A Hybrid Approach for Fault Detection and Diagnosis in Autonamous Systems

Eliahu Khlastchi and **Meir Kalech** and **Lior Rokach**
Ben-Gurion University of the Negev, Beer-Sheva, Israel
email: {khalastc, kalech, liorrk}@bgu.ac.il

Abstract

One of the challenges of fault detection in the domain of autonomous systems is the handling of unlabeled data, meaning, most data sets are not recognized as normal or faulty. This fact makes it very challenging to be used as a training set such that learning algorithms would produce a successful fault detection model. Traditionally unsupervised algorithms try to address this challenge. In this paper we present a hybrid approach that combines unsupervised and supervised methods. An unsupervised approach is utilized for labeling a training set, and then by a standard supervised algorithm we build a fault detection and diagnosis model that is much more accurate. We show promising results on simulated and real world domains.

1. Introduction

Autonomous systems such as Unmanned Vehicles (UVs) or robots are susceptible to a variety of hardware and software faults. These faults might lead to mission failure or even endanger the safety of the expensive system or its environment. For example, a pitot-static system failure in an Unmanned Aerial Vehicle (UAV) might lead to a crash.

To continue operate autonomously, the system must have an accurate fault detection mechanism. Upon fault detection a diagnosis process can be triggered and a decision on how to continue can be made. An accurate fault detection mechanism presents several challenges in the domain of autonomous systems: (1) there is a great variety of faults, such as stuck value, drift or abrapt, (2) a fault expression can span over time, (3) a fault should be detected as quickly as possible, online, and with high accuracy, (4) a fault detection mechanism should be kept light, finally (5) this domain is also characterized for having unlabeled data.

The Diagnosis should be accurate as well. A diagnosis report should return a minimal set of components that includes the root cause of the fault.

Three general approaches are usually used for fault detection and diagnosis: Knowledge-Based systems, Model-Based, and Data Driven approaches [Isermann 2005]. Knowledge based systems typically associates recognized behaviors with predefined known faults and hence, are less likely to detect an unknown fault. Model-based approaches are potentially very accurate. The correct behavior of each component is modeled analytically. The system output is compared to the modeled output and a high residual indicates a fault. The diagnosis can be deduced from the information the model provided. In complex autonomous systems, the task of modeling the behavior of components is very hard or even impossible. Data driven approaches are model free. The online data is usually used to statistically differentiate a fault from a normal behavior. However, this task may be not as light as the system requires.

In our previous work [Khalastchi *et al.*, 2013] we presented an unsupervised approach for fault detection in the domain of autonomous systems. This approach combines model based and data driven approaches, and shows a high rate of detection and a low rate of false positives. In this paper we aim to show a hybrid approach that uses an accurate **unsupervised** approach for fault detection similar to our previous approach, to create more accurate and light fault detection and diagnosis model in a **supervised** manner.

Empirical evaluation on simulated and real word domains show that the learnt fault detection model is more accurate than the original unsupervised approach. Finally, we argue that this hybrid approach can be generalized to a classification problem with an unlabeled training set.

2. Related work

Steinbauer conducted a survey on the nature of faults of autonomous robot systems [Steinbauer 2011]. The survey participants are developers competing in different leagues of the Robocup competition [Robocop]. The reported faults were categorized as hardware, software, algorithmic and interaction related faults. The survey concludes that hardware faults have a high negative impact on mission success. In this paper we focus on detecting such faults.

We presented an unsupervised approach for sensor fault detection that combines model-based and data driven techniques to achieve greater accuracy [Khalastchi *et al*, 2013]. The hybrid approach presented in this paper utilizes this unsupervised approach.

The chosen unsupervised approach has a high detection rate for single dimension faults such as "drift" and "stuck". These types of faults appear in a variety of related domains. For example, the Advanced Diagnostics and Prognostics Testbed [ADAPT] depicts the following faults to sensors on an electrical circuit. This testbed is used for the diagnosis competition [DXC,2011]. Another example is the work of [Hashimoto *et al.*, 2005] that uses kalman filters along with kinematical models to detect sensor faults such as "stuck", "abrupt" and "scale" on a mobile robot. Our hybrid approach relies on a function that returns the state of the sensor (i.e. abrupt, drift, stuck, etc.).

[Leeke *et al,* 2011] present a methodology for generating efficient error detection mechanisms. Their approach relies on injecting faults into data that is used as a training set for a learning algorithm. The learnt error detection model is of high accuracy. Our hybrid approach is similar in concept, but does not rely on fault injection.

3. Decreasing the False Positive Rate

In this section we describe the problem of fault detection in the domain of unmanned vehicles. We continue with demonstrating how the unsupervised labeling is done. Finally, we describe the learning process.

3.1 Problem Description

Let $A = \{a_1 \dots a_n\}$ be a set of attributes that are monitored in real time (e.g. air-speed, heading, pitch, altimeter, etc.) and let $V_t = \{v_1 \dots v_n\}$ be the set of values for attributes $\{a_1 \dots a_n\}$ at time t where $v_i \in \mathbb{R}$ is the value assigned to a_i at time t. Past data of m time units of these values $H_m(t) = (V_{t-m}, V_{t-m+1}, V_{t-m+2}, \dots, V_t)$ is also available. $H_m(t)$ is a sliding window containing at time t the latest m values of the monitored attributes. In addition, unlabeled past recordings of the unmanned vehicle operations are also available. These recordings can be used as a training set for a machine learning algorithm if labeled. We denote this training set as $\mathcal{H} = \{H_{l_1}(e_1), H_{l_2}(e_2), \dots, H_{l_k}(e_k)\}$ where l_i is the length of operation i and $e_i$is the end time operation i, thus $H_{l_i}(e_i)$ denotes all the values recorded for the monitored attributes in A during operation i.

Given $A, V_t, H_{m(t)}$and $\mathcal{H}$, the goal is to online recognize whether a fault has occurred to any of the attributes in A. This decision should be made as quick as possible after a fault has occurred, and should be as accurate as possible. By 'accurate' we mean that a fault detector should have a high detection rate and a low false positive rate. In addition, a minimal set of diagnosis that includes the root cause of the fault should be returned.

3.2 The Outline of the Approach

We introduce a hybrid approach which consists of an **offline** preprocess and an online fault detection and diagnosis process. The offline preprocess conducts an **unsupervised** algorithm to label an unlabeled training set. Then, a **supervised** learning algorithm is used to construct a fault detection and diagnosis model that can be used **online** with greater accuracy. Figure 1 depicts the outline of our approach.

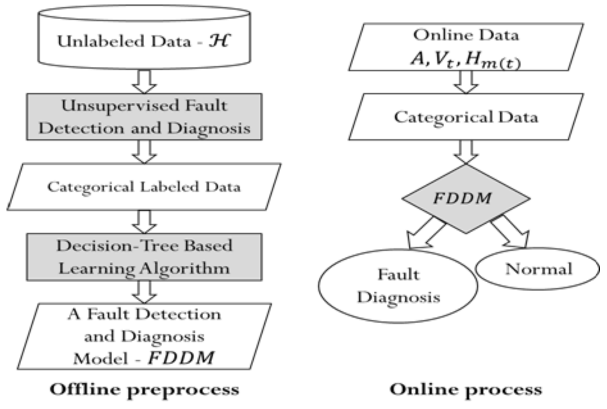
Unlabeled Data - $\mathcal{H}$
Unsupervised Fault Detection and Diagnosis
Categorical Labeled Data
Decision-Tree Based Learning Algorithm
A Fault Detection and Diagnosis Model - $FDDM$
Offline preprocess
Online Data $A, V_t, H_{m(t)}$
Categorical Data
$FDDM$
Fault Diagnosis
Normal
Online process

Figure 1: The Outline of the Approach

3.2.1 The offline preprocess

The unlabeled past operations recordings $\mathcal{H}$are delivered as an input to an unsupervised fault detection algorithm. This algorithm is described in section 3.3. In summary this algorithm translates the behavior of each attribute in each sliding window to normal or to one of pre-defined suspicious patterns (e.g. stuck, drift). In addition, each sliding window is labeled according to the unsupervised classification (i.e. "*Normal*", or the diagnosis of the fault). However, a small portion of the normal examples may be misclassified as faults; these are the false positives of the unsupervised fault detection. The resulted labeled data is fed into a decision-tree based leaning algorithm. The resulted fault detection and diagnosis model – $FDDM$ can be used online, and is more accurate than the original unsupervised fault detection and diagnosis approach as shown in the results section.

3.2.2 The online fault detection process

The online input is a time series confined into a sliding window $H_{m(t)}$. With each time step t the data in $H_{m(t)}$ is transformed into a categorical data which can be fed into the fault detection and diagnosis model - $FDDM$ (resulted by the offline preprocess). The fault detection and diagnosis model classifies the data presented in $H_{m(t)}$ as *normal* or as faulty by returning the fault diagnosis as a classification. In section 3.5 we describe the fault detection and diagnosis online process in detail.

3.3 Unsupervised Labeling

Each past operation of the autonomous system $H_{l_i}(e_i) \in \mathcal{H}$is fed into an unsupervised diagnosis engine. This diagnosis engine uses a heuristic decision which is based on a prior knowledge of a structural model as well as the online consumed data $H_{m(t)}$ (sliding window) to determine an occurrence of a fault and its diagnosis. We chose this approach due to its very low rate of false positives and the high fault detection rate which usually is 1 or very close to 1. Note that any unsupervised fault detection and diagnosis approach can be used to label the unlabeled training set.

Being online and unsupervised, the approach relies on correlated attributes to provide the necessary comparison between expected and unexpected behavior of attributes. The approach assumes that correlated attributes behave as redundant to one another. This is usually the case for the domain of unmanned vehicles with a rich array of sensors and modeled variables.

Each attribute is subjected to tests by suspicious pattern recognizers. A suspicious pattern recognizer is a part of the fault detection system input and is domain specific. The latest m values of an attribute extracted from $H_{m(t)}$ are tested. For example, we use a *drift* test and a *stuck* test. However, when an attribute shows a suspicious pattern it does not necessarily suggest a fault; it could be a reaction to a normal action of the unmanned vehicle. For example, maintaining altitude may appear as *stuck*, and altitude climbing may appear as a *drift*.

To differentiate between a normal reaction and a fault, the approach uses a heuristic decision based on a structural model. A structural model depicts components dependency. The heuristic decision compares an attribute that shows a suspicious pattern with an attribute that used to be correlated to it in the previous sliding window. If they do not share component dependency and show different patterns then this is due to a fault. Otherwise, it might be a reaction to a normal action of the unmanned vehicle.

For example, if the altimeter is suspected for a *drift*, and the GPS indicated altitude is also drifting then this is probably due to the altitude climbing action of the UAV (and not a fault). If the altimeter is *stuck* while the GPS indicated altitude is not, then it is probably due to a fault. Furthermore, these attributes are dependent on different subsystems. This fact makes these attributes less likely to be affected by the same fault.

The data of each sliding window, $\forall t \in (m \dots e_i)$, $H_{m(t)} \subseteq H_{l_i}(e_i) \in \mathcal{H}$, is transformed into a line of categorical data with the addition of the classification made by the unsupervised fault detection algorithm. A recognized suspicious pre-defined pattern is replaced by its class. We use a *drift* and a *stuck* pattern recognizers which correspond to "*drift*" and "*stuck*" categories. An additional "*regular*" category is given to an attribute which is not suspected by any suspicious pattern recognizer.

For example, assume a sliding window of size $m = 4$ containing the values of 3 attributes at a time step t:

Time step	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$
t-4	0.9	1	3
t-3	0.1	2.5	3
t-2	0.05	3	3
t-1	0.15	3.1	3
t	0.05	3.9	3

Assume that the unsupervised diagnosis engine reported a fault for this time step including diagnosis components $\{c_3, c_{15}\} \subseteq \{c_1, \dots c_{40}\}$. The data of the sliding window will be registered as one training sample, where a_1 is categorized as "*regular*", a_2 is categorized as "*drift*", and a_3 is categorized as "*stuck*". In addition, this sample is classified by the diagnosis of the fault - $\{c_3, c_{15}\}$. Thus, the registered training sample for this time step is: *Regular,Drift,Stuck,* $\{\boldsymbol{c_3}, \boldsymbol{c_{15}}\}$. Note that the fact that this sample contains *drift* and *stuck* does not necessarily entail a fault. In some cases it may indicate a normal behavior.

The "Fault" classification (e.g. $\{c_3, c_{15}\}$) may be correct or not; it depends on the accuracy level of the unsupervised approach. The next section will discuss how this inaccuracy affects the constructed decision tree.

3.4 A Decision Tree Based Fault Detection Model

3.4.1 The effect of falsely classified training examples

Decision trees have some tolerance towards falsely labeled examples in their training set. During the model construction, the algorithm grows a decision tree. In each step the algorithm computes the information gain obtained by selecting each attribute and chooses the one with the highest information gain ($IG(a_j)$). The information gain is determined by the entropy of the attribute which is affected by the ratio between the normal and faulty samples. The lower faulty samples the higher information gain. If a portion of samples are misclassified as "fault" then this might reduce the information gain of a node. However, if the degree of reduction is small enough the construction of the tree is unaffected.

The degree of information gain reduction due to falsely classified training examples is dependent on several factors. Let

- S be the training set.
- α_i be $|S_{a=v_i \wedge Class=fault}|$ the size of examples of which attribute a has the value v_i and the classification is "Fault" (i.e. some diagnosis classification of a fault) when all examples are classified correctly by an oracle.
- β_i be $|S_{a=v_i \wedge Class=normal}|$ the size of examples of which attribute a has the value v_i and the classification is "*Normal*" when all examples are classified correctly by an oracle.
- x_i be the number of samples from $|S_{a=v_i}|$, that were falsely classified as "Fault" due to false positives of the unsupervised approach (note that $|S_{a=v_i}| = \alpha_i + \beta_i$).

The effect of the misclassified x_i samples on the information gain of attribute a is the new (affected) information gain $IG_x(a)$ minus the original information gain $IG(a)$:

$$f(x) = IG_x(a) - IG(a)$$

$$f(x) = H(S) - \sum_{v_i} \frac{|S_{a=v_i}|}{|S|} H_x(S_{a=v_i}) - H(S) + \sum_{v_i} \frac{|S_{a=v_i}|}{|S|} H(S_{a=v_i})$$

Where H is the entropy function and H_x is the entropy affected by falsely classified examples. For given x_i falsely classified samples in $S_{a=v_i}$ the affected entropy is:

$H_x(s_{a=v_i}) = -\frac{\alpha_i + x_i}{\alpha_i + \beta_i} \log \frac{\alpha_i + x_i}{\alpha_i + \beta_i} - \frac{\beta_i - x_i}{\alpha_i + \beta_i} \log \frac{\beta_i - x_i}{\alpha_i + \beta_i}$. Note that x_i samples that are falsely added to α_i are in the expense of β_i. After some algebra $f(x_i)$ can be presented for given x_i falsely classified examples in $S_{a=v_i}$ as:

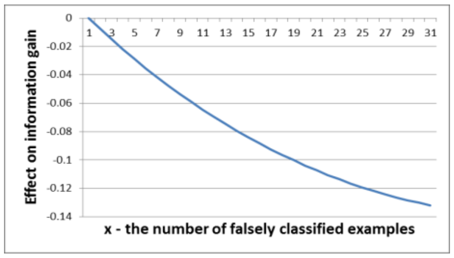

Figure 2: The negative effect on

$$f(x_i) = \frac{1}{|S|}\left(\alpha_i \log\frac{\alpha_i + x_i}{\alpha_i} + \beta_i \log\frac{\beta_i - x_i}{\beta_i} + x_i \log\frac{\alpha_i + x_i}{\beta_i - x_i}\right)$$

We can see that when $x_i = 0$ then $f(x_i) = 0$. When x_i grows, $f(x_i)$ decreases. This affect is depicted in Figure 2. Figure 2 illustrates an example for this effect on the information gain where $|S| = 300, \alpha_i = 20, \beta_i = 100$.

As x_i grows, the information gain decreases. The negative effect will eventually return to 0 as x_i grows. When $\alpha_i + x_i = \beta_i$ the number of samples for each classification is the same as the original, only with opposite classifications; the entropy is the same as the original entropy and thus the information gain is unaffected.

If the information gain of some attribute $IG(a_i)$ is greater than the information gain of another attribute $IG(a_j)$ when all training set examples are classified correctly by an oracle, then for a small enough number of falsely classified examples x the effect of $f(x)$ will not change the decision tree construction $IG_x(a_i) = IG(a_i) + f(x) > IG(a_j)$.

In some cases the number of falsely classified examples x will affect the decision tree construction $IG(a_i) + f(x) < IG(a_j)$. This might lead to the appearance of false positives when using the resulted fault detection and diagnosis model. When compared to a model that was constructed of correctly classified training examples this approach has more false positives due to the effect of $f(x)$. However, this approach has significant less false positives than the original unsupervised approach, as we show in the results section.

3.4.2 Using the *FDDM* online

When given a new online input, we consume it in a sliding window fashion $H_m(t)$. We transform the time-series data in $H_m(t)$ to a line of categorical data in the same manner we used in the original unsupervised approach. Only when the new line is different than the previous line (i.e. some attribute changed its state) the new line is fed into the offline-learnt fault detection and diagnosis model - $FDDM$. The $FDDM$ decides (online) whether or not it is a fault. In case of a fault the $FDDM$ returns a diagnosis as the classification.

For example, consider a static-system failure. One of the expressions of this failure is the frozen value of the altimeter. In $H_m(t)$ the values of the altimeter attribute are all equal, while the values of the GPS indicated altitude attribute diverse. The equal values of the altimeter are recognized as a suspicious pattern by the stuck-pattern detector. Therefore, the corresponding categorical line to $H_m(t)$ has the value "*stuck*" for the altimeter attribute and the value "*regular*" for the GPS indicated altitude attribute (other attributes also get their own categorical values) . This line is fed into the $FDDM$ that decides whether or not these values express a fault. It is possible that the next categorical line that corresponds to $H_m(t+1)$ will not be different than its predecessor. In this case, the $FDDM$ is not triggered again; only if at least one of the attributes changed its state (e.g. from regular to stuck as the altimeter) then the $FDDM$is triggered.

To summarize our hybrid approach, in the first stage we use an unsupervised fault detection and diagnosis algorithm to label unlabeled training set. The suspicious pattern detectors of the unsupervised approach are used to transform the time-series data in each sliding window into categorical data. Each sample is classified according to the unsupervised decision (i.e. "*Normal*" or other fault modes). Then, we apply a decision tree based learning algorithm on the training data and produce a fault detection and diagnosis model - $FDDM$ that can be applied online. The online process uses the same suspicious pattern detectors to produce a categorical line for each sliding window. The line is fed into the $FDDM$ which was learnt offline. The $FDDM$ makes a choice whether or not the online input is an expression of a fault. In case of a fault, the classification is the diagnosis.

4. Experiment Setup

To examine our approach we tested the fault detection accuracy as well as the diagnosis accuracy. We use three domains to test the fault detection accuracy. The first is a high fidelity flight simulator [FlightGear] the second is a commercial UAV, and the third is a laboratory robot *Robtican1* [Robotican] (see Figure 3). In addition, we use the FlightGear domain to test the diagnosis accuracy. We expect our proposed hybrid approach to be more accurate in fault detection and diagnosis than the original unsupervised approach.

Flighgear domain: *Flighgear* is an open source flight simulator designed for research purpose and is used for a variety of research topics. FlightGear has built-in realistically simulated instrumental and system faults. For example, if the vacuum system fails, the HSI gyros spin down slowly with a corresponding degradation in response as well as a slowly increasing bias/error.

We recorded 32 flights. Each flight had duration of 5 minutes, and included a take-off and left and right turns. 23 attributes were sampled in 4Hz. Each flight was injected with a different type of fault. Each fault had duration of 35 seconds and was injected twice to the same flight at random times. In total, we tested 11 different types of instrumental and system failures. In total, the test set contains 25,977 examples out of which 5,880 are expressions of faults.

The unsupervised process applied on the training set achieved a detection rate of 1 (all faults were detected). The resulted categorical and classified data was used as a training set.

In addition, due to its richness of data and deep level of components dependencies, we use the FlightGear domain to further test the diagnosis accuracy of the proposed hybrid approach. We tested two types of system failures and 4 types of instrumental failures. In total, 6 flights were used as a training set and 12 flights were used as a testing set. The goal was to accurately detect and diagnose the faulty components out of 40 possible components.

Commercial UAV domain: The real UAV domain consists of 6 recorded real flights of a commercial UAV. 53

attributes were sampled in 10Hz. The attributes consists of telemetry, inertial, engine and servos data. Flights duration varies from 37 to 71 minutes. The UAV manufacture injected a synthetic fault to two of the flights. The first scenario is a value that drifts down to zero. The second scenario is a value that remains frozen (stuck). The detection of these two faults were challenging for the manufacture since in both scenarios the values are in normal range. These two flights were used as a test set. The remaining four flights were used as a training set where into two flights we injected similar synthetic faults. In total, the test set contains 65,741 examples out of which 1,593 are expression of faults.

Laboratory robot domain: *Robotican1* is a laboratory robot that has 2 wheels, 3 sonar range detectors in the front, and 3 infrared range detectors which are located right above the sonars, making the sonars and infrareds redundant systems to one another. This redundancy reflects real world domains such as unmanned vehicles. In addition, the Robotican has 5 degrees of freedom arm. Each joint is held by two electrical engines. These engines provide a sensed reading of the voltage applied by their action.

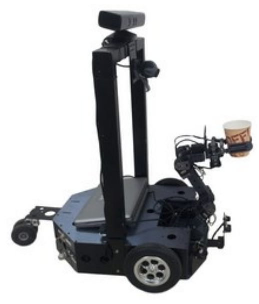
Figure 3: Robotican1

We devised 10 different scenarios that included different injected faults while the robot performed different tasks. Faults were injected to each type of sensor (motor voltage, infrared and sonar). The injected faults to the sensors were of type *stuck* or *drift*. These faults were injected to one or more sensors in different time intervals. 15 attributes were sampled in 8Hz. Scenarios duration lasted 10 seconds where the last 5 seconds expressed a fault. 4 scenarios were used as an unlabeled training set and the other 6 were used as a test set. Note that in this domain, the training set did not cover all the examples included in the test set.

For the supervised learning we have experimented with several decision tree algorithms: ID3, J48, and a Random Tree [Breiman, 2001]. As expected the Random Tree performed better and its results on the three domains are shown in the next section.

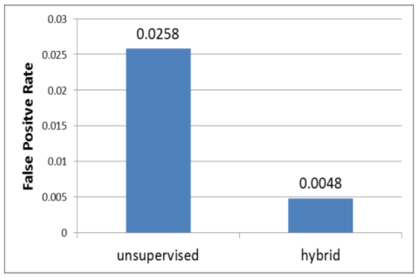

Figure 4: FP rate,unsupervised vs. hybrid

5. Results

Figure 4 illustrates the average false positive rate of the hybrid vs. the original unsupervised approach, taken over the 21 test flights of the FlightGear domain, using a sliding window size of 250. The hybrid approach significantly improved the false positive of the unsupervised algorithm.

To demonstrate the degree of reduction of the false positive rate by the suggested hybrid approach we used different sizes of sliding windows during the offline training phase. Smaller sizes create more opportunities for reports and thus more opportunities for false positives. Figure 5 illustrates the degree of reduction in the average false positive rate over the 21 test flights in the FlightGear domain. Note that the false positive rate is in logarithmic scale. We can see that with each size of sliding window the false positive rate of the hybrid approach is significantly lower than the original unsupervised approach.

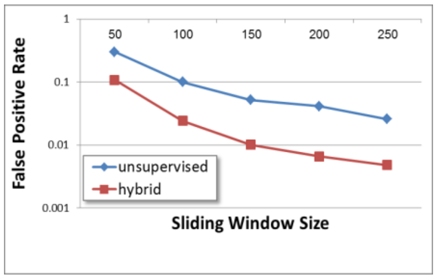

Figure 5: FP rate vs. s.window size

The different parameters used by the unsupervised approach during the offline phase can be viewed as different unsupervised approaches; each with its own rate of false positives. The hybrid approach contributes to the reduction of false positive rate for each of these unsupervised approaches.

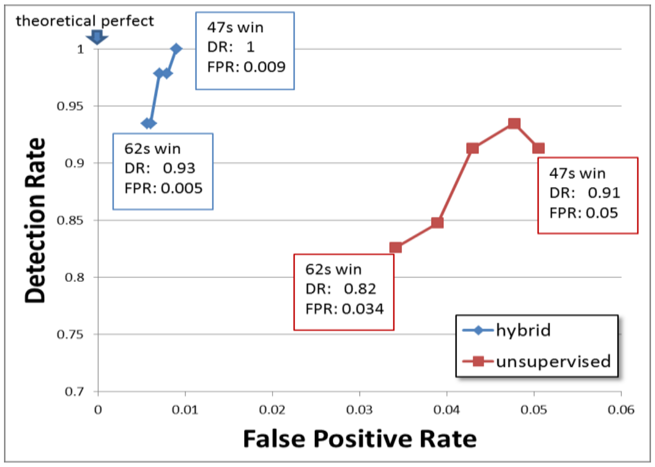

Figure 6: ROC - Hybrid vs. Unsupervised

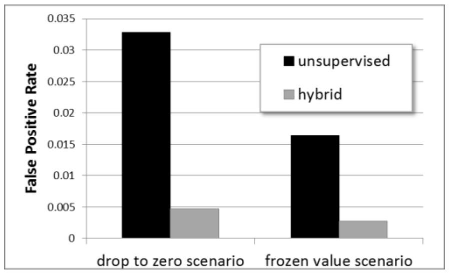

Figure 7: UAV, unsupervised vs. hybrid

Satisfied by the very low rate of false positives, we decreased the sliding window size used **online**. It is suggestible to use a smaller m for $H_m(t)$ when classifying an online input than the m used during the offline training. This increases the number of reports, and since the false alarm rate is very low, we can tolerate an increase of false positives in return for a higher true positives rate.

Figure 6 illustrates the ROC of false alarm rates and the detection rates of the unsupervised approach verses the hybrid approach under the influence of a changing size of the online sliding window (62sec – 47sec). Note that scale of Figure 6 zooms in on high detection rate (close to 1) and low false alarm rate (close to 0). The added of false positives to the hybrid approach is of little significance while the effect on the unsupervised approach is apparent.

In addition, the detection rate of the hybrid approach is getting higher as the size of the sliding window decreases. This is explained by the fact that a smaller size of a sliding window increases the frequency of state changes and hence the total amount of reports. Therefore, there is a greater chance for detection as well as some false positives. The hybrid approach gets a lower rate of false alarms and a higher rate of fault detection than the original unsupervised approach.

In the UAV domain the hybrid approach keeps a similar trend. In the two examined scenarios, both the hybrid and unsupervised approaches had a detection rate of 1. However, the hybrid approach had a significantly lower false positive rate than the unsupervised approach as figure 7 shows.

In the *Robotican1* domain, even though the training set did not include all possible faults that were included in the test set, the detection rate of the learnt fault detection model was 1. Being online and unsupervised, it is not surprising that the unsupervised approach also scored a detection rate of 1 on the test set. However, it is interesting to note that the offline learnt *FDDM* of the hybrid was able to generalize the heuristic decision of the unsupervised approach such that unseen faults were detected.

The average false alarm rate of the unsupervised approach on the 6 tested scenarios was 0.067 while the hybrid approach scored 0.041. Again, the hybrid approach reduced the false positive rate.

We also tested the diagnosis of the proposed approach on the FlightGear domain as table 1 depicts.

The unsupervised approach produced very good results: A detection rate of 1, false alarm rate of 0.0086, and the diagnosis set contained an average of 2.88 components out of 40 possibilities, and always included the single root cause, making the diagnosis false positive rate as 0.048. But still, the hybrid approach is able to improve the results. The hybrid approach got a detection rate of 1, a false alarm rate of 0.0077, and an average diagnosis set size of 2.14 out of 40 possible components that included the single root cause, making the diagnosis false positive rate 0.029.

Table 1:diagnosis results, FlightGear domain

Approach	Fault Detection rate	False alarm rate	Diagnosis true positive rate	Diagnosis false positive rate
Unsupervised	1	0.0086	1	0.048
Hybrid	1	0.0077	1	0.029

6. Discussion

The offline step labels the data with an unsupervised approach. An alternative approach for labeling the data is a clustering algorithm (e.g. K-means where k=2). However, an unsupervised fault detection approach is more specific to the fault detection problem and thus expected to be more accurate than the general clustering algorithm.

We chose to demonstrate the hybrid approach with the use of our previous unsupervised approach [Khalastchi *et al.* 2013] since it showed a high detection rate and a very low false positive rate. Any other highly accurate unsupervised approach could have been used for that matter. The high detection rate is very important since all faults should be labeled as such.

The learnt *FDDM* generalized the original heuristic decision of the unsupervised approach. The model is independent of online correlation calculations and thus is lighter and less susceptible to false positives than the original unsupervised approach. Moreover, The *FDDM* returned less diagnosis candidates than the original unsupervised approach, further isolating the root cause possibilities. Finally, we argue that the hybrid approach could be generalized to any classification problem when the training data is unlabeled.

References

[ADAPT] http://ti.arc.nasa.gov/tech/dash/diagnostics-and-prognostics/adapt-diagnostics/

[Breiman, 2001] Breiman L. "Random forests." *Machine learning 45.1 : 5-32.*

[DXC, 2011] International Diagnostic Competition - website, http://sites.google.com/site/dxcompetition2011/

[FlightGear] website, http://www.flightgear.org/

[Hashimoto, 2005] Hashimoto M. A multi-model based fault detection and diagnosis of internal sensors for mobile robot. *Intelligent Robots and Systems ,pp.3787- 3792.*

[Isermann 2005] Isermann R. Model-based fault-detection and diagnosis—Status and applications. *Annual Reviews in Control, 29(1), 71–85.*

[Khalastchi *et al*, 2013] Khalastchi E., Kalech M., Rokach L. Sensor fault detection and diagnosis for autonomous systems. In proceedings: The Twelfth International Conference on *Autonomous Agents and Multi-Agent Systems.*

[Leeke *et al,* 2011] Leeke M, Saima A, Arshad J, and Sarabjot S.A., A methodology for the generation of efficient error detection mechanisms." In *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on, pp. 25-36.*

[Robocop] Robotcup competition, website, http://www.robocup.org/

[Robotican] website, http://www.robotican.net/

[Steinbauer 2011] Steinbauer G. a survey on the nature of faults of autonomous robot systems. website, http://www.ist.tugraz.at/rfs/index.php/Main_Page