Airplane flight safety using error-tolerant data stream processing

Article *in* IEEE Aerospace and Electronic Systems Magazine · April 2017

DOI: 10.1109/MAES.2017.150242

6 authors, including:

Shigeru Imai
Rensselaer Polytechnic Institute
27 PUBLICATIONS **390** CITATIONS
SEE PROFILE

Erik Blasch
Air Force Research Laboratory
939 PUBLICATIONS **20,090** CITATIONS
SEE PROFILE

Feature Article:

DOI. No. 10.1109/MAES.2017.150242

Airplane Flight Safety Using Error-Tolerant Data Stream Processing

Shigeru Imai, **Rensselaer Polytechnic Institute, Troy, NY, USA**
Erik Blasch, **Air Force Research Laboratory, Rome, NY, USA**
Alessandro Galli, Wennan Zhu, Frederick Lee, Carlos A. Varela, **Rensselaer Polytechnic Institute, Troy, NY, USA**

INTRODUCTION

Ubiquitous sensing is pervasive in society for such applications as biometrics for health care, smart grids for power delivery, and avionics for transportation safety [1]. As society continues to rely ever more on sensors for various applications, there is a need to address the accuracy of sensor readings for health maintenance, signal identification, and control [2]. While there have been advances in information fusion [3] for avionics control [4] and user warnings [5], there is still a need for further research in methods that allow for fault detection and recovery techniques to be easily realized and implemented with minimal risk of software errors.

SENSOR DATA ERRORS

Aircraft sensor errors have been at the root of many flight accidents [6]. Sensor faults inducing erroneous airspeed, altitude, and attitude data have led to pilot inability to correctly fly a plane. In February 1996, Birgenair Flight 301's airspeed indicator failed due to a blocked pitot-static system that gave the indication that airspeed was increasing during climb-out, leading to a stall and subsequent fatal crash [6]. In Dec. 1999, the Korean Air Cargo Flight 8509 crashed after takeoff due to pilot error induced by an unreliable attitude director indicator that resulted in the pilot applying excessive bank angle [6]. In both cases, an air data inertial reference unit could benefit from logical redundancy with independently-measured data. A third example involves a military B-2 plane which resulted in a loss of $1 billion [7]. Caused by bad weather, the three pressure sensors (transducers) were improperly calibrated due to condensation inside the devices. During flight, the sensor errors caused diesel fuel to enter skin-flush air-data sensors that determine the airspeed, altitude, and attitude. Investigation of the accident determined that computers calculated inaccurate aircraft angle of attack and airspeed [8].

The 2009 crash of Air France Flight 447 [9] resulted from iced pitot tubes that led to erroneous air speed data. In this article, we advocate the use of logical redundancy; specifically, the ground speed and estimated wind speed data—which were available from onboard Global Positioning System (GPS) monitors and weather forecasts—could have been used to catch erroneous air speed readings coming from the malfunctioning pitot tubes.

Figure 1 highlights the airspeed indicator components of the Airbus 330 configurations, including the pitot tubes for dynamic pressure measurement, the static port for static pressure measurement, and the instrumented diaphragm that determines airspeed by measuring the difference between these two quantities. To maintain accurate airspeed measurements, it is important to heat the pitot tube at high altitudes or in cases of low temperatures in which moisture would freeze and obstruct airflow.

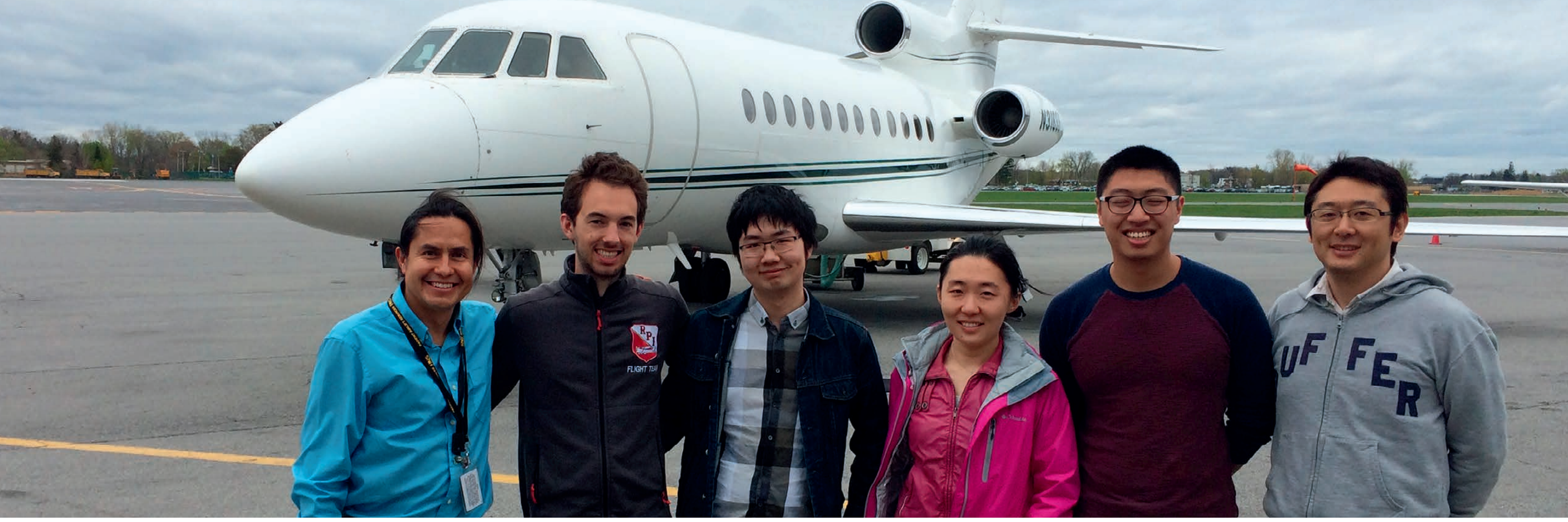
SENSOR FAULT DETECTION

Two important directions of research for safe air transportation are: Data Stream Management Systems (DSMS) and Fault Detection, Isolation, and Reconfiguration (FDIR).

DSMS are prevalent in transportation systems designs, but few studies have been applied to aviation. Available systems such as PLACE [10] and Microsoft StreamInsight [11], which are DSMS-based systems supporting spatio-temporal streams, combine data stream processing and database management. These DSMS-based spatio-temporal stream management systems support general continuous queries for multiple moving objects such as “Find all the planes flying within a sphere diameter of X from a point Y in the past Z time.” [12]. Unlike these DSMS-based systems which handle multiple spatio-temporal objects, the **P**rogramm**I**ng **L**anguage for spati**O**-**T**emporal data **S**treaming applications (PILOTS) [13] assumes each application to be running on a single platform and tries to extrapolate data that is relevant to its current location and time. The PILOTS approach for spatio-temporal data stream filtering, and error detection and correction, uses high-level abstractions, so that users can more easily design error models to correct data in flight.

FDIR has been intensively studied in the control and aerospace communities [14]. Mission critical systems, such as nuclear power plants, flight control systems, and automotive systems, are key ap-

Authors' current addresses: S. Imai, A. Galli, W. Zhu, F. Lee, C. A. Varela, Rensselaer Polytechnic Institute, New York, 110 8th Street, Troy, NY 12180, USA. E. Blasch, Air Force Research Lab, Information Directorate, 525 Brooks Road, Rome, NY 13441-4505, USA.
Manuscript received October 19, 2015, revised February 16, 2016, and ready for publication July 18, 2016.
Review handled by P. Willett.
0885/8985/17/$26.00 © 2017 IEEE



plication domains of FDIR. FDIR systems (1) generate a set of residuals to detect if a fault has occurred [15]–[19], (2) isolate the type of the fault, and (3) reconfigure the system according to the fault type, using precomputed strategies or recalculating control parameters online [20]. To alleviate the effect of noise on residuals, robust residual generation techniques, such as a Kalman Filter based approach [21], have been used. Hansen and Blanke have used error residuals from different sources including ground speed, estimated wind speed, and propeller speed, to successfully detect and isolate airspeed sensor faults [22]. The false positive and false negative detection rate of a FDIR method can be evaluated by several statistical models [20], [23].

PILOTS

The PILOTS programming language and software [13], [24], [25] uses established mathematical relationships between different data streams to monitor incoming data and detect errors.[1] Combining dynamic data driven modeling and real-time instrumentation monitoring, PILOTS is an effective way to catch and correct errors before they become a problem in order to ensure continued use of reliable data. Our PILOTS approach differs from information fusion by combining data and physics-based models to resolve anomalies as opposed to averaging out the estimation errors to maintain flight positions. The central theme is a dynamic data-driven applications system (DDDAS) [26] applied to avionics that utilizes measurements, model relationships, and software for rapidly correcting for aircraft sensor errors.

Since PILOTS implements fault-tolerant functionality to withstand data errors, it is expected that PILOTS' framework has resemblance to FDIR systems. Due to PILOTS' domain-specific programming language approach, PILOTS users have the ability to control error conditions more generally through error signatures. The differences between PILOTS' error signatures and FDIR's error residuals [27], [28] will be analyzed in the Discussion section.

In this article, we first introduce our model for data-error-tolerant stream processing and review flight accidents primarily due to sensor faults. Second, we describe the mathematical concepts we have developed to detect sensor faults from independently-measured redundant data streams, namely *error signatures* [24] and *mode likelihood vectors*. Third, we illustrate the use of the PILOTS system in analyzing two commercial aviation accidents initially caused by erroneous data coming from malfunctioning or incorrect sensors: Air France 447's pitot tubes icing and Tuninter 1153's installation of a wrong fuel quantity indicator. Finally, we discuss related work and advocate advances in sensor processing for safer air transportation.

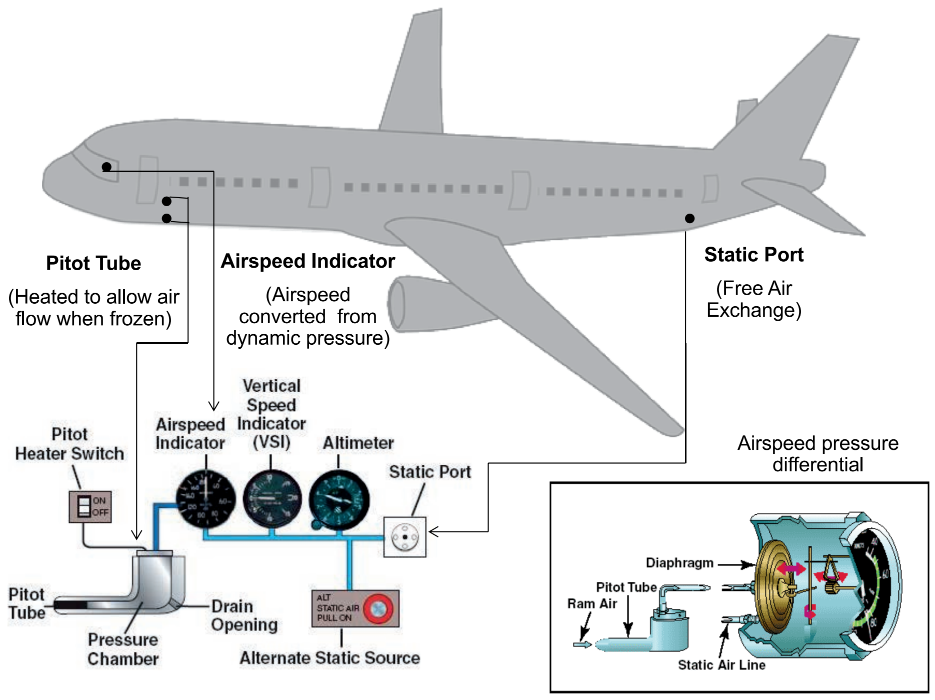


Figure 1.
Aircraft airspeed sensor configuration. (Adapted from https://en.wikipedia.org/wiki/Pitot-static_system.)

[1] Available in open-source form at: http://wcl.cs.rpi.edu/pilots/.

MOTIVATION

DATA-ERROR-TOLERANT STREAM PROCESSING

Fault detection is based on data errors, which can come from sensor, processing, or information errors. To detect data errors, PILOTS exploits logical redundancy that exists between fault-independent sensor sources. In the presence of faults, PILOTS uses known redundancy to estimate incorrectly reported data.

Consider an example shown in Figure 2a, which shows the relationship between three speed vectors: airspeed ($\vec{v}_a$), ground speed ($\vec{v}_g$), and wind speed ($\vec{v}_w$). These speeds are obtained through *independent* data collection methods: the ground speed is typically computed from GPS hardware, the airspeed is computed from dynamic air pressure measurements by pitot tubes, and the wind speed from weather forecast computer models. Since any one of the three speeds can be calculated using the other two with Equation (1) according to physical principles, they are *redundant* to each other. PILOTS first uses *crosscheck* on whether the relationship holds, and if not, estimates the correct values. For example, if $\vec{v}_a$ is incorrect, then $\vec{v}_a$ is estimated using vector arithmetic: $\vec{v}_g - \vec{v}_w$.

$$\vec{v}_g = \vec{v}_a + \vec{v}_w \tag{1}$$

Even though wind speed estimation from computer models may not be highly accurate, we only need a reasonable approximation to get the current fault mode (e.g., normal condition vs pitot-tubes fault). If Equation (1) holds within that error threshold, we can assume normal conditions and estimate wind speed from airspeed and ground speed. However, once Equation (1) is significantly violated, signaling an error—e.g., in the case of Air France 447, airspeed dropped from 470 knots to 180 knots after pitot tubes iced—the system can, upon issuing a warning and getting agreement from the pilots, use the best-known wind speed estimate to compute airspeed as shown in Figure 2b. If the error remains for too long—in AF447, pitot tubes recovered after 33 seconds—there are methods to estimate wind speed on board, such as making a circular standard-rate turn and calculating wind speed by GPS-measured position shift [29]. It is also possible to regularly get wind speed data from nearby airplanes or from updated weather forecasts.

Numerous flight accidents have resulted from sensor faults causing erroneous speed indications. Automated real-time sensor data cross-checks could help prevent similar accidents in the future.

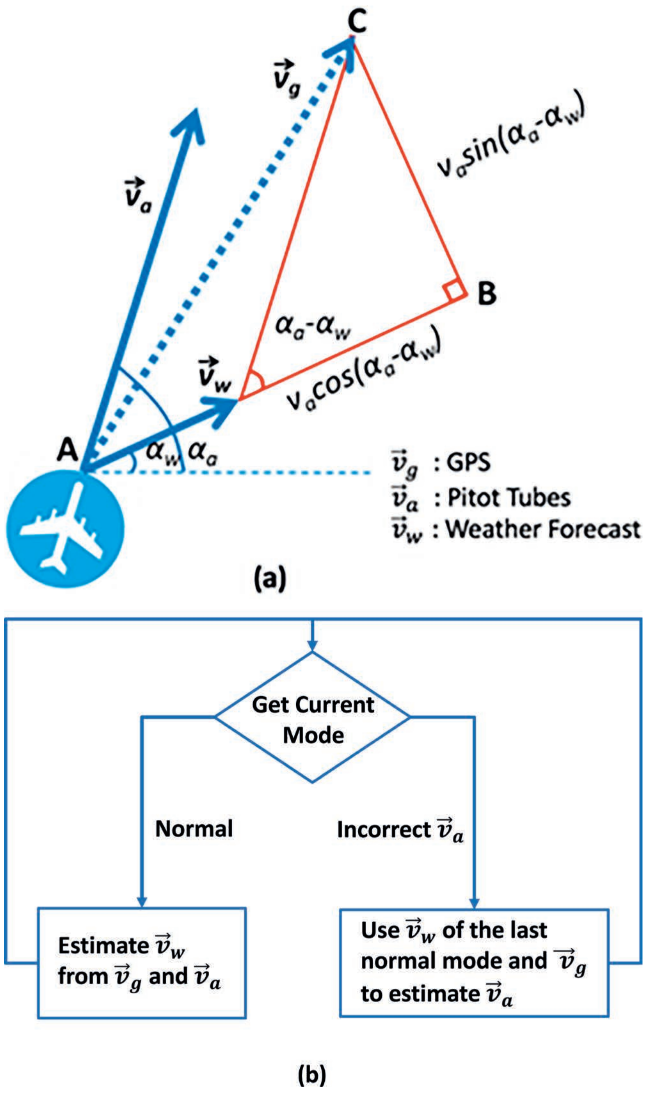

C
$\vec{v}_g$
$\vec{v}_a$
$v_a \sin(\alpha_a - \alpha_w)$
B
$\alpha_a - \alpha_w$
$\vec{v}_w$
$v_a \cos(\alpha_a - \alpha_w)$
A
α_w α_a
$\vec{v}_g$: GPS
$\vec{v}_a$: Pitot Tubes
$\vec{v}_w$: Weather Forecast
(a)
Get Current Mode
Normal
Incorrect $\vec{v}_a$
Estimate $\vec{v}_w$ from $\vec{v}_g$ and $\vec{v}_a$
Use $\vec{v}_w$ of the last normal mode and $\vec{v}_g$ to estimate $\vec{v}_a$
(b)

Figure 2.
(a) Independently-produced correlated data streams: airspeed ($\vec{v}_a$), ground speed ($\vec{v}_g$), and wind speed ($\vec{v}_w$). (b) Calculating wind speed in normal mode and using it to correct airspeed in the pitot-tube error mode.

FLIGHT ACCIDENTS

Table 1 presents several flight accidents that motivate the use of methods to perform fault-tolerant sensor data evaluation through data redundancy or system-advised precautionary measures. For example, in the accident of Azerbaijan Airlines Flight 217 in 2005, all three gyroscopes failed during the climb, while in the Copa Airlines Flight 201 in 1993, a short circuit rendered the attitude indicator inoperative. In both cases, a dynamic data-driven avionics system could recreate a virtual artificial horizon from nongyroscopic data, e.g., the path of the plane, and high-fidelity dynamics models [23] to give plausible/estimated attitude information to pilots.

Given the numerous aviation accidents that have occurred due to sensor faults inducing data errors, it is imperative to further research avionics systems for detecting and correcting for sensor malfunctions using logical redundancy.

DETECTING SENSOR FAULTS

ERROR SIGNATURE-BASED FAULT DETECTION

A key need in avionics health monitoring is capturing the redundancy between input variables, which requires an error function, also known as a *residual*. It is possible to recognize the shape of the error function on known faults by using *error signatures*. Signatures can then be used to identify an erroneous variable and compute a new value for that variable from redundant data. An

Table 1.

Flight Accidents and Possible Precautionary Measures

Flight	Date	Description of the Accident	Precautionary Measures
TransAsia Airways Flight 235	February 4th 2015	Two minutes after takeoff, pilots report engine flameout. Right engine failure alert, warning sounds for 3 sec. Crew reduces and then cuts the left engine.	Decision support system to not turn off the left engine.
Asiana Airlines Flight 214	July 6th 2013	Descent below visual glide path and impact with seawall. 82 seconds before impact at 1,600 ft, autopilot was turned off and throttles set to idle. Final approach speed was 34 knots below the target approach speed of 137 knots. Pilots unaware that the auto-throttle was failing to maintain that speed.	Internal glide-path assistance. Airspeed crosscheck.
Turkish Airlines Flight 522	February 25th 2009	Aircraft had an automated reaction which was triggered by a faulty radio altimeter. Auto-throttle decreased the engine power to idle during approach. Crew noticed too late. Although the pilots did try to hold the glide slope after increasing the throttle, the auto-throttle decreased it to idle again.	Sensing the altimeter error using crosschecks.
British Airways Flight 38	January 17th 2008	Although aware of the outside temperature conditions being -65C to -74C, the crew simply did not monitor the temperature of the fuel, which was well below freezing point. A small quantity of water within the fuel did freeze, causing ice on the fuel lines, ultimately leading to fuel starvation near the final stages of the approach.	Check for fuel temperature when outside air temperature outside normal range.
Azerbaijan Airlines Flight 217	December 23rd 2005	After climbing to 6,900 ft entered a descending spiral tightening from 500 m to 100 m. Absence of all three gyroscopes during the climb. Lack of pitch, roll, and heading performance.	Attitude indicator crosscheck. Re-create a virtual artificial horizon from nongyroscopic data.
Helios Airways Flight 522	August 14th 2005	Lack of cabin pressurization and pilot error leading to crew incapacitation due to hypoxia. Cabin altitude warning horn sounded at 12,040 ft and should have stopped the climb.	Prevent the climb or strongly advise pilots against it.
Air Midwest Flight 5481	January 8th 2003	Elevator range of motion cut to only 7 degrees out of the full 14. Stalled after take-off due to overloading and maintenance error.	Weight and systems check from sensors onboard before departure.
Austral Lineas Aereas Flight 2553	October 10th 1997	Pitot tube icing caused faulty airspeed readings. Pilots interpreted as a loss of engine power and added power. No improvement to airspeed, so they descended and increased the speed. Wing slats were torn off one wing and the plane became uncontrollable.	Airspeed crosscheck.
Copa Airlines Flight 201	June 6th 1993	At 25,000 ft, the plane entered a steep dive at an angle of 80 degrees to the right and began to roll. Exceeding the speed of sound at 10,000 ft, the plane broke apart. Faulty readings were caused by a short circuit traced to a faulty wiring harness in the attitude indicator.	Attitude indicator crosscheck. Recreate a virtual artificial horizon from nongyroscopic data.

overview of the error detection method based on different error signatures is summarized in Figure 3.

The DDDAS PILOTS approach includes five steps: (1) error function model development, (2) error signature pattern analysis, (3) likelihood estimation of anomalies against stochastic errors, (4) dynamic mode estimation, and (5) application-based data correction based on logical redundancy in environmental and system conditions.

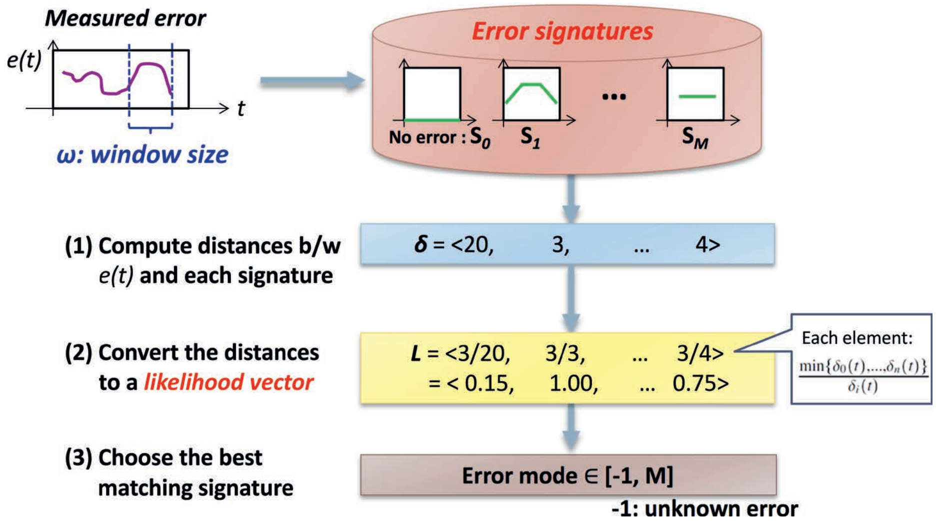

Measured error
e(t)
t
ω: window size
Error signatures
No error : S0
S1
SM
(1) Compute distances b/w e(t) and each signature
δ = <20, 3, ... 4>
(2) Convert the distances to a likelihood vector
L = <3/20, 3/3, ... 3/4>
= < 0.15, 1.00, ... 0.75>
Each element: min{δ0(t),...,δn(t)} / δi(t)
(3) Choose the best matching signature
Error mode ∈ [-1, M]
-1: unknown error

Figure 3.
Error signature-based fault detection method.

1. **Error functions:** An *error function* is a function that computes a numerical value from independently measured input data. It is used to examine the validity of redundant data. If the value of an error function is zero, we interpret it as no error in the given data. For the speed vector example shown in Figure 2a, we can define an error function as follows [22].

$$e(\vec{v}_g,\vec{v}_a,\vec{v}_w)=|\vec{v}_g|-|\vec{v}_a+\vec{v}_w| \\ = v_g-\hat{v}_g = v_g-\sqrt{v_a^2+2v_av_w\cos(\alpha_a-\alpha_w)+v_w^2} \quad (2)$$

The values of input data are sampled periodically from corresponding data streams. Thus, an error function e changes its value as time proceeds, represented as $e(t)$.

2. **Error signatures:** A *error signature* is a set of constrained mathematical functions that is used to capture the characteristic pattern of an error function $e(t)$ under a specific fault condition. Using a vector of constants $\bar{K}=\langle k_1, k_2, \ldots, k_n\rangle$, a function $f(t, \bar{K})$, and a set of constraint predicates $\bar{P}(\bar{K})=\{p_1(\bar{K}),\ldots,p_l(\bar{K})\}$, the error signature S is defined as follows:

$$S\left(f\left(t,\bar{K}\right),\ \bar{P}\left(\bar{K}\right)\right)\triangleq\left\{f\mid p_1\left(\bar{K}\right)\wedge\cdots\wedge p_l\left(\bar{K}\right)\right\}. \quad (3)$$

For example, an interval error signature S_I can be defined using a set of constraint predicates $\bar{P}_I(\bar{K},\bar{A},\bar{B}) = \{a_1 \le k_1 \le b_1, \ldots, a_m \le k_m \le b_m\}$ denoting intervals, as follows:

$$S_I=S\left(f\left(t,\bar{K}\right),\bar{P}_I\left(\bar{K},\bar{A},\bar{B}\right)\right)=\left\{f\mid a_1\le k_1\le b_1,\ldots,a_m\le k_m\le b_m\right\}. \quad (4)$$

For example, letting $\bar{K}=\langle k\rangle, f(t,\langle k\rangle)=t+k, \bar{A}=\langle 2\rangle$ and $\bar{B}=\langle 5\rangle$, the error signature S_1 contains all linear functions with slope 1, and crossing the Y-axis at values from 2 to 5 as shown in Figure 4.

3. **Mode likelihood vectors:** Given a vector of error signatures $\langle S_0, S_1, \ldots, S_m\rangle$, we calculate $\delta_i(t)$, the distance between the monitored error function $e(t)$ and each error signature S_i by:

$$\delta_i(t)=\min_{g(t)\in S_i}\int_{t-\omega}^{t}\left|e(t)-g(t)\right|dt, \quad (5)$$

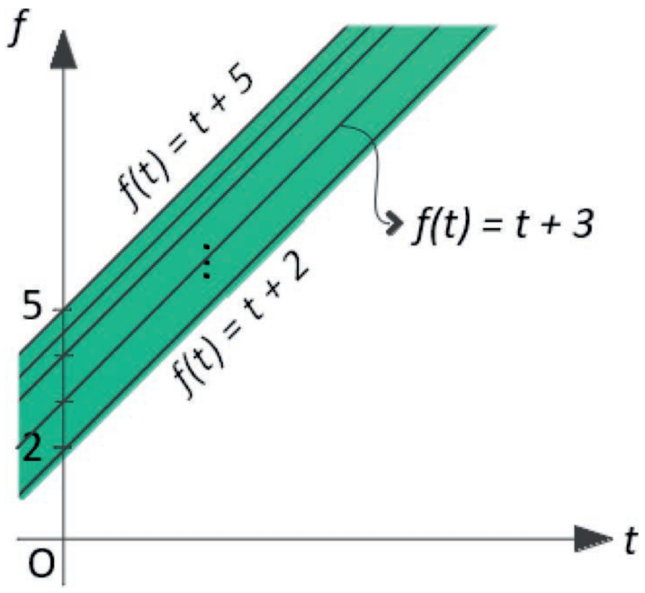


Figure 4.
Error signature $S_1 = S(f(t, \langle k\rangle) = t + k, \{2 \le k \le 5\})$.

where ω is a *window size* parameter. Note that our convention is to capture *normal* conditions as signature S_0. The smaller the distance δ_i, the closer the measured data is to the theoretical signature S_i. We define the *mode likelihood vector* as $L(t) = \langle l_0(t), l_1(t), \ldots, l_m(t)\rangle$ where $l_i(t)$ is defined as:

$$l_i(t)=\begin{cases}1, & \text{if } \delta_i(t)=0\\ \dfrac{\min\{\delta_0(t),\ldots,\delta_m(t)\}}{\delta_i(t)}, & \text{otherwise}\end{cases}. \quad (6)$$

Observe that each $l_i \in (0,1]$ represents the ratio of the likelihood of signature S_i being matched with respect to the likelihood of the best matching signature.

4. **Mode estimation:** Using the mode likelihood vector, the final *mode,* which corresponds to the most likely signature, is

estimated. Because of the way $L(t)$ is created, the largest element will always be equal to 1. Given a significance threshold $\tau \in (0,1)$, PILOTS checks for one likely candidate l_j that is sufficiently more likely than its successor l_k by ensuring that $l_k \leq \tau$. Thus, j is determined to be the correct mode by choosing the most likely error signature S_j. If $j = 0$, then the system is in *normal mode*. Well-designed vectors of error signatures should produce mode likelihood vectors forming an orthonormal basis under the modeled $m + 1$ conditions. However, if $l_k > \tau$, then both modes j and k match the measured error within the significance threshold, and therefore, *unknown error* mode (−1) is estimated.

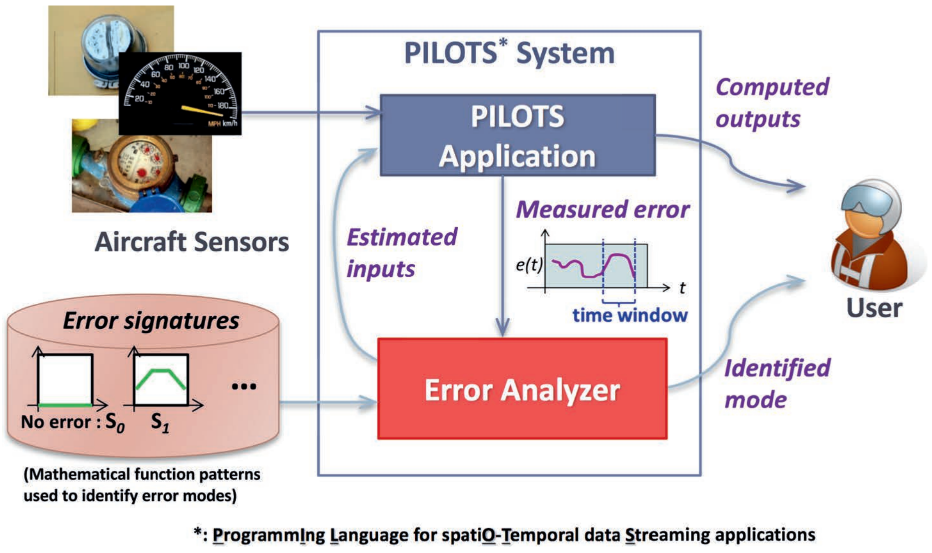


Figure 5.
High-level architecture of the PILOTS system.

5. **Fault-tolerant sensor fusion:** It is problem dependent if a known error mode i is recoverable or not. If there is a mathematical relationship between an erroneous value and other independently measured values, the erroneous value can be replaced by a new value computed from the other independently measured values. In the case of the speed vector example used in Equation (2), if the ground speed v_g is detected as erroneous, its true value $\hat{v}_g$ can be estimated as follows:

$$\hat{v}_g = \sqrt{v_a^2 + 2v_a v_w \cos(\alpha_a - \alpha_w) + v_w^2}. \qquad (7)$$

PROGRAMMING SYSTEM SUPPORT

PILOTS enables data analysts and programmers to use a high-level application programming environment for designing error signatures for fault-tolerant dynamic data stream processing.

System Architecture

Figure 5 shows the high-level architecture of the PILOTS runtime system. The *PILOTS application* is written in the PILOTS programming language, which has been developed for aviation studies to analyze inputs from aircraft sensors. The PILOTS initial studies and experimental analyses have focused on aviation data analytics, but could be adapted to other sensor and actuator systems. The *error analyzer* collects the latest ω error values from the PILOTS application and keeps detecting faults based on known error signatures. If PILOTS detects a recoverable error, it replaces an erroneous input with the estimated one by applying a corresponding data estimation function. Finally, the PILOTS application computes output streams based on the estimated inputs produced from the error analyzer. These output values could be displayed in the cockpit or used directly in unmanned flight systems.

PILOTS PROGRAMMING LANGUAGE

PILOTS is a declarative, domain-specific programming language specifically developed to help users analyze spatio-temporal data streams and design error signatures to detect erroneous data. Using PILOTS, application programmers can easily develop systems that handle data streams by writing a high-level program specification. Table 2 compares the number of lines to write the three programs we will introduce using PILOTS and Java.

PILOTS programs must contain an `inputs` and an `outputs` section. The *inputs* section specifies the data streams and how data is to be extrapolated from incomplete data, typically using declarative geometric criteria such as `closest`, `interpolate`, and `euclidean` keywords (refer to [25] for details of the geometric criteria). The *outputs* section specifies data streams to be produced by the application, as a function of the input streams with a given frequency. The optional `errors` and `signatures` sections are required in order to detect errors. The *errors* section specifies error

Table 2.
Comparison of Lines of Code Between PILOTS and Java Applications

Programming Language	Twice	SpeedCheck	WeightCheck
PILOTS	16	23	18
Java	100	141	114

```
program Twice;
inputs
  a(t) using closest(t);
  b(t) using closest(t);
outputs
  o: b – 2 * a at every 1 sec;
errors
  e: b – 2 * a;
signatures
  S0   :e = 0                 "Normal";
  S1(k):e = 2 * t + k         "A failure"
    estimate a = b / 2;
  S2(k):e =-2 * t + k         "B failure"
    estimate b = a * 2;
  S3(k):e = k, abs(k) > 20    "Out-of-sync";
end
```

Figure 6.
Twice PILOTS program to detect and correct errors in a simple two-stream application.

streams to be produced by the application and to be analyzed by the runtime system to recognize known error patterns as specified in the *signatures* section. If a detected error is recoverable, output values are computed from input data after being estimated using formulas under the **estimate** clause of the *signatures* section.[2]

An example PILOTS program called ***Twice*** is shown in Figure 6 that compares two data streams. The two input streams, *a* and *b*, are specified in the *inputs* section. The value of *b* is always supposed to be twice as large as the value of *a*, so that the value of *o* and *e* in the *outputs* and *errors* sections should always be zero under normal conditions. There are four error signatures *S0, ..., S3* under the *signatures* section. In this example, *S1* and *S2* are detectable and correctable whereas *S3* is detectable, but not correctable.

APPLYING SENSOR FAULT DETECTION AND CORRECTION TO ACTUAL COMMERCIAL FLIGHT ACCIDENTS' DATA

We present two examples of using PILOTS for aviation data error analysis: (1) Air France Flight 447 and (2) Tuninter Flight 1153. For each example, the flight data, experimental design, and results are illustrated.

AIR FRANCE FLIGHT 447: PITOT TUBES ICING

Air France Flight 447 (AF447), which departed from Rio de Janeiro bound for Paris on June 1st, 2009 (flight path shown in Figure 7), was one of the worst flight accidents in aviation history [9]. The initial cause of the accident was identified as incorrect airspeed readings caused by pitot tubes clogged with ice crystals. The AF447 challenge was that all the pitot tubes failed under the same icing conditions. Other sensors or information could have been incorporated to cross-check the sensor readings to inform the pilots or auto-pilot of the correct aircraft speed. Research suggests that the accident could have been prevented by endowing the flight system with the ability to estimate the speed using the physical relationship between different speed measurements (Equation 1) to detect and correct for the error. To demonstrate the applicability of error signatures to recover the airspeed from the actual ground speed stored in the flight data recorder and the weather forecast-based wind speed; we created an error signatures vector to detect airspeed sensor faults and developed a corresponding PILOTS program.

[2] We used *correct* in earlier versions of the software/papers, but *estimate* better reflects that the mathematical formula is an estimation of a data stream. Thus, the new terminology and syntax will be used in PILOTS release 0.3.1 and higher.

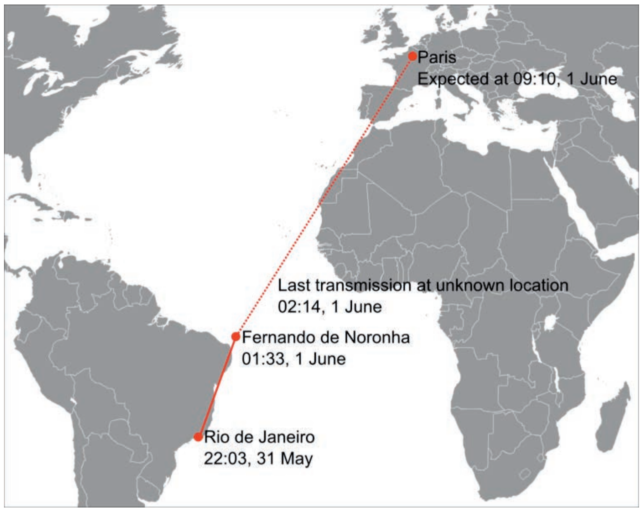


Figure 7.
Air France Flight 447 flight investigation. (Adapted from https://commons.wikimedia.org/wiki/File:Air_France_Flight_447_path.png.)

Error Signatures Vector

In addition to considering the pitot tubes fault that actually occurred in the AF447 flight, we design a general error signatures vector that can detect the following four conditions: (1) normal, (2) pitot tube fault due to icing, (3) GPS fault, and (4) simultaneous pitot tube and GPS faults. Suppose the airplane is flying at airspeed v_a, from which other speeds are expressed in relation to v_a as follows:

- Wind speed: $v_w \le a v_a$, where *a* is the maximum reasonable wind speed to airspeed ratio.
- Ground speed: $(1 - a)v_a \le v_g \le (1 + a)v_a$
- Pitot tube failed airspeed: $b_l v_a \le \tilde{v}_a \le b_h v_a$, where b_l and b_h are the lower and higher values of pitot tube clearance ratio $b_l, b_h \in [0,1], b_l \le b_h$. 0 represents a fully clogged pitot tube, while 1 represents a fully clear pitot tube, and
- GPS failed ground speed: $\tilde{v}_g = 0$.

We use $\tilde{v}_a$ and $\tilde{v}_g$ to denote erroneous airspeed and ground speed, respectively. We assume that when a pitot tube icing occurs, it is gradually clogged and thus the airspeed data reported from the pitot tube system (which may combine multiple physical probe measurements) also gradually drops and eventually remains at a constant speed while iced. This is consistent with data from AF447. The resulting constant speed is characterized by ratio b_l and b_h. On the other hand, when a GPS fault occurs, the ground speed suddenly drops to zero. This is why we model the failed ground speed as $\tilde{v}_g = 0$. Using these parameters and the error model Equation (2), we design the error signatures vector shown in Table 3 (for more details on how to create error signatures vectors, please refer to [24]).

Speed Check PILOTS Program

A PILOTS program called ***SpeedCheck*** implementing the error signatures vector of Table 3 is presented in Figure 8. ***SpeedCheck*** checks if the wind speed, airspeed, and ground speed are consistent with each other or not, and computes a crab angle, which is used to adjust the direction of the aircraft to keep a desired ground track based on the estimated values. If a "GPS fault" is detected by the error signature *S2*, the *Error Analyzer* estimates the ground speed from the airspeed and wind speed Equation (7). Similarly, if a "Pitot tube fault" is detected by *S1*, a similar data correction can be applied to estimate the airspeed. On the other hand, if a "Both pitot tube and GPS faults" condition is detected by *S3*, PILOTS cannot estimate any values due to not having sufficient redundancy in the data. With the inability to estimate speeds, an alert or warning should be sent to the aircrew. It is assumed that with aircraft inspections, the likelihood of redundant fault-independent sensors producing data errors would be limited to situations of aircraft distress, such as structural failure.

```
program SpeedCheck;
  /* v_w: wind speed,           a_w: wind speed angle
     v_a: airspeed,             a_a: airspeed angle
     v_g: ground speed,         a_g: ground speed angle */
inputs
  v_w, a_w (x,y,z,t) using euclidean(x,y), closest(t), interpolate(z,2);
  v_a, a_a (x,y,t)   using euclidean(x,y), closest(t);
  v_g, a_g (x,y,t)   using euclidean(x,y), closest(t);
outputs
  crab_angle : arcsin(v_w * sin (a_w - a_a) /
                sqrt(v_a^2 + 2 * v_a * v_w * cos(a_w - a_a) + v_w ^2))
                at every 1 sec;
errors
  e: v_g - sqrt(v_a^2 + v_w^2 + 2 * v_a * v_w * cos(a_w - a_a));
signatures
  /* v_a = 470 knots, a = 0.1, b_l = 0.2, b_h = 0.33 */
  S0(k):e = k,    -47 <= k, k <=  47    "Normal";
  S1(k):e = k,  220.9 <= k, k <= 517    "Pitot tube fault"
    estimate  v_a = sqrt(v_g ^2 + v_w^2 + 2 * v_g * v_w * cos(a_g - a_w));
  S2(k):e = k,   -517 <= k, k <=-423    "GPS fault"
    estimate  v_g = sqrt(v_a ^2 + v_w^2 + 2 * v_a * v_w * cos(a_w - a_a));
  S3(k):e = k,-203.66 <= k, k <=- 47    "Both pitot tube and GPS faults";
end
```

Figure 8.
PILOTS program to detect and correct for speed data faults.

For the ***SpeedCheck*** PILOTS program to be applicable to the AF447 flight, we use a cruise speed of 470 knots as v_a, the cruise speed of the AF447 flight available from the flight data recorder information before the accident. Note that the angle signs are reversed in the PILOTS implementation as opposed to the equations presented in the previous sections. In trigonometry, angles increase counter-clockwise (with 0° representing East) while in aviation, angles increase clockwise (with 0° representing North).

Evaluation

Flight data: The ground speed and airspeed were collected based on Appendix 3 in the final accident report of Air France Flight 447 [9]. Note that the (true) airspeed was not recorded in the flight data recorder, and so it was computed from recorded Mach (*M*) and static air temperature (*SAT*) data. The airspeed was obtained by using the relationship: $v_a = a_0 M\sqrt{SAT / T_0}$, where a_0 is the speed of sound at standard sea level (661.47 knots) and T_0 is the temperature at standard sea level (288.15 Kelvin). Independent wind speed information was not recorded either. According to the description from page 47 of the final report: "(From the weather forecast) the wind and temperature charts show that the average effective wind along the route can be estimated at approximately *ten knots tail-wind*." We followed this description and created the wind speed data stream as ten knots tail wind. As shown in Figure 2b, weather forecast wind speed was used for checking the correctness of air speed and ground speed. If the ***SpeedCheck*** PILOTS program detects a pitot tube failure (*S1*) or a GPS failure (*S2*), accurate wind speed from the last normal mode is used for correction of air speed (*S1*) or ground speed (*S2*).

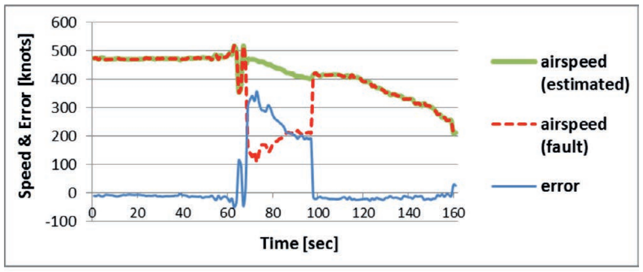
Experimental settings: According to the final report [9], speed data was provided from 2:09:00 UTC on June 1st, 2009 and it became invalid after 2:11:42 UTC on the same day. Thus, we examine the valid 162 seconds of speed data including a period of pitot tube failure which occurred from 2:10:03 to 2:10:36 UTC. We used the ***SpeedCheck*** PILOTS program shown in Figure 8.

Results: With $\omega = 1$ and $\tau = 0.8$, Figure 9 shows the PILOTS results. From Figure 9a, the airspeed error is successfully corrected at 69 seconds, which is 5 seconds after the start of the fault, and seamlessly transitions to the normal airspeed when pitot tubes recover at 98 seconds. From Figure 9b, we can see that the mode spikes twice to the both faults condition at around 63 seconds. This is due to the gradual change in the error; however, the pitot tube fault condition is successfully detected in the time it would have taken a human expert (e.g., cockpit crew) given the same data. The accuracy of the prediction is 96.3% for the entire test period, with 3.7% false negative rate, and 0% false positive rate.

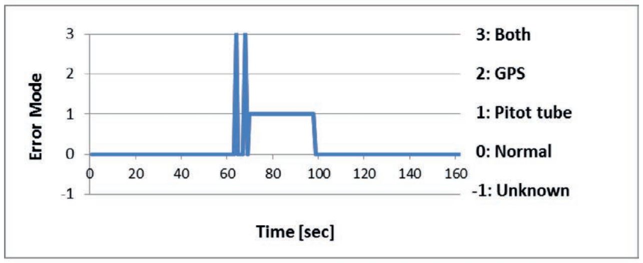
Table 3.

Error Signatures Vector for Speed Data

Mode	Error Signature	
	Function	Constraints
Normal	$e = k$	$-av_a \le k \le av_a$
Pilot tube fault		$(1 - 2a - b_h)v_a \le k \le (1 + a - \lvert a - b_l \rvert)v_a$
GPS fault		$-(a + 1)v_a \le k \le -\lvert a - 1 \rvert v_a$
Both faults		$-(a + b_h)v_a \le k \le -\lvert a - b_l \rvert v_a$



(a) Speeds and error



(b) Detected modes

Figure 9.
Estimated airspeed and detected modes for AF447 flight. Once an error is detected, the airspeed is recalculated (green line) by the appropriate correction formula, i.e., S1 line in correct section of `SpeedCheck` PILOTS program (Figure 8).

TUNINTER FLIGHT 1153: WRONG FUEL QUANTITY INDICATOR

Tuninter Flight 1153 (TU1153) was a flight from Bari, Italy to Djerba, Tunisia on August 6th, 2005 (flight path shown in Figure 10). About an hour after the departure, the ATR 72 aircraft ditched into the Mediterranean Sea due to the exhaustion of its fuel, killing 16 of 39 people on board (see Figure 11 for altitude transition obtained from the accident report [30]). The accident was caused by the installation of an incorrect fuel quantity indicator. That is, a quantity indicator for the smaller ATR 42 model was mistakenly installed, reporting 2,150 kg more fuel than actually available.

How could this accident be prevented? If all other conditions are the same between two flights except for the weight, the one with lighter weight would have a higher airspeed. If a system could compute expected airspeed from the monitored weight, the expected and monitored airspeed could be compared to detect if there is a discrepancy between the two. Once the system detects the discrepancy, it can warn pilots in an early stage of the flight to prevent accidents. Using this idea, we design a vector of error signatures and evaluate it with real data recorded during the TU1153 flight.

Error Signatures Vector

In the ATR 72 fight crew operating manual [31], there are tables for pilots to estimate cruise airspeed (knots) under certain conditions of engine power settings, temperature difference to the International Standard Atmosphere (Celsius), flight level (feet), and weight (kg) of the aircraft; which are denoted by v_a, t_Δ, h, and w, respectively. To interpret the tables, we fix $t_\Delta = +10$, since according to the accident report, that is a reasonable approximation at the time of the accident. To obtain a relationship that holds among these variables, a polynomial regression was applied to the operating manual tables to derive an equation to estimate the airspeed. Then, to get an improved functional fit to the actual data, a calibration term (= – 2.59), computed from the first cruise phase from the actual TU1153 flight, was added to the regression equation. As the result, the regression estimate airspeed $\hat{v}_a$ model is:

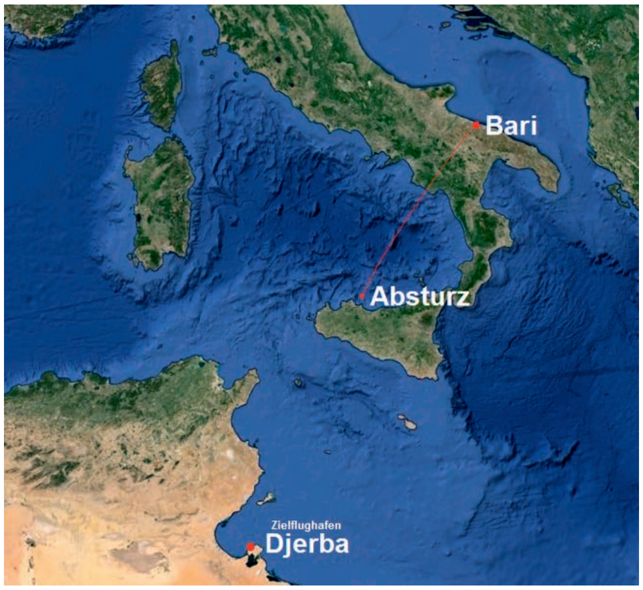


Figure 10.
Tuninter Flight 1153 flight path. (Tuninter crash.png by DOWIMA, adapted from https://commons.wikimedia.org/wiki/File:Tuninter_crash.png.)

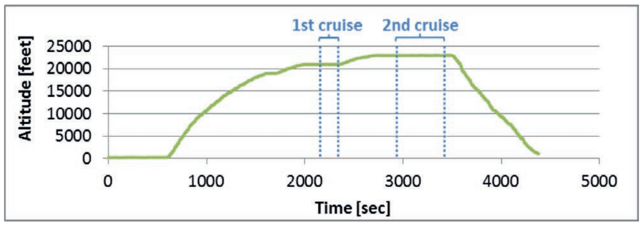


Figure 11.
Altitude transition of the TU1153 flight.

$$\hat{v}_a = \begin{bmatrix} 1 \\ w \\ h \\ w \cdot h \\ w^2 \\ h^2 \end{bmatrix}^T \cdot \begin{bmatrix} 6.4869E+01 \\ 1.4316E-02 \\ 6.6730E-03 \\ -3.7716E-07 \\ -2.4208E-07 \\ -1.1730E-07 \end{bmatrix} - 2.59. \tag{8}$$

Likewise, an error function is defined as follows:

$$e(v_a, w, h) = v_a - \hat{v}_a \tag{9}$$

Since the error function is close to zero on average after the calibration, naturally the error value of zero with small margins identifies a normal condition. For underweight conditions, we set a constraint that will identify 10% discrepancy in weight. Since the 10% weight difference leads to a 4.69 knots difference in airspeed from Equation (8) (computed by averaging over $w = 13{,}000 \sim 22{,}000$ kg at 23,000 feet), 4.69 is used as the boundary for the underweight conditions. In summary, the error signatures vector is shown in Table 4.

Table 4.

Error Signatures Vector for the Tuninter 1153 Flight

Mode	Error signature	
	Function	Constraints
Normal	$e = k$	$-2 < k < 2$
Underweight		$4.69 < k$

Weight Check PILOTS Program

A PILOTS program called ***WeightCheck*** implementing the error signatures vector of Table 4 is presented in Figure 12. ***WeightCheck*** detects an underweight condition by comparing the monitored and estimated airspeed computed from the weight and altitude. Once ***WeightCheck*** detects the underweight condition by the error signature *S1*, it estimates the corrected weight, which can be obtained by solving Equation (9) for *w*.

```
program WeightCheck;
  /* v_w: wind speed, w: weight, h: altitude */
inputs
  v_a, w, h(t) using closest(t);
outputs
  corrected_w: w at every 1 sec;
errors
  e: v_a – ((6.4869E+01 + 1.4316E-02 * w + 6.6730E-03 * h +
            (-3.7716E-07) * w * h + (-2.4208E-07) * w * w +
            (-1.1730E-07) * h * h) - 2.59);
signatures
  S0(k): e = k, -2 < k, k < 2              "Normal";
  S1(k): e = k, 4.69 < k                   "Underweight"
    estimate w = 3.34523E-12 *
          (sqrt(1.09278E+22 * h * h + (-1.65342E+27) * h +
                (-3.69137E+29) * v_a + 1.01119E+32) –
           2.32868E+11 * h + 8.83906E+15);
end
```

Figure 12.
PILOTS program to detect and correct for weight data faults.

Evaluation

Flight data: The airspeed, monitored fuel weight, and altitude of the aircraft are collected from Attachment H of the accident report [30]. Since the total weight of the aircraft is needed for the model used in ***WeightCheck***, the fuel weight comparison to the zero fuel weight of the aircraft is computed and added to the model. As a result, before the departure, the monitored "fictitious" weight of the aircraft is 19,420 kg, whereas the actual weight is 17,270 kg.

Experimental settings: We run the ***WeightCheck*** PILOTS program in Figure 12 for 1,500 seconds, from 2,000 to 3,500 seconds after departure including the first (2,170–2,370 seconds) and the second cruise phases (2,960–3,450 seconds), to see how the error signatures vector works for these two cruise phases. Note that we evaluate only the second cruise phase since the error signatures vector is adjusted by the first cruise phase.

Results: With $\omega = 1$ and $\tau = 0.8$, Figure 13 presents the results. As shown in Figure 13b, the PILOTS program correctly detects the underweight condition for the first cruise phase whereas it detects the underweight condition well before the second cruise phase starts. The fault is detected early because the error goes beyond 4.69 (the boundary for the underweight condition) around 2,770 sec-

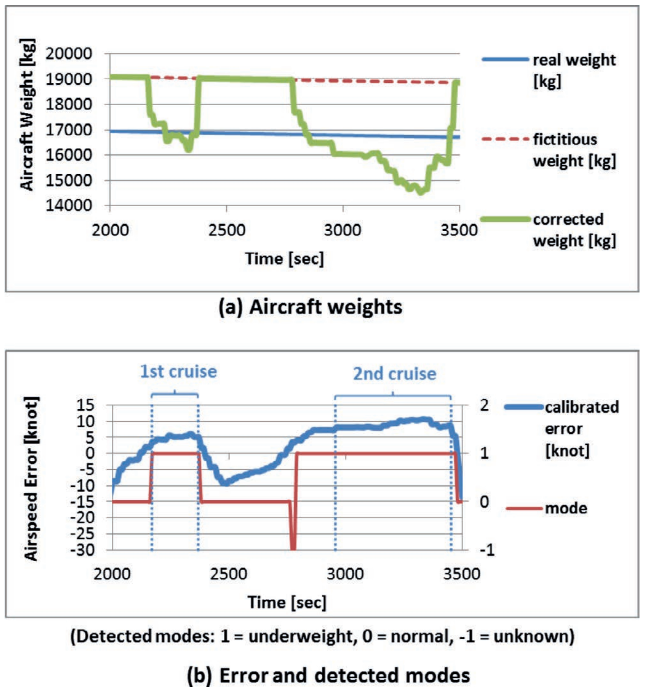


Figure 13.
Aircraft weights and detected modes for the TU1153 flight.

Table 5.

Comparison of Error Signatures and Binary Residuals

	Error Functions and Signatures	Binary Residuals
Expected value in normal mode	Zero	Zero
Expected value in error modes	Not zero, value is compared to signatures and classified into different modes.	Not zero, value is not further considered.
Number of error functions/residuals for fault isolation	One	Multiple
Need to estimate signatures for different error modes	Yes	No
Can detect single source fault	Yes	Yes
Can isolate single source fault	Yes, but it depends on accurate fault modeling.	Yes, but it depends on multiple models/residuals.
Can isolate most common faults	Yes	Yes

onds. From Figure 13a, PILOTS indicates that the estimated weight is reasonably close to the real weight during the first cruise phase, but is not very close during the second phase. That is, the differences between the estimated and real weights are at most ~643 kg for the first cruise phase and ~1,935 kg for the second cruise phase. The PILOTS corrected weight discrepancy between the two phases can be explained by inaccuracy of the airspeed estimation during the second cruise phase, which may be caused by variables that are not considered such as angle of attack, center of gravity, and aircraft engine and configuration settings. The results reveal that the PILOTS airspeed prediction model is not accurate enough to precisely estimate the weight from the airspeed; but nonetheless, it is able to detect the underweight condition.

DISCUSSION

COMPARISON OF ERROR SIGNATURES AND BINARY RESIDUALS

In the PILOTS programs, error functions and signatures are used for fault detection and isolation. Error signatures are compared to binary error residuals in Table 5. The expected value of error functions and residuals under normal conditions is zero. When the value of an error function deviates from zero, it is compared to a vector of error signatures that are determined theoretically based on models of different fault types. Using mode likelihood vectors, the most likely mode is estimated which may correspond to a single sensor fault or multiple sensor faults. In the case of a single sensor fault, PILOTS uses logical redundancy to estimate the correct value for the erroneous data. In the simplified FDIR methods implemented in the PILOTS declarative language, the value of a residual was considered as binary. As shown in the ***SpeedCheck*** program, one error function and four signatures are used to detect and isolate different fault conditions. Similar work could be done using residual based methods, with three different residuals for isolation [22].

PILOTS works best in detecting and isolating faults within the assumptions used to generate the error signatures. In ***Speed–Check***, a pitot tube clearance ratio of 0.1–0.33 is assumed when pitot tubes fail due to icing, and a ground speed indication of zero when GPS fails (GPS denied). If the pitot tube clearance ratio is not in the expected range, or ground speed fails in some other unanticipated way, the PILOTS program will most likely report “unknown error”. While in a binary residual based method, with properly chosen redundant residuals, the unanticipated fault would be correctly isolated. Even though the error signature-based approach is less capable of isolating unanticipated errors with respect to residual-based approaches, the error signatures require fewer error functions (redundancy models) and may be therefore less sensitive to modeling errors, demonstrating very good performance in properly anticipated error conditions. Also, in some FDIR research for automotive applications [27] [28], the residuals are not considered as binary values, but are assumed to have different distributions according to different modes. In these cases, the residual based methods are similar to the error signatures approach.

Assuming sensor faults exhibit characteristic error function “signatures,” PILOTS is able to alert aircrew of possible anomalies. Trusting the output is based on the fidelity of the error and signatures model; however, even properly designed simple analytics could provide the aircrew timely updates of possible sensor faults using additional information from weather forecasting, other sensors, and physical models of airplane parameters such as weight profiles. Increasing the fidelity of any of these methods would enhance the results from PILOTS and would be needed for any aviation equipment with flight passengers. In summary, error signatures require modeling specific fault modes, but have the potential to enhance pilot awareness complementing residual-based error detection and isolation techniques.

SENSOR MATHEMATICAL MODELING

Further research is needed in understanding and formalizing the relationships between hundreds of data streams that are available to flight systems. In the Air France Flight 447 case study, the relationship between speed vectors was mathematically simple (i.e., Equation (1)), whereas in the Tuninter 1153 case study, the relationship between weight and aircraft performance was not so easy to formalize since different factors beyond weight affect air speed. Figure 14 shows the relationships between the two case studies.

While it is possible to use aerodynamics theory to relate lift and weight (equal in cruise phases of flight) to airspeed, other factors such as altitude, temperature (affecting air density), angle of attack, and engine settings play an important role in the data relationship.

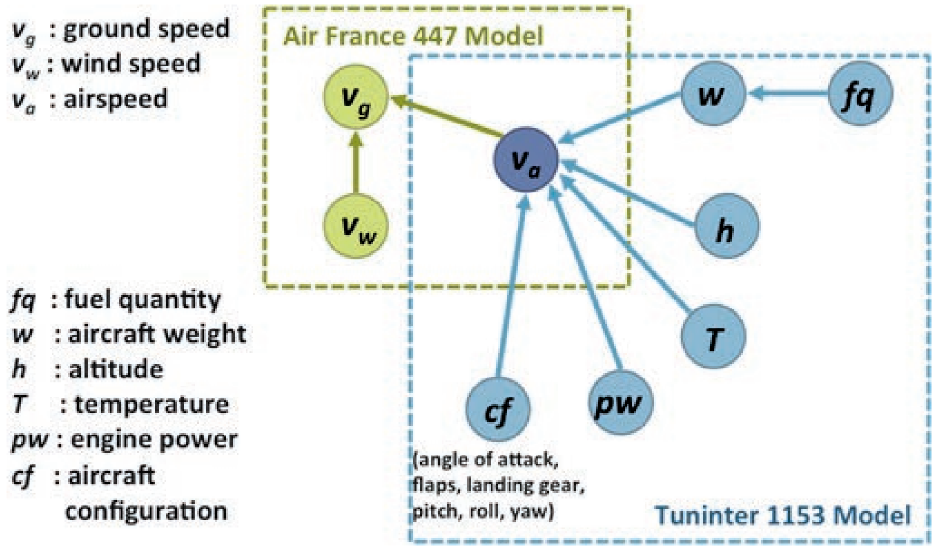


Figure 14.
Examples of data stream relationships in AF447 and TU1153.

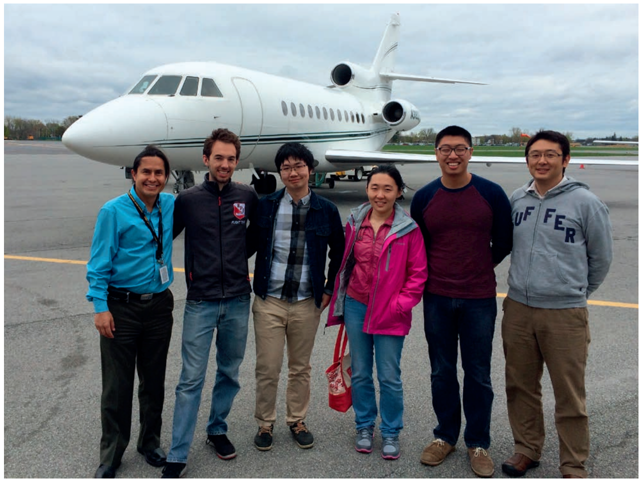
SENSOR FUSION

Sensor fusion analysis for aircraft safety includes three areas: (1) health monitoring, (2) pilot awareness, and (3) air-ground coordination. For health monitoring, there are many opportunities that include the DDDAS paradigm with multimeasurement sensor fusion, software enhancements, and uncertainty modeling [32]. Using the normal flight operations, models can be used to cross-check sensor readings and with our proposed software and techniques [13], alert when unexpected sensor readings occur. Inherently, sensor fusion techniques can determine inconsistency between readings. Examples of redundant sensor monitoring would best be achieved with more than two sensors to resolve conflicts between only two sensors [33]. The second category supports pilot awareness which in and of itself is the leading cause of accidents (e.g., pilot error). Sensor fusion would be able to use multiple sensors to display warnings in the cockpit. The challenge is to experiment (e.g., with a simulator) the optimum display technology to warn pilots of critical challenges. There have been many accidents when the pilot failed to take action even in the presence of stall lights and horns [33]. The final category is air-ground coordination which could be both a machine (e.g., GPS) and communications (e.g., with Air Traffic Controllers) that alert to low approaches, see-through bad weather conditions, relevant pilot reports and data from other aircraft, and identification of unavoidable terrain. The future of sensor fusion for aircraft accident mitigation is in a variety of methods to provide external validation of unsafe flight that can enhance action plans when health monitoring and pilot awareness are not sufficient.

CONCLUSION AND FUTURE WORK

Supplementing flight systems with error detection and data estimation based on error signatures can add another layer of (logical) fault-tolerance enabling safer flight. In this article, we overviewed the **P**rogramm**I**ng **L**anguage for spati**O**-**T**emporal data **S**treaming applications (PILOTS) system which is a declarative language. With tens of lines of code, a data scientist can test and validate their error detection models in the form of error signatures. Examples were presented to demonstrate dynamic data-driven error detection and correction using data from the Air France Flight 447 and the Tuninter Flight 1153 accident reports. We view our work on error signatures as complimentary to existing work on fault detection, isolation, and reconfiguration. In particular, it is possible to use high-fidelity models that constrain the relationship between sensor and actuator data following aerodynamic principles. Such systematic analysis of fault behaviors can result in a set of residuals and error signatures that improve the performance and accuracy of fault detection, isolation, and reconfiguration strategies in flight systems. Such additional avionics support is needed to alert pilots with warnings of sensor faults in a more effective way for timely diagnostics and safer flight.

Future work includes exploring automated error signature acquisition by using machine learning and distributed computing [35], [36]. Machine learning techniques [37] may help physics-based models be more accurate by taking data into account. A direction of future work for the PILOTS project is to be expanded into a *DDDAS Model Learning Toolkit*: to facilitate Monte Carlo simulations to learn model parameters from data, to enable Kalman filters to reduce the impact of noise in the data, and to use probabilistic (Bayesian) techniques to continuously update models to account for new data. Distributed computing helps address scalability as the number of data streams to be analyzed increases with the increased complexity of avionics systems. For example, due to the increasing number of sensors in aircraft, more complex aircraft fault models, error signatures, and damaged aircraft performance profiles would need to be assessed. Two methods are being investigated: high-level abstractions and uncertainty quantification. High-level abstractions [38] enable data scientists and engineers to more easily develop concurrent software to analyze


Researchers from Rensselaer Polytechnic Institute. From left to right: Carlos A. Varela, Alessandro Galli, Sida Chen, Wennan Zhu, Frederick Lee, Shigeru Imai.

data and facilitate distributed computing optimizations needed for real-time response. Finally, uncertainty quantification [39], [40] is an important future direction to associate data confidence and error estimation in support of pilot decision making.

As automated flight systems take into account terrain data, updated weather, and information from other planes, a more complete picture can be formed to assist pilots, especially in emergency conditions. Fundamental developments needed for human expert-level machine flight assistants include a combination of quantitative and qualitative spatio-temporal logics and reasoning systems, as well as stochastic reasoning. Further research in these fundamental directions will enable automated spatio-temporal situational awareness as required for computer flight assistance in emergency conditions.◆

ACKNOWLEDGEMENTS

This research is partially supported by the DDDAS program of the Air Force Office of Scientific Research Grant No. FA9550-15-1-0214, NSF Grant No. 1462342, and a Yamada Corporation Fellowship. We appreciate the thoughtful comments from Alexandra Zytek, Carlos Gomez, David Glowny, and the reviewers which led to improvements to the PILOTS software.

REFERENCES

[1] Oliva, R., Blasch, E., and Ogan, R. Applying aerospace technologies to current issues using systems engineering: 3rd AESS chapter summit. *IEEE Aerospace and Electronic Systems Magazine*, Vol. 28, 2 (Feb. 2013), 34–41.

[2] Byington, C. S., Kalgren, P. W., Johns, R., and Beers, R. J. Embedded diagnostic/prognostic reasoning and information continuity for improved avionics maintenance. In *Proceedings of the AUTOTESTCON 2003 IEEE Systems Readiness Technology Conference*, 2003, 320–329.

[3] Blasch, E. P., Lambert, D., Valin, P., Kokar, M. M., Llinas, J., Das, S., et al. High Level Information Fusion (HLIF): Survey of models, issues, and grand challenges. *IEEE Aerospace and Electronic Systems Magazine*, Vol. 27, 9 (Sep. 2012), 4–20.

[4] Xu, J., and Xu, L. Health management based on fusion prognostics for avionics systems. *Journal of Systems Engineering and Electronics*, Vol. 22, 3 (Jun. 2011), 428–436.

[5] Ananda, C. M. General aviation aircraft avionics: Integration & system tests. *IEEE Aerospace and Electronic Systems Magazine*, Vol. 24, 5 (May 2009), 19–25.

[6] Jacobson, S. Aircraft loss of control causal factors and mitigation challenges. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2010.

[7] Rolfsen, B. Moisture confused sensors in B-2 Crash. *Air Force Times*, Jun. 2008.

[8] B-2A, S/N 89-0127, 20080223 KSZL501A, accessed as: AFD-080605-054.pdf, 2015.

[9] Bureau D'Enquetes et d'Analyses (BEA), Final report on the accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro–Paris. *Ministère l'École du Développement Durable des Transports et du Logement*, 2012.

[10] Mokbel, M. F., Xiong, X., Hammad, M. A., and Aref, W. G. Continuous query processing of spatio-temporal data streams in PLACE. *Geoinformatica*, Vol. 9, 4 (Nov. 2005), 343–365.

[11] Ali, M., Chandramouli, B., Sethu, B., and Katibah, R. Spatio-temporal stream processing in Microsoft StreamInsight. *IEEE Data Engineering Bulletin*, 2010, 69–74.

[12] Chakravarthy, S., Aved, A., Shirvani, S., Annappa, M., and Blasch, E. Adapting stream processing framework for video analysis. *Procedia Computer Science*, Vol. 51, (2015), 2648–2657.

[13] Imai, S., Galli, A., and Varela, C. A. Dynamic data-driven avionics systems: Inferring failure modes from data streams. *Procedia Computer Science*, Vol. 51, (2015), 1665–1674.

[14] Hwang, I., Kim, S., Kim, Y., and Seah, C. E. A survey of fault detection, isolation, and reconfiguration methods. *IEEE Transactions on Control Systems Technology*, Vol. 18, 3 (May 2010), 636–653.

[15] Fravolini, M. L., Brunori, V., Campa, G., Napolitano, M. R., and La Cava, M. Structural analysis approach for the generation of structured residuals for aircraft FDI. *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 45, 4 (Oct. 2009), 1466–1482.

[16] Khorasgani, H., Jung, D. E., Biswas, G., Frisk, E., and Krysander, M. Robust residual selection for fault detection. In *Proceedings of the 53rd IEEE Conference on Decision and Control*, 2014, 5764–5769.

[17] Gertler, J. Designing dynamic consistency relations for fault detection and isolation. *International Journal of Control*, Vol. 73, 8 (Jan. 2000), 720–732.

[18] Trave-Massuyes, L., Escobet, T., and Olive, X. Diagnosability analysis based on component-supported analytical redundancy relations. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, Vol. 36, 6 (Nov. 2006), 1146–1160.

[19] Blanke, M., and Hansen, S. Towards self-tuning residual generators for UAV control surface fault diagnosis. In *Proceedings of the 2013 Conference on Control and Fault-Tolerant Systems (SysTol)*, 2013, 37–42.

[20] Zolghadri, A. Advanced model-based FDIR techniques for aerospace systems: Today challenges and opportunities. *Progress in Aerospace Sciences*, Vol. 53, (Aug. 2012), 18–29.

[21] Menke, T. E., and Maybeck, P. S. Sensor/actuator failure detection in the Vista F-16 by multiple model adaptive estimation. *IEEE Transac-*

tions on Aerospace and Electronic Systems, Vol. 31, 4 (Oct. 1995), 1218–1229.

[22] Hansen, S., and Blanke, M. Diagnosis of airspeed measurement faults for unmanned aerial vehicles. *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 50, 1 (Jan. 2014), 224–239.

[23] Marzat, J., Piet-Lahanier, H., Damongeot, F., and Walter, E. Model-based fault diagnosis for aerospace systems: A survey. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 226, 10 (Jan. 2012), 1329–1360.

[24] Imai, S., Klockowski, R., and Varela, C. A. Self-healing spatio-temporal data streams using error signatures. In *Proceedings of the 2013 IEEE 16th International Conference on Computational Science and Engineering*, 2013, 957–964.

[25] Imai, S., and Varela, C. A. Programming spatio-temporal data streaming applications with high-level specifications. In *Proceedings of the Third ACM SIGSPATIAL International Workshop on Querying and Mining Uncertain Spatio-Temporal Data—QUeST'12*, 2012, 18–25.

[26] Darema, F. Dynamic data driven applications systems: A new paradigm for application simulations and measurements. In *Computational Science-ICCS* 2004, pages 662–669. Springer, 2004.

[27] Svärd, C., Nyberg, M., Frisk, E., and Krysander, M. Data-driven and adaptive statistical residual evaluation for fault detection with an automotive application. *Mechanical Systems and Signal Processing*, Vol. 45, 1 (2014), 170–192.

[28] Jung, D., Eriksson, L., Frisk, E., and Krysander, M. Development of misfire detection algorithm using quantitative FDI performance analysis. *Control Engineering Practice*, Vol. 34, (2015), 49–60.

[29] Cho, A., Kim, J., Lee, S., and Kee, C. Wind estimation and airspeed calibration using a UAV with a single-antenna GPS receiver and pitot tube. *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 47, 1 (Jan. 2011), 109–117.

[30] ANSV, Agenzia Nazionale per la Sicurezza del Volo. Final Report: Accident involving ATR 72 aircraft registration marks TL-LBB ditching off the coast of Capo Gallo (Palermo—Sicily). 2005.

[31] ATR. *ATR72—Flight Crew Operating Manual, Aerei da Trasporto Regionale*. 1999.

[32] Allaire, D., Kordonowy, D., Lecerf, M., Mainini, L., and Willcox, K. Multifidelity DDDAS methods with application to a self-aware aerospace vehicle. *Procedia Computer Science*, Vol. 29, (2014), 1182–1192.

[33] Blasch, E., Laskey, K. B., Jousselme, A. L., Dragos, V., Costa, P. C. G., and Dezert, J. URREF reliability versus credibility in information fusion (STANAG 2511). *), 2013 16th International Conference on Information Fusion (FUSION)*, 2013, 1600–1607.

[34] Paces, P., Jalovecky, R., Blasch, E., and Stanek, J. Pilot controller design using the CTU flight simulator for shared situation awareness. In *Proceedings of the 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, 2015, 3E3–1–3E3–10.

[35] Maghraoui, K. E., Desell, T. J., Szymanski, B. K., and Varela, C. A. The Internet operating system: Middleware for adaptive distributed computing. *International Journal of High Performance Computing Applications*, Vol. 20, 4 (Nov. 2006), 467–480.

[36] Imai, S., Chestna, T., and Varela, C. A. Elastic scalable cloud computing using application-level migration. In *Proceedings of the 2012 IEEE Fifth International Conference on Utility and Cloud Computing*, 2012, 91–98.

[37] Chen, S., Imai, S., Zhu, W. and Varela C. A. Towards Learning Spatio-Temporal Data Stream Relationships for Failure Detection in Avionics. In *Dynamic Data-Driven Application Systems* (DDDAS 2016), Hartford, CT, August 2016.

[38] Varela, C. A. *Programming Distributed Computing Systems: A Foundational Approach*. Cambridge, MA: The MIT Press, 2013.

[39] Prudencio, E. E., Bauman, P. T., Williams, S., Faghihi, D., Ravi-Chandar, K., and Oden, J. T. Real-time inference of stochastic damage in composite materials. *Composites Part B: Engineering* Vol. 67, (Dec. 2014), 209–219.

[40] Allaire, D. and Willcox, K., A mathematical and computational framework for multifidelity design and analysis with computer models, *International Journal for Uncertainty Quantification*, 4(1):1–20, 2014.