithms exists. Specifically, for IMS,

the averaged optimized AUC value achieved falls within the range (0.9728, 0.9921), for Orca the range for optimized AUC values is (0.8924, 0.9201), and for OCSVM the range for optimized AUC values is (0.919, 0.9891). Therefore, there is some overlap between the optimum ranges for IMS and OCSVM, albeit a very small overlap of 0.0163 in which SVM might outperform IMS. Orca has no range overlap with IMS, but has an extremely small overlap with SVM of 0.0011.

Our next and final step is to perform threshold or alert selection based upon the ROC curves corresponding to the optimized parameters found from the previous step. Recall that this allows for alert threshold selection based upon specified minimum false alarm and/or missed detection rates with a robust anomaly detection capability and the best classification discriminability. Fig. 12 shows examples of the optimized ROC curves averaged over all 7 folds and their minimum/maximum confidence interval counterparts. Additionally, corresponding alert thresholds have been selected according to an established maximum allowable false positive rate of 0.01, and the results are shown in the legend.

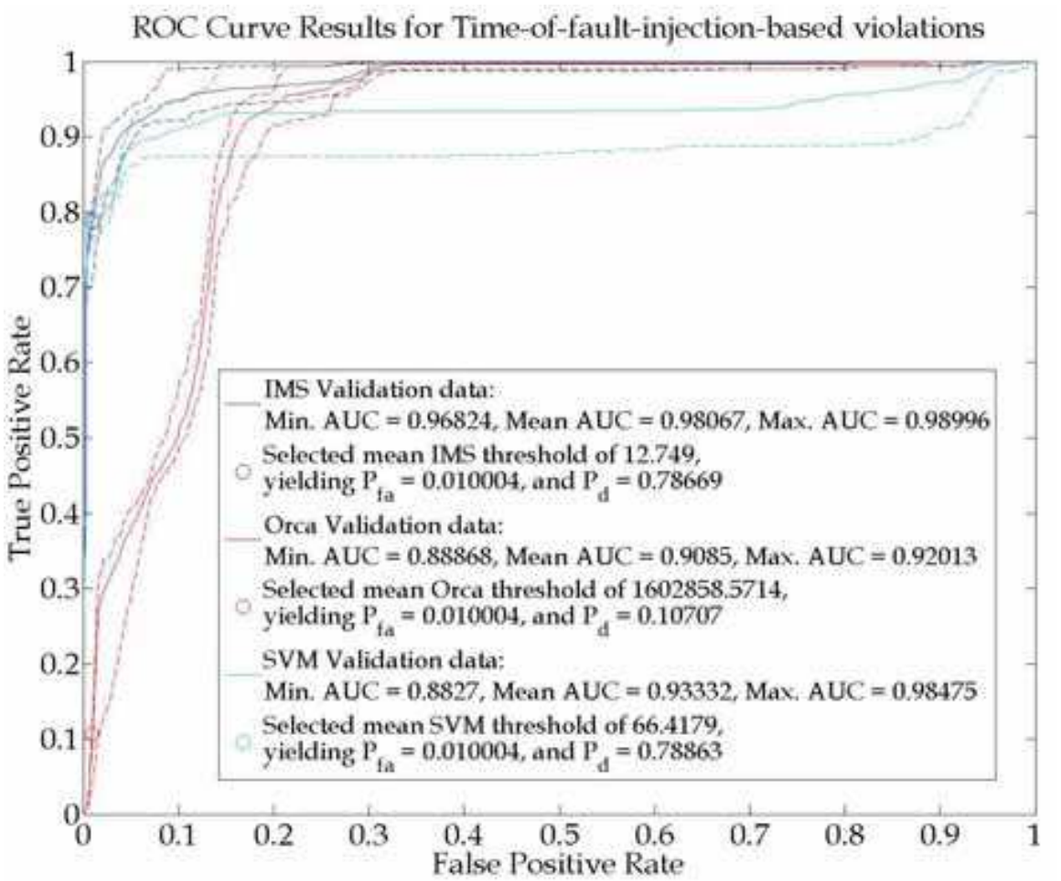

Fig. 12. ROC Curve Comparison and Alert Threshold Selection

The slight variation in ranges shown in the legend differ from the optimized AUC ranges previously provided due to the revised manner in which they have been computed. The minimum and maximum ROC curves were constructed in a modified manner in order to prevent any excursions of the mean ROC curve outside of the min/max envelope for visualization purposes. As such, the mean false positive rate is used consistently to construct

ROC curves representing the min/max bounds, while the min/max true positive rate is used respectively, resulting in different min/max AUC values than previously reported, however these variations are negligible. Note that for a fixed false alarm rate ceiling (P_{fa}) of 0.01, the resulting OCSVM threshold yields a true positive rate (P_d) on par with that of IMS (mean 0.79), and the resulting Orca threshold yields a much lower a true positive rate (P_d) having a mean of 0.11. In due fairness to Orca, it is possible that the AUC value may have increased beyond the range of the 200 nearest neighbors shown on the upper right panel of Fig. 11. However, due to scalability and complexity issues, the experiment was terminated as shown. Indeed, the complexity of running Orca with an increased number of nearest neighbors scales quadratically when not using any pruning rules.

Finally, Fig. 13 illustrates realizations of validation flight STS-117 to which optimized parameters and thresholds are applied.

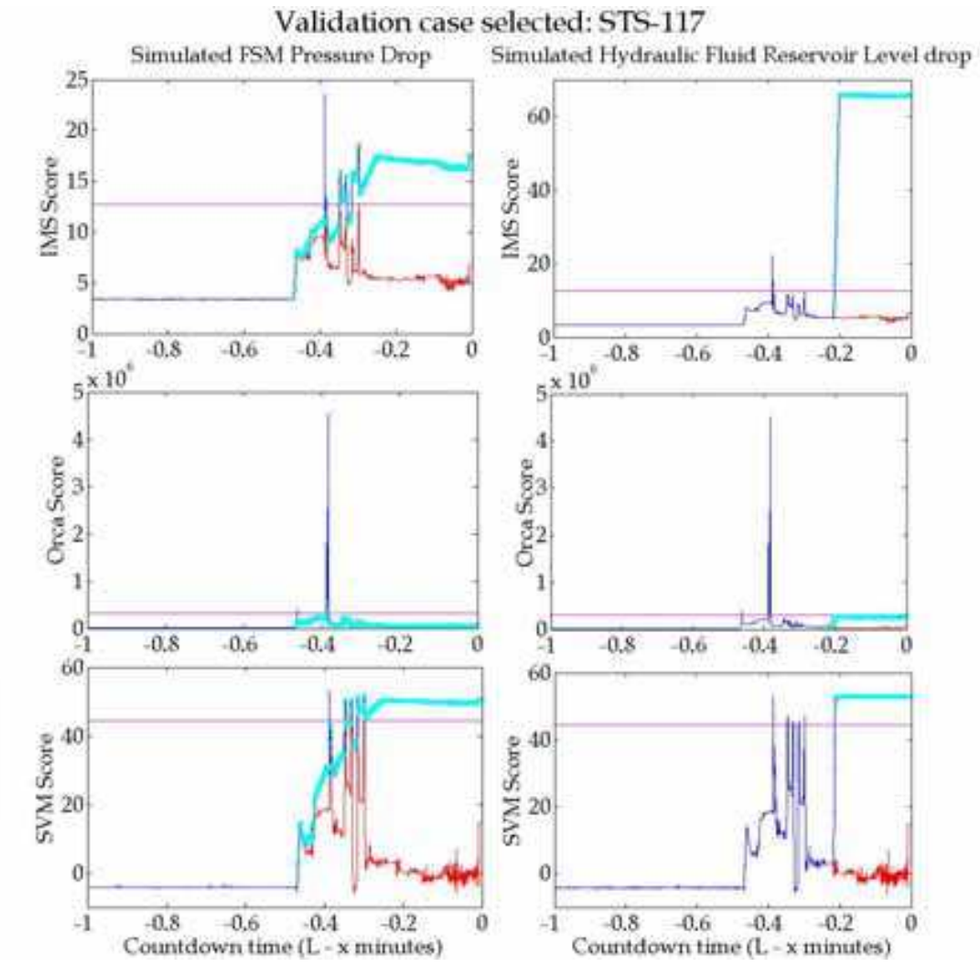

Fig. 13. Realizations of Optimized Parameters and Thresholds Applied to Validation Flight

Fig. 13 illustrates both the nominal (red lines) and fault injected scores (blue lines) from T – 1 min to launch at the pad for all algorithms. In all cases, both failure scenarios involving an FSM pressure drop (left panels) and a hydraulic fluid reservoir leak (right panels) are shown, using the optimized thresholds (fixed magenta colored lines) found from the ROC analysis. The actual threshold values may vary slightly from the values specified in Fig. 12

due to realizations being based on independent experiments, and as such should still fall within the confidence bands formed in both Figs. 11 and 12. The light blue highlights superimposed over the failure injected scores in dark blue represent the "ground truth" time of failure injection and duration, so as to give a feel for the false alarm and correct detection rates. Evidently, there is a clear bifurcation between nominal and anomalous scores for both IMS and SVM, and for Orca the same is true although it is less apparent. As can be discerned from Figs. 11 – 13, we have identified the fact that both in complexity and accuracy, IMS seems to be the best choice among all of the algorithms investigated. However, there is some overlap in the confidence intervals for IMS and SVM AUC values, and the alert thresholds applied for both corresponding ROC curves yield almost identical true positive rates.

6. Conclusion and next steps

We have provided a thorough end-to-end description of the process for evaluation of three different data-driven algorithms for anomaly detection. Through optimization of algorithmic parameters using the AUC, we were able to choose parameters yielding the best detection capability. The respective ROC curves corresponding to these parameters were then used to inform alert threshold selection by enforcement of a maximum allowable false alarm rate. It was found that IMS was the best performing algorithm when considering both computational complexity and accuracy. However, when evaluating the results based upon accuracy alone, the OCVSM approach is competitive with IMS due to overlapping confidence intervals present in the accuracy results.

In subsequent research studies, we will provide results of unseen hold out test cases to which optimized parameters and thresholds will be applied, in order to provide additional evidence demonstrating the superiority of a particular algorithmic technique. Furthermore, we will employ a variant of the AUC that only considers performance evaluation for algorithmic comparison restricted to low false positive rates. A slightly modified definition of false alarms and missed detections that accounts for pre-defined latencies and prediction horizons will also be investigated.

7. Acknowledgements

The author would like to acknowledge the support of the Ares I-X Ground Diagnostics Prototype activity, which was funded by the NASA Constellation Program, by the NASA Exploration Technology Development Program, and by the NASA Kennedy Space Center Ground Operations Project. Furthermore, the author graciously acknowledges the reviews of Dr. Mark Schwabacher, Bryan Matthews, and Dr. Ann Patterson-Hine. The author also extends appreciation to John Wallerius for his contribution of the subsection on IMS score computations, and his ideas pertaining to next steps on consideration of performance evaluation for algorithmic comparison restricted to low false positive rates. Finally, the author acknowledges the permission to use of Figs. 7 and 8 from Dr. Santanu Das, and Fig. 1a with supporting text from David Iverson.

8. References

Bay, S.D. & Schwabacher, M. (2003). 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, pp. 29–38, New York, NY, 2003.

Cavanaugh, K. (2001). An integrated diagnostics virtual test bench for life cycle support, *Proceedings of the IEEE Aerospace Conference*, pp. 7–3235–7–3246, ISBN: 0-7803-6599-2, Big Sky, Montana, March 2001.

Cohen, G.; Hilario, M. & Pellegrini, C. (2004). One-class support vector machines with a conformal kernel: a case study in handling class imbalance, In: *Structural, Syntactic, and Statistical Pattern Recognition*, A. Fred *et al.* (Eds.), pp. 850–858, Springer-Verlag, Berlin, Heidelberg, Germany.

Das, S.; Srivastava, A. & Chattopadhyah, A. (2007). Classification of Damage Signatures in Composite Plates using One-Class SVM's, *Proceedings of the IEEE Aerospace Conference*, Big Sky, MO, March 2007.

Fragola, J.R. (1996). Space shuttle program risk management, *Proceedings of the International Symposium on Product Quality and Integrity: Reliability and Maintainability Symposium*, ISBN: 0-7803-3112-5, pp. 133–142, Jan 1996.

Hart, G.F. (1990). Launch commit criteria performance trending analysis, *Annual Proceedings of the Reliability and Maintainability Symposium*, pp. 36–41, Jan 1990.

Iverson, D.L.; Martin, R.; Schwabacher, M.; Spirkovska, L.; Taylor, W.; Mackey, R. & Castle. J.P. (2009). General purpose data-driven system monitoring for space operations, *Proceedings of the AIAA Infotech@Aerospace Conference*, Seattle, Washington, April 2009.

Mackey, R.; James, M.; Park, H. & Zak. M. (2001). BEAM: Technology for autonomous self-analysis, *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, 2001.

Martin, R. (2007). Unsupervised anomaly detection and diagnosis for liquid rocket engine propulsion, *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, March 2007.

Martin, R.; Schwabacher, M.; Oza, N. & Srivastava, A. (2007). Comparison of unsupervised anomaly detection methods for systems health management using Space Shuttle main engine data, *Proceedings of the 54th Joint Army-Navy-NASA-Air Force Propulsion Meeting*, Denver, CO, May 2007.

Paté-Cornell, E. & Dillon, R. (2001). Probabilistic risk analysis for the NASA space shuttle: a brief history and current work. *Reliability Engineering & System Safety*, Vol. 74, No. 3, (Dec. 2001) pp. 345 – 352.

Paté-Cornell, E. & Fischbeck, P.S. (1994). Risk Management for the Tiles of the Space Shuttle. *Interfaces*, Vol. 24, No. 1, (Jan. – Feb. 1994) pp. 64–86.

Schwabacher, M. (2005). Machine learning for rocket propulsion health monitoring, *Proceedings of the SAE World Aerospace Congress*, pp. 1192–1197, Dallas, Texas, 2005.

Schwabacher, M. & Waterman, R. (2008). Pre-launch diagnostics for launch vehicles, *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, March 2008.

Schwabacher, M; Aguilar, R & Figueroa, F. Using Decision Trees to Detect and Isolate Simulated Leaks in the J-2X Rocket Engine, *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, 2009.

Tu, H.; Allanach J.; Singh S.; Pattipati K.R. & Willett, P. (2006). Information integration via hierarchical and hybrid Bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, Vol. 36, No. 1 (Jan. 2006) pp.19–33.

Tumer, I. (2005). Design methods and practices for fault prevention and management in spacecraft, *Technical report, NASA Ames Research Center*, 2005.

Aerospace Technologies Advancements

Edited by Thawar T. Arif

ISBN 978-953-7619-96-1

Hard cover, 492 pages

Publisher InTech

Published online 01, January, 2010

Published in print edition January, 2010

Space technology has become increasingly important after the great development and rapid progress in information and communication technology as well as the technology of space exploration. This book deals with the latest and most prominent research in space technology. The first part of the book (first six chapters) deals with the algorithms and software used in information processing, communications and control of spacecrafts. The second part (chapters 7 to 10) deals with the latest research on the space structures. The third part (chapters 11 to 14) deals with some of the latest applications in space. The fourth part (chapters 15 and 16) deals with small satellite technologies. The fifth part (chapters 17 to 20) deals with some of the latest applications in the field of aircrafts. The sixth part (chapters 21 to 25) outlines some recent research efforts in different subjects.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen