Fleet Level Anomaly Detection of Aviation Safety Data

Santanu Das
UARC, UC Santa Cruz
Nasa Ames Research Center
Moffett Field, CA 94035
Email: Santanu.Das-1@nasa.gov

Bryan L. Matthews
SGT Inc.
Nasa Ames Research Center
Moffett Field, CA 94035
Email: Bryan.L.Matthews@nasa.gov

Robert Lawrence
Consultant/Contractor
Nasa Ames Research Center
Moffett Field, CA 94035
Email: captndaddy@gmail.com

***Abstract*—For the purposes of this paper, the National Airspace System (NAS) encompasses the operations of all aircraft which are subject to air traffic control procedures. The NAS is a highly complex dynamic system that is sensitive to aeronautical decision-making and risk management skills. In order to ensure a healthy system with safe flights a systematic approach to anomaly detection is very important when evaluating a given set of circumstances and for determination of the best possible course of action. Given the fact that the NAS is a vast and loosely integrated network of systems, it requires improved safety assurance capabilities to maintain an extremely low accident rate under increasingly dense operating conditions. Data mining based tools and techniques are required to support and aid operators' (such as pilots, management, or policy makers) overall decision-making capacity. Within the NAS, the ability to analyze fleetwide aircraft data autonomously is still considered a significantly challenging task. For our purposes a fleet is defined as a group of aircraft sharing generally compatible parameter lists. Here, in this effort, we aim at developing a system level analysis scheme. In this paper we address the capability for detection of fleetwide anomalies as they occur, which itself is an important initiative toward the safety of the real-world flight operations. The flight data recorders archive millions of data points with valuable information on flights everyday. The operational parameters consist of both continuous and discrete (binary & categorical) data from several critical subsystems and numerous complex procedures. In this paper, we discuss a system level anomaly detection approach based on the theory of kernel learning to detect potential safety anomalies in a very large data base of commercial aircraft. We also demonstrate that the proposed approach uncovers some operationally significant events due to environmental, mechanical, and human factors issues in high dimensional, multivariate Flight Operations Quality Assurance (FOQA) data. We present the results of our detection algorithms on real FOQA data from a regional carrier.**

I. Introduction

Suppose the entire database that is available for a given system, covering both inputs and outputs, is given by the set $(\mathcal{U}, \mathcal{Y})$. In real-world flight operations, the input $\mathcal{U}$ can be due to exogenous disturbances or provided by the pilot following some standard-operating procedure for given airport, aircraft, weather conditions, instructions from air traffic control, and other contextual elements of the flight, the flight procedures are well determined. The observed characteristics $\mathcal{Y}$ are representative of the flight behavior and hold a complex relationship with the input $\mathcal{U}$. Such a system that can be functionally described by the following equations:

$$\begin{aligned} \mathbf{h}_t &= \Gamma(\mathbf{h}^*_{t-1}) \\ \mathbf{x}_t &= \Psi(\mathbf{x}^*_{t-1}, \mathbf{h}^*_t, u_t) \\ y_t &= \Omega(\mathbf{x}_t) \end{aligned} \tag{1}$$

Equation 1 describes a system with u_t as the observed system input, and y_t is the observed system output which can take any form of discrete, categorical, and continuous features. Here we assume that the functions Γ and Ψ are unknown. The functions Γ and Ψ determine the evolution of the hidden system state $\mathbf{h}_t$ and govern the evolution of the continuous state vector respectively. We assume that the vector $\mathbf{x}$ is an N dimensional state vector, and $\mathbf{x}^*_{t-1}$ is its history for the last D time steps: $\mathbf{x}^*_{t-1} = [\mathbf{x}_{t-D}, \mathbf{x}_{t-D+1}, ..., \mathbf{x}_{t-1}]$. The hidden state $\mathbf{h}_t$ is assumed to correspond to different mode configurations within the system and each mode affects the output dynamics Ψ. In response to any malfunction in the system, $\mathbf{h}_t$ could move to an abnormal state, thus also changing the nature of the observed output. It is not necessary that $\mathbf{h}_t$ always has to move to an abnormal state. It can very well reside in some unfamiliar state within the normal operational regime. In practice a similar situation will arise when all pilots are attempting to follow the same standard operating procedures, while some of them may deviate from these procedures which could lead to a different input sequence $\mathcal{U}'$, resulting in a different set of observed flight characteristics $\mathcal{Y}'$ which may be unusual but not necessarily wrong. The problem that we address in this paper is to develop a

method to discover whether or not the current observed vector $\mathcal{Y}$ along with the discrete pilot inputs $\mathcal{U}$ represent a state that is atypical or abnormal based on the observed history of the system. We want to emphasize, here that in this particular research we don't intend to model dynamic behavior or relationship between input and output variables in systems described above. Rather the objective is to identify simultaneously unusual patterns in any combinations of input $(\mathcal{U})$ and output $(\mathcal{Y})$.

Over the last few decades, with improved sensing capabilities, we have seen tremendous increase in information flow, in terms of the volume and complexity, created at an unprecedented pace in several disciplines. The aviation industry is no exception. A good example is the Distributed National FOQA (Flight Operations Quality Assurance) Archive (DNFA) data base established by NASA. DNFA contains millions of flight data from most of the major carriers in the U.S. Typical FOQA parameters consist of both continuous and discrete data from the avionics, propulsion system, control surfaces, landing gear, the cockpit switch positions, and other critical systems. These data sets can have up to 500 parameters and are sampled at 1 Hz. For a moderate sized fleet that operates 1000 flights per day, these FOQA data sets become very large. Today, we are left with the challenge of dealing with such a vast amount of heterogeneous information resources in varieties of semantic structures. Knowledge discovery from these heterogeneous resources is still a challenging task. With this increased complexity of data sources there is a potential need for building intelligent refinement and integration frameworks, focusing on information content and semantics.

The key aspect of any data analysis method depends on how the input data was measured within the process and transformed into information. The data sources are categorized as structured data and unstructured data. In the aviation domain, a typical example of unstructured data is free text data, for example reports or scripts from pilots describing some events, expert feedback on the process etc. However in this paper we will restrict our discussion to analysis using structured data which has two main categories, continuous and discrete data. Discrete attributes can be either binary or categorical or logical order i.e. sequential in nature.

In our earlier research we demonstrated the potential of the Multiple Kernel based Anomaly Detection (MKAD) algorithm [9] in detecting anomalies. In this paper we attempt to expand on the analysis by reporting a variety of different interesting anomalies that we have observed in commercial aircraft data. Here we conduct the analysis on a much larger-scale dataset. In addition we provide some useful insight from domain oriented explanations. We demonstrate the capability of the proposed methodology in terms of its flexibility to process a variety of heterogeneous data sources without reformulating the problem whenever there is a change in information content or data structure. The formulation we demonstrate in this paper is very simple and can easily be adopted for fleetwide analysis in various domains including medical applications, airspace safety, business analysis etc.

II. Anomaly Detection Algorithms

The theme of this paper is anomaly detection, also known as outlier detection or surprise pattern detection. Outlier or anomaly detection refers to the task of identifying new or unknown patterns which, in many cases, are abnormal or inconsistent. The problem of outlier detection has been extensively studied using several approaches [14], [15], [16], [7]. Supervised and unsupervised outlier detection are the two broader categories. In the supervised approach a model is built for detection purpose and this model assumes known class labels. Typical classification based techniques such as Bayesian inference, decision trees, Support Vector Machines (SVMs)or neural network models are built on previously labeled instances of both normal and abnormal data instances. However class labels are expensive and they are not easily available, especially for most of the historical data. Given the fact that it is impossible to always have prior knowledge about all possible classes or have known data labels or have data representing all possible scenarios or classes, unsupervised techniques hold an edge in many applications and play an important role in identifying "*what is desired and what is not*" in a dataset. Novelty detector is one such specialized tool that classifies the members of a given set of objects into two groups on the basis of whether the model has seen those objects before or not. Kernel based classification methods like single class SVMs, one-class kernel Fisher Discriminants etc fall under the novelty detection category and are unsupervised in nature. In these techniques a model is built on the normal data and the idea is to come up with a threshold for determining abnormality and to use a distance based score for evaluating the extent of abnormality. In nearest neighbor based approach, the aim is to infer the outliers based on the data itself e.g. by finding those points which are at a greater distance from most of the other data points or by finding those points which are in a low density region. To address the quadratic time complexity which is a bottleneck of k-nn based solutions, researchers like

Angiulli and Pizzuti [3], Angiulli and Fassetti [2], Ramaswamy et al. [17], and Bay and Schwabacher [5] have proposed promising techniques with improved run time.

A. Production Level Implementations

In the airline industry, algorithms that are chosen for production level implementation must be heavily tested and developed to produce reliable and meaningful results before they are selected by the airlines for everyday use. The most widespread method for detecting operationally significant anomalies is with domain expert defined threshold exceedances. Methods for detecting unstable and high energy approaches commonly involve exceedance queries on multiple parameters. Examining the min and max values for each parameter and determining if they fall outside the expected normal range over a given phase of flight is also used to yield interesting anomalies. These methods have been in use for as long as the FOQA program has been in existence and have provided analysts with valuable results. This is in part due to the fact that the exceedance events identified are easily interpretable since the user has defined what the algorithm is looking for. Another attractive feature is the highly scalable implementation of the algorithm on large data sets. However, the drawback to exceedance based analysis is the fact that typically only the anomalies that are defined are reported to the analyst, leaving the undefined anomalous events undetected. In other words the method only answers the questions that someone thought to ask. Algorithms discussed in this paper address the issue of detecting "unexpected anomalies". They are still in the research level stage, but with increasing interest from the airlines may one day be running in a production level environment.

B. Research Level Implementations

In this section we will describe some anomaly detection techniques that have been extensively used to analyze FOQA data. Some of these algorithms include Morning Report, Orca, IMS, SequenceMiner, and one-class Support Vector Machines. These research level algorithms help complement the exceedance based methods by being able to identify the "unexpected anomalies". Once the anomalies found are analyzed, new parameter exceedances can be developed and incorporated into the airlines' daily analysis to track future or past events.

1) Morning Report: Morning Report [20], [1] is an algorithm designed to detect atypical flights over a set of aircraft and identify the contributing anomalous parameters and phases of flight. The algorithm calculates statistical signatures across the parameters of a given flight and clusters the flights based on the multivariate signatures. Similar flights are grouped together and atypical flights are considered to be far away from a cluster and therefore have higher scores. The distribution of the anomaly scores are a function of the Mahalanobis distance from centroids of the multivariate cluster. The results provide the degree of the atypicality along with the contributing parameters, which can be useful for the analysts. The algorithm was designed to be executed on a large set of flights overnight and return the results the next morning (hence the name Morning Report).

2) Orca: "Orca" [5] is a method used for detecting anomalies in both continuous and discrete (binary format) data in vector space, using a nearest neighbors based approach to detect anomalous points. For continuous data, Orca takes a nominal reference data set and calculates the nearest neighbors using Euclidean distance to all test points in the original vector space. For binary data points the hamming distance [21] is used and combined with Euclidean distance. "Orca" is a k-nearest neighbor based algorithm adopting nested loop structure in conjunction with randomization and simple pruning rule. Pruning used in this algorithm helps in achieving near linear time performance with high dimensional data. This makes the algorithm scalable for analyzing large data sets. The algorithm uses a distance-based metric for finding an outlier by examining the distance of any test point to k existing examples who are considered its nearest neighbors. If one looks at the local neighborhood and finds that the test points are relatively close, then the examples are considered normal. In this algorithm, each data point is scored independently and therefore anomalies in the temporal domain are undetectable. The pseudo code of "Orca" is shown in algorithm 1.

3) IMS: The Inductive Monitoring System (IMS) [11] is a distance based anomaly detection tool that uses a unsupervised anomaly detection algorithm that uses incremental clustering to build models of the expected operation of the system on a set of nominal data. The models are used to test new data to determine whether an anomaly is present or not. The underlying concept states that if the system behaves similar to the normal operating modes that the data was trained on, the distance scores will be lower than data that are generated from a system that is in an anomalous state. IMS evaluates each sample, which is a multivariate vector, by calculating the Euclidean distance to the cluster bounds of each cluster in the model, and reporting the distance to the closest cluster as the anomaly score. A 2D representation in Fig 1 can be seen. The normal operating regions are defined

Algorithm 1 Orca Algorithm

1: Input:
A_{pq} : Matrix with q dimensional dataset having p instances arranged in a random order
k : Number of nearest neighbors (default 5)
n : Number of outliers used (default: $n = p$)
2: Output:
$O(n)$: Set of outliers
S_{global} : Global scores
3: Argument:
a : Entries in A
B : Block of examples from A
b : Entries in B
C : Cut-off threshold
w_d : Weight of discrete parameters
4: Definitions:
$x \in (x_c, x_d)$
$D \in (D_c, D_d)$
$H_d(x_d, D_d) = w_{d_1}(x_{d_1}! = D_{d_1}) + w_{d_2}(x_{d_2}! = D_{d_2}) + \ldots + w_{d_n}(x_{d_n}! = D_{d_n})$
$d(x, D) = \sqrt{(x_{c_1} - D_{c_1})^2 + \ldots + (x_{c_n} - D_{c_n})^2} + H_d(x_d, D_d)$
$d(x, D)$: maximum distance between x and an example in D
$M^k_{x,D}$:k closest example in D to x
$S(D, x) = \frac{1}{m}\sum_{i=1}^{m} d(x, D)$ distance based score
5: Initialize:
Let p instances of A_{pq} be divided in N_B blocks and $K_{nn}(x)$ be the matrix that keeps track of the nearest neighbors/examples of x. And $C = 0$ and $O = \Phi$, where Φ is a null vector
6: For $block = 1 \colon N_B$ {
$B = A(:, block); K_{nn}(b) = \Phi$
7: For each a in A, {
8: For each b in B and $b \neq a$ {
9: If $Length(K_{nn}(b)) < k$ or $d(b, a) < dm(b, K_{nn}(b))$ {
$K_{nn}(b) \leftarrow M^k_{b,K_{nn}(b) \cup a}$
10: If $S(K_{nn}(b), b) < c$ {
Remove example b from set B
}}}}
11: $O = O \cup B$
12: $S_{global} = score(O)$
13: $C \leftarrow min(S_{global}(0))$
}

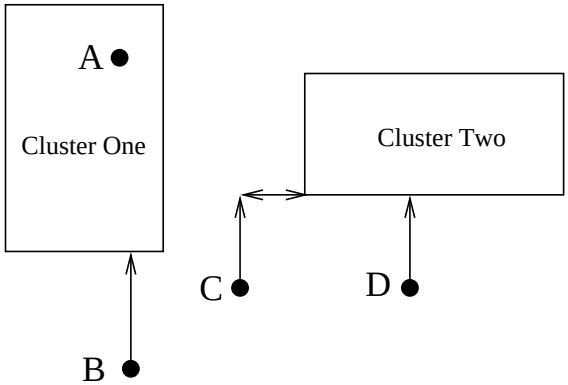


Fig. 1. A conceptual diagram to describe the working principle of IMS.

by the two boxes, with the distances computed to the edge of the nearest box. In the context of flight data IMS will train on a set of nominal flights, either identified by domain experts or another algorithm, and test on the remaining flights. Each time point within the test flight will be evaluated, producing a profile of anomaly scores for each flight. The anomaly scores for each flight can be combined in many different ways, however, typically the scores are averaged for each flight and the flights with the highest average score are ranked most anomalous. As with Orca, IMS evaluates each point independently and therefore suffers from the same drawback of not being able to detect anomalies in the temporal domain. The pseudo code for IMS is shown in the algorithms 2 & 3.

4) SequenceMiner: SequenceMiner [6] was developed to address the problem of detecting and describing anomalies in large sets of high dimensional symbol sequences such as recordings of switch sensors in the cockpits of commercial aircraft. SequenceMiner works by first using an unsupervised clustering algorithm to cluster the sequences using the normalized longest common subsequence (nLCS) as a similarity metric. Once the clusters are defined, anomalies can be detected using the nLCS as the distance measure. In this context anomalies are determined to have low similarities between the clusters of other sequences and are defined to be far away from a cluster. Once anomalies are identified, SequenceMiner applies a genetic algorithm to modify the sequence to draw it closer to the cluster. Keeping track of the changes made to the sequence, the algorithm reports back the missing and extra symbols, giving the user some context for the anomaly. Since SequenceMiner focuses on the sequential nature of the anomalies it can find anomalies that other algorithms such as Orca and IMS are unable to detect, however it is ineffective at handling continuous parameters without somehow

Algorithm 2 IMS: Train

1: Input: X_t (nominal system data vectors)
i (initial tolerance percent)
e (expansion percent)
m (max distance from cluster centroid to input vector)
2: **for** each input vector $\mathbf{X}_t$ normalize the values of $\mathbf{X}_t$ and find the cluster with the closest centroid to $\mathbf{X}_t$.
3: **if** no clusters exist, create a new cluster centered on $\mathbf{X}_t$, adding initial tolerance percent i to each vector value to create upper and lower bounds.
4: **else if** a closest cluster is found and $\mathbf{X}_t$ is within distance m of the centroid of that cluster, expand the cluster parameter boundaries as necessary to include $\mathbf{X}_t$ adding the expansion percent e to each parameter bound that is changed.
5: **else if** a closest cluster is found and $\mathbf{X}_t$ is beyond distance m of the centroid of that cluster, create a new cluster centered on $\mathbf{X}_t$, as in step **3**.
6: **end if**
7: **end for**
8: **Output**: B_c (parameter boundaries for each cluster)

Algorithm 3 IMS: Test

1: **Input**: $\mathbf{x}_t$ (test input)
2: **for** each $\mathbf{x}_t$ normalize the values of $\mathbf{x}_t$ and find the closest nominal cluster in B_c to $\mathbf{x}_t$.
3: **if** all $\mathbf{x}_t$ parameter values fall within the bounds of a cluster, the distance from $\mathbf{x}_t$ to the cluster is zero.
4: **else if** no cluster contains $\mathbf{x}_t$ locate the cluster with the hyper-box boundary that is closest to $\mathbf{x}_t$. Calculate the distance between a vector of $\mathbf{x}_t$ and a cluster hyper-box by summing the squares of the differences between each $\mathbf{x}_t$ parameter and the nearest cluster boundary value for that dimension, then find the square root of that sum.
5: **end if**
6: **end for**
7: **Output**: D_t (distance of vector to nearest cluster)
d_{tp} (distance of each parameter to cluster bounds)

drastically changing the nature of the data.

5) One-class SVMs: One-class SVM is a kernel based method that builds a model on single (known) class data and then finds a set of outliers using a decision boundary. Data which is non-linearly separable (or sometimes non-separable) in input space is mapped into much higher dimensional feature space ($\mathcal{F}$) where the data are linearly separable. The mapping of the data into $\mathcal{F}$ can be done by defining a similarity measure using the dot product in $\mathcal{F}$ in terms of a function operating on the input data space and thus computing the inner products more efficiently which is commonly referred as the "kernel trick" in the machine learning literature. This results in a kernel matrix K which gives a relative similarity between objects.

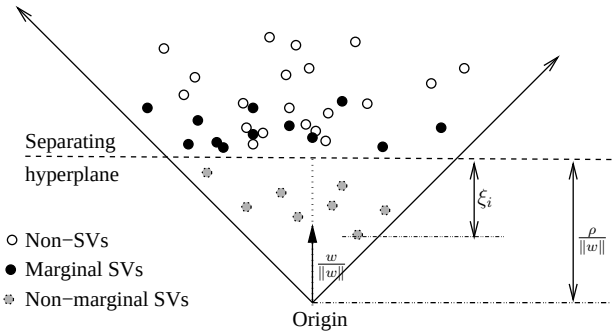


Fig. 2. This figure illustrates the geometric interpretation of optimal hyperplane for one class SVMs.

Schölkopf [18] showed that in the high dimensional feature space it is possible to construct an optimal hyperplane by maximizing the margin between the origin and the hyperplane in the feature space by solving the following dual problem,

$$\begin{aligned} \text{minimize} \quad & Q = \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j(\sum_{\lambda}\beta_\lambda K^{\lambda}_{i,j}) \\ \text{subject to} \quad & 0 \le \alpha_i \le \frac{1}{\ell\nu}, \sum_i \alpha_i = 1 \\ & \sum \beta_\lambda = 1, \rho \ge 0, \quad \nu \in [0,1] \end{aligned} \tag{2}$$

where ν is a user specified parameter that defines the upper bound on the training error, and also the lower bound on the fraction of training points that are support vectors. Based on the suggestion of [13], we have modified the optimization formulation of [18] such that the resulting kernel is the convex combination of different kernels. β_λ correspond to the weights of the kernels and α_i is Lagrange multiplier. The key idea is to construct a hyperplane that can separate outliers (also known as

slack variables) from the rest of the training examples, as shown in Fig. 2. In Fig. 2, the distance of slack variables from the hyperplane is represented by the ξ_i and the offset (also known as bias) by ρ. We wish to develop a decision rule (Eqn. 3) from the training samples, so that new data can be labeled as normal or anomalous.

$$f(\vec{x}_z, \alpha, \beta, \rho) = sign(\sum_{i \in \mathcal{I}} \alpha_i (\sum_{\lambda} \beta_\lambda K^\lambda_{i,z}) - \rho) \quad (3)$$

The decision boundary is defined by a set of special training examples which are known as support vectors. In Fig. 2, support vectors are shown in shaded circles. The dark shaded circles are the called marginal or unbounded support vectors ($\{x_i : i \in [\ell], 0 < \alpha_i < 1\}$) and the light shaded circles are non-margin or bounded support vectors ($\{x_i : i \in [\ell], \alpha_i = 1\}$). The hollow circles are the non support vectors and they are not used in the model. This means that it is possible to re-build the same model with just using the training examples corresponding to the support vectors and not taking any of the non support vectors into account. If the decision function predicts a negative label for a given test point z, then it is classified as an outlier. Test examples with positive labels are classified normal. The pseudo code of one class SVMs is shown below.

Algorithm 4 One-class SVMs Algorithm

1: Input vector: $X = \{x_1, x_2 .... x_m, z\}$, $X \in \mathcal{R}^d$.
2: Map multiple features: $\sum_\lambda \beta_\lambda K^\lambda_{i,j})$.
3: Solve Eqn. 2 to obtain α corresponding to Support Vectors (SVs).
4: Calculate bias: ρ
5: Calculate score, $f(\vec{z}) = \sum_{k=1}^{Ns} \alpha_k (\sum_\lambda \beta_\lambda K^\lambda_{\vec{z},\vec{x_k}})$.
6: **if** $f(\vec{z}) > \rho$ **then**
7: return 1
8: **else**
9: return 0
10: **end if**

III. Fleetwide Analysis of Heterogeneous Data

Not only is aviation data extremely large in size, it also has many aspects that create natural sources of heterogeneity. Some examples include flights that have common origin or destination airports, city pair routes, tail numbers, aircraft models, as well as seasonal aspects such as flights within a month. Even within a flight there exist several phases such as take offs, landings and cruise. In aviation safety data heterogeneity may result from the presence of multiple attributes where the attributes do not belong to the same data type. For example the attributes can be either continuous or discrete or it may be a mixture of both. Another source of heterogeneity can be the behavioral or functional properties of these attributes. Since there exists a standard operating procedures for flying the aircraft, the sequence of the discrete pilot inputs along with the measured quantities or parameters are extremely meaningful. It is important to note that in order to enable knowledge discovery, algorithms in general require an integrated and merged view of the data available across various resources. Besides Orca, the rest of the algorithms described in the previous section have been explored either in the context of continuous real-valued data attributes or within discrete domains. However Orca in its current form is not well suited for the sequential anomaly detection task where considering the ordered information of the data instances are extremely important in the analysis. Having said that, with the increasing number of data sources, there is a need to develop intelligent knowledge refinement and integration techniques, focusing on the descriptions of underlying heterogeneous data sources.

This leads us to "kernelized" methods such as one-class SVMs where we can encode the knowledge about the data, expressed in terms of pairwise similarities. This provides us with the opportunity to incorporate vast amounts of knowledge from heterogeneous sources using "appropriate" kernel functions. This field of research is known as Multiple Kernel Learning (MKL) [4], [9], [13]. MKL takes advantage of the mathematics of kernels allowing us to derive new kernels from existing kernels, provided each kernel satisfies the "Mercer condition" which states that the kernel function must be continuous, symmetric, and positive definite. There are several classes of kernel which coincide with the Mercer kernel. Interested readers can explore literatures [8], [10], [12], [19], [22] that look into various other classes of kernel like RBF, polynomial, bag-of-word, sigmoid, spline, graph based, tree based, mismatch based functions etc. A common practice is to use a convex combination (i.e. $\sum_\lambda \beta_\lambda = 1$) of various kernels (Eqn. 2) which may be constructed on very different feature sets.

When analyzing FOQA data the concept of a system level analysis is paramount. The flight data consists of many parameters that monitor the various subsystems within an aircraft. Given a single flight this is not a simple task, and when considering additional flights and multiple aircraft the task can quickly grow beyond the classical timeseries analysis problem. With this increased complexity it is important to understand the hierarchical

system structure and design algorithms to address this paradigm. Another challenging aspect is the sheer size of the data that must be consumed by the algorithm. Typically flights are recorded at 1 Hz and may last anywhere from a couple of hours for some regional flights and up to 10 to 19 hours for some international flights. This is compounded by the number of parameters that are recorded, which is typically hundreds of measurements. Some airlines are flying thousands of flights a day and millions of flights a year, which can add up to terabytes of data very quickly. To address this in the system level approach the algorithm must treat the flights as a fleet of aircraft with some sort of intelligent way of compressing the important features of each flight, and identifying the anomalies at the flight level. We will elaborate more on this in the following section.

IV. FOQA Data Analysis

The real world data set chosen for analysis is from a U.S. regional carrier. All aircraft analyzed were of the same fleet and type (narrow body jet), over a one year period resulting in over 176,000 flights. Each flight consists of 160 parameters sampled at 1 Hz with the average flight length between 1.5 and 2 hours. Due to privacy reasons, each pilot's identity and the exact date of the flight is kept confidential by the airline industry.

A. Data Preparation

Data analysis was focused on the approach portion of the flight from 10,000 ft. Mean Sea Level (MSL) to landing, using the deployment of the thrust reversers as a means to determine touchdown. Flights that were not found to reach 10,000 ft. or did not have thrust reversers deployed were removed from the data sets. For parameter selection a domain expert provided a list of 26 relevant continuous parameters that were extracted for analysis. Using information from the domain expert in conjunction with the statistics from the data, the flap parameter, which is categorical in nature, was decomposed into 3 binary state variables and then combined with landing gear and ground spoilers for sequence analysis.

The working data set consists of approximately 174,000 flights with varying lengths. Each flight is represented by a multidimensional heterogeneous time series. A random set of 2048 flights was chosen for training and the remaining were used for testing. For continuous data, the mean and standard deviation are calculated for each parameter across all training flights. These statistics are then used in both training and testing to z-score normalize each parameter across flights to maintain consistency. Once the continuous parameters are normalized they are quantized over a window length and in amplitude to convert them into a SAX representation (details can be found in [9][1]). The discrete parameters are handled by marking the on and off transitions between switch states with unique symbols and concatenating the symbols, while preserving the time ordering, into a sequence vector.

B. Experimental Details

In the Multiple Kernel Anomaly Detection (MKAD) algorithm [9], since we want to model switching sequences for a given process and where the order of the switching is important, normalized Longest Common Subsequence (nLCS) based kernel was chosen as a potential candidate. Given two sequences $\vec{x}_i$ and $\vec{x}_j$, if z denotes a subsequence of them it means that removing some symbols from $\vec{x}_i$ produces $\vec{x}_j$ or vice versa. The longest such subsequence of $\vec{x}_i$ and $\vec{x}_j$ is called the longest common subsequence (LCS) and is denoted by $LCS(\vec{x}_i, \vec{x}_j)$ and $|LCS(\vec{x}_i, \vec{x}_j)|$ is its length. Such a kernel over discrete sequences, when normalized, takes the form of,

$$k(\vec{x}_i, \vec{x}_j) = nLCS(\vec{x}_i, \vec{x}_j) = \frac{|LCS(\vec{x}_i, \vec{x}_j)|}{\sqrt{l_{\vec{x}_i} l_{\vec{x}_j}}}, \quad (4)$$

where $l_{\vec{x}}$ is the number of symbols in sequence $\vec{x}$. Each sequence of switches is compared against other sequences by using the longest common subsequence (LCS) as the metric for comparison. Sequences that are similar are bound to hold high nLCS values, while dissimilar sequences will hold very low nLCS values. For the MKAD algorithm, once the sequences are generated the discrete kernel is computed pairwise across all possible flight combinations in the training set. For the continuous data, each time series is SAX transformed. In the original version of SAX, the z-score normalization is an integral part of the algorithm. However in this research, we normalized each time series (only once) before it is SAX transformed. We are able to maintain consistency in choosing the alphabet size for both reference and test sets. The window size was also kept fixed throughout the analysis. The window size was set to 30 sec. We assumed that any changes smaller than the window size is not significant for our analysis. The alphabet size was set to 10, which is typically based on the resolution the user wants to achieve. Once the SAX representations are obtained, another kernel is computed pairwise across all possible flight combinations. Each element of this kernel is the average of the pairwise

[1]The source code of SAX can be obtained from the authors' website at http://www.cs.ucr.edu/ eamonn/SAX.htm.

comparison across the parameters of any two flights. In the optimization, we have set the ν parameter of one-class SVMs to 0.1. For testing, the support vectors are used to calculate the pairwise similarity between all testing flights. The discrete and continuous kernels for test data were generated in a similar fashion as the training.

C. Flight Analysis

Out of 174,000 flights the algorithm identified over 4,700 flights as anomalous. This paper will present two flights identified by a domain expert to be operationally significant.

1) High Energy Approach: The first flight can be categorized as a high energy approach. There are basically two conditions which result in a high energy approach - the aircraft may be too high or too fast or both. Each of these conditions can be converted into the other in a process called "trading altitude for airspeed" or vice versa. But ultimately, drag devices, such as flight spoilers, gear or flaps must be deployed to permit a return to the ideal flight profile. However, complicating this effort, most of the drag devices have a maximum speed for deployment, limiting corrective options in the "too fast" scenario, and airlines and aircraft have rate of descent limits which affect the "too high" situation. Finally, Air Traffic Control (ATC) is responsible for traffic separation, which may limit the maneuvering needed to dissipate excess energy. The flight shown in figures 3 & 4 managed to encounter every one of these issues. Techniques were employed, rather competently, which enabled the aircraft to "go down and slow down", a supposed dichotomy. In order to dissipate energy, the pilot lowered the landing gear just below the maximum allowable airspeed for that operation, and prior to any flap extension (see fig 3). The algorithm found this anomaly. But this was insufficient to permit a return to an ideal flight profile, so the pilot presumably obtained an ATC clearance to make a 360 degree turn (see Fig. 4) Due to considerations of conflicting traffic this is always something which must be coordinated with ATC, and sometimes a simple 360 is not an option, resulting in the performance of a go around maneuver. 360 degree turns on final approach, while not unheard of, are rare, so the algorithm found this anomaly as well.

It turns out that the 360 degree turn was the last in a whole series of energy dissipating steps taken by the pilot. Prior to the turn, the aircraft was both too high and too fast to make a stabilized approach, so drag devices, in this case flaps and gear were deployed. The landing gear had provided some drag, but the aircraft was still too fast to deploy flaps beyond an initial setting, so the pilot did something entirely counterintuitive when too high - he climbed for about 25 seconds, bleeding off sufficient airspeed to permit him to extend the flaps further, which in turn enabled him to descend more rapidly (fig 3 again), with the goal of enabling him to lose sufficient altitude to return to the approach in a much better position for a stabilized approach. When even this was not enough, he made the turn.

Climbing on approach to dissipate energy is an unusual but not unheard of maneuver. It usually results in in the aircraft being above the ideal approach profile, necessitating a high rate of descent at a time when stable approach procedures aim to minimize such diversions. Much research has shown such anomalies to result in an increased number of long or hard landings, definite safety issues. The comparison of potential and kinetic energy between aircraft is problematic, due at least in part to the number of variables involved. But an entirely new concept - the rate of change of energy - may allow comparisons between flights, and provide fertile ground for future research.

2) Turbulent Approach: The second flight falls under the category of a turbulent approach. The amount of lift supplied by an aircraft wing is a function of the speed of the air flowing across it. In turbulent conditions, it is possible to experience enough loss of lift that one or both wings stall, possibly causing roll or pitch changes, which in extreme cases could be dangerous. All of this has been well known since the earliest days of flight, but recent crashes in the South Atlantic and in upper New York State remind us that despite advanced aircraft design and current operational procedures, the issue of turbulence, especially clear air turbulence, remains a threat. Airline flight data monitoring programs watch for excessive aircraft attitudes and speeds, but generally downplay environmental factors like wind speed and direction, as well as throttle position and engine speed (N1). This flight descended from about 5,000 ft to landing on a fairly straight path, but the pilot had to work hard to accomplish this (see figure 5(a)).

MKAD discovered atypical fluctuations in engine speed, which is represented by an RPM parameter called N1. Engine speed is normally fairly consistent during a stabilized approach, increasing a bit each time flaps are extended, to make up for the increased drag. On this flight, N1 varied between 35% and 80% (maxium being 100%) in more than 7 cycles see figure 5(b). This is a considerable amount of engine speed variation.

After some analysis, the reason for all this variation was that the pilot had to cope with gusty wind conditions which caused the airspeed to vary between 140 - 180

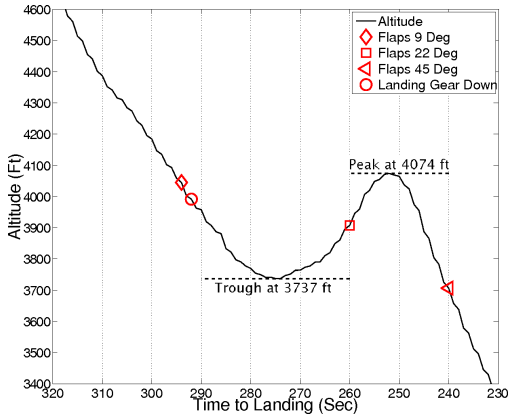


Fig. 3. Altitude plot showing when the drag devices were deployed.

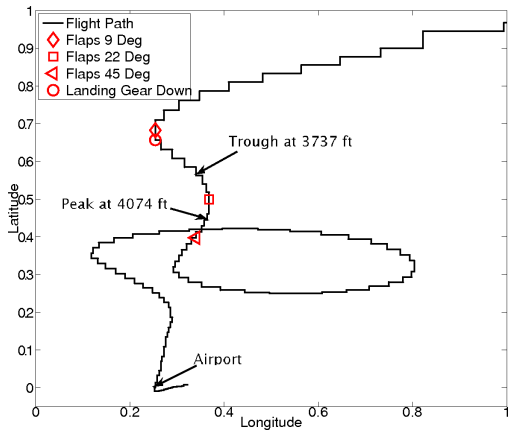
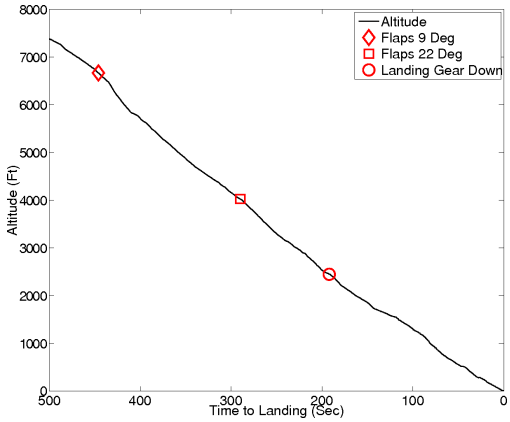
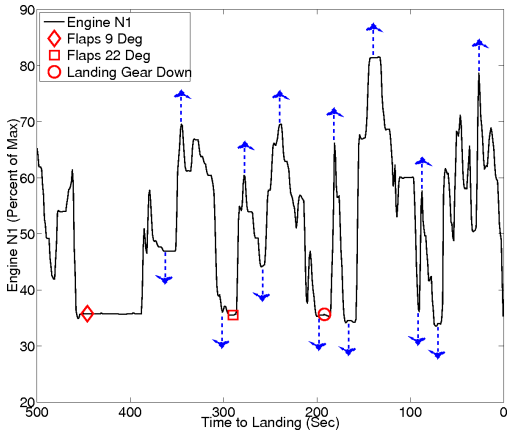


Fig. 4. Flight path plot showing when flaps and gear were deployed. The arrows indicate location of the altitude peak and trough.



(a) Altitude



(b) Engines N1 Speed

Fig. 5. Figure 5(a) shows straight altitude approach profile. The arrows in figure 5(b) are shown to help visualize the frequent increase and decrease in Engine Speed.

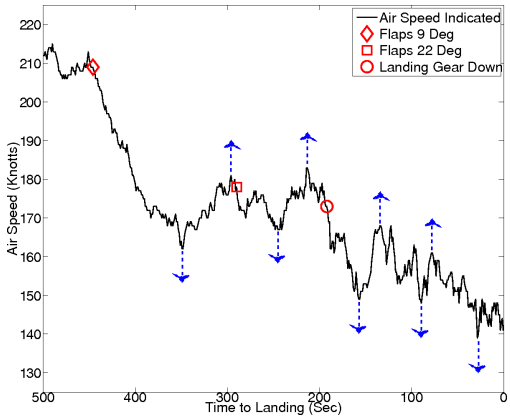
knots. Part of the reason for this variation, of course, was the extension of drag devices (gear and flaps), but an examination of the airspeed trace seen in figure 6(a) shows about 4 cycles of acceleration and deceleration. The headwind over this same time segment varied between 23 - 28 knots in about 4 cycles (see figure 6(b)), not dangerous, but certainly not ideal.

It is important to note that the anomalies identified by MKAD were determined to have contributions from both the discrete and continuous parameters. In the high energy approach the landing gear was statistically out of order from the normal approaches in the data set and when combined with the unusual altitude profile signaled an anomaly. Additionally the preprocessing steps used to discretize the continuous parameters preserved the sequential nature of the data allowing the kernel function to highlight an unusual parameter profile such as the throttle changes in the turbulent approach flight. This is different from other algorithms that treat each time sample independently which hinders them from detecting events in this way.

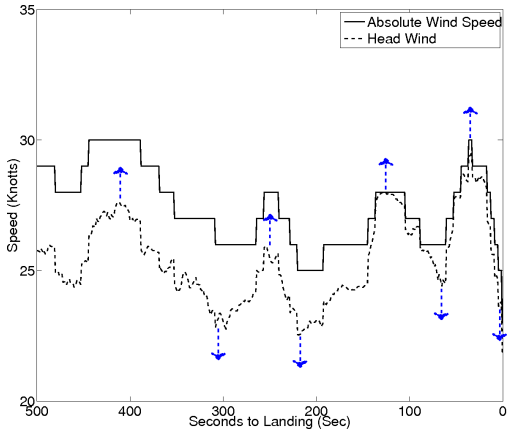
V. Conclusion

Parameter anomalies discovered by MKAD are frequently like little windows into a larger reality. It will frequently require analysis by a domain expert to ferret out whether an anomaly stemmed from a hazardous issue, or whether the condition, while atypical, was safe and explainable. One thing is certain - while a large amount of experience and expertise underlie the current aviation data analysis programs, they only answer questions that someone thought to ask. There will always be value in the search for the unexpected, which is the purpose of MKAD.

An interesting observation is that detection techniques which involve distance related functions or similarities



(a) Air Speed



(b) Wind Speed

Fig. 6. The arrows in figure 6(a) & 6(b) are shown to help visualize the frequent increase and decrease in AirSpeed and Wind speed.

between elements of the set somewhere in the algorithm, are more versatile to address different data structures. For example the MKAD technique described above can easily be extended to include unstructured data (text), if available. Choosing the right *metric function* and integrating knowledge from multiple sources must be done judiciously.

ACKNOWLEDGMENT

This project was supported by the NASA Aviation Safety Program, System-Wide Safety and Assurance Technologies (SSAT) Project.

REFERENCES

[1] Amidan, B.G., Swickard, A.R., Allen, R.E. and Ferryman T. A. Identifying In-Close-Approach-Changes in Air Traffic Control (ATC) Data. 2002. PNWD-3210. Battelle Pacific Northwest National Laboratory: Richland, WA.

[2] F. Angiulli and F. Fassetti. Very Efficient Mining of Distance-based Outliers. In *Proceedings of CIKM '07*, pages 791–800, 2007.

[3] F. Angiulli and C. Pizzuti. Outlier Mining in Large High-Dimensional Data Sets. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):203–215, 2005.

[4] F.R. Bach, G.R.G. Lanckriet, and M.I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *International Conference on Machine Learning*, 2004.

[5] S. D. Bay and M. Schwabacher. Mining Distance-based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule. In *Proceedings of KDD'03*, pages 29–38, 2003.

[6] Suratna Budalakoti, Ashok N. Srivastava, and Matthew E. Otey. Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 39(1):101–113, 2008.

[7] Banerjee A. Chandola V. and Kumar V. Anomaly detection: A survey. *ACM Comput. Surv.*, 41:1–58, 2009.

[8] O. Chapelle and P. Haffner. Support vector machines for histogram-based classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999.

[9] Santanu Das, Bryan L. Matthews, Ashok N. Srivastava, and Nikunj C. Oza. Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 47–56, New York, NY, USA, 2010. ACM.

[10] Z. Harchaoui and F.R. Bach. Image classification with segmentation graph kernels. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[11] David L. Iverson. Inductive system health monitoring. *Proceedings of the 2004 International Conference on Artificial Intelligence (IC-AI'04), CSREA Press, Las Vegas, NV*, 2004.

[12] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Kernels for graphs. *Kernel Methods in Computational Biology*, 39(1):101–113, 2004.

[13] G.R.G. Lanckriet, N. Cristianini, L. El Ghaoui, P. Bartlett, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[14] Markos Markou and Sameer Singh. Novelty detection: A review - part 1: Statistical approaches. *Signal Processing*, 83:2003, 2003.

[15] Markos Markou and Sameer Singh. Novelty detection: A review - part 2: Neural network based approaches. *Signal Processing*, 83:2499–2521, 2003.

[16] John A. Quinn and Christopher K. I. Williams. Known unknowns: Novelty detection in condition monitoring. In *Proceedings of 3 rd Iberian Conference on Pattern Recognition and Analysis, Lecture Notes in Computer Science 4477. Springer-Verlag*, pages 1–6. Springer LNCS, 2007.

[17] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. *SIGMOD Rec.*, 29(2):427–438, 2000.

[18] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural Comput.*, 13(7):1443–1471, 2001.

[19] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[20] Thomas R. Chidester. Understanding Normal and Atypical Operations through Analysis of Flight Data. *In Proceedings of the 12th International Symposium on Aviation Psychology*, 2003. Dayton, Ohio.

[21] Wikipedia. Hamming distance, the free encyclopedia, 2010. [Online; accessed 04-December-2010].

[22] H. Zhang, A.C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition*, pages 2126–2136, 2006.