Self-Healing Spatio-Temporal Data Streams Using Error Signatures

Shigeru Imai, Richard Klockowski, Carlos A. Varela
Department of Computer Science, Rensselaer Polytechnic Institute
110 Eighth Street, Troy, NY 12180, USA
Email: {imais,klockr,cvarela}@cs.rpi.edu

***Abstract*—Self-healing spatio-temporal data streaming systems enable error detection and data correction based on *error signatures*. Error signatures are mathematical function patterns with constraints and are used to identify and categorize errors in redundant spatio-temporal data streams. In this paper, we apply these methods to real data from a private Cessna flight and from the Air France AF447 accident in June 2009. For the private Cessna flight, three error scenarios are simulated: pitot tube failure, GPS failure, and simultaneous pitot tube and GPS failures. The error detection accuracy is approximately 93% and the response time to correct data is at most 5 seconds. For the AF447 flight, 162 seconds of available flight data including the pitot tubes failure is collected from the accident report. The pitot tube failure of the AF447 flight is successfully detected and corrected after 5 seconds from the beginning of the failure. Overall error mode detection accuracy reaches 96.31%. Furthermore, our simulations show that the system never corrects data incorrectly, *i.e.*, all inaccurate mode detections produce either unknown or unrecoverable errors.**

I. Introduction

Spatio-temporal data streams generated from sensors can be erroneous and could lead to serious problems. For example, pitot tubes icing which occurred to Air France flight 447 (AF447) in June 2009 led to faulty airspeed readings and eventually caused a fatal accident killing all 228 people on board. Airplanes are one of the most complicated machines to operate since pilots have to deal with a lot of information provided from the instruments in a cockpit. In the event of instrument failures, making the right decision becomes even more difficult because of potentially partially erroneous data. In the worst case scenario, misinterpreting the data could lead to deadly accidents such as the Air France flight 447 [1].

The aircraft of the AF447 flight crashed in the Atlantic Ocean due to ice which temporarily formed in the pitot tubes causing erroneous airspeed readings, and the subsequent inability of the auto-pilot and human pilots to recover. The accident could have been prevented by endowing the flight system with the ability to understand the following data relationship:

$$\overrightarrow{v_g} = \overrightarrow{v_a} + \overrightarrow{v_w}. \quad (1)$$

where $\overrightarrow{v_g}$, $\overrightarrow{v_a}$, and $\overrightarrow{v_w}$ represent the *ground speed*, the *airspeed*, and the *wind speed* vectors. These speeds are obtained through *independent* data collection methods: the ground speed is typically computed from Global Positioning System (GPS) data, the airspeed is computed from 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), they are *redundant* to each other. Using the available redundancy in the data, we can detect and correct errors. Note that we use this speed example throughout the paper. Our error signature-based detection and correction methods can fix erroneous data readings caused by sensor failures within a few seconds and thereby keep flight systems working properly. Such self-healing flight systems could have prevented the tragic AF447 accident from happening and saved the lives of all crew members and passengers.

We have created a highly declarative programming language called *PILOTS* (**P**rogramm**I**ng **L**anguage for spati**O**-Temporal **S**treaming applications) [2], [3], [4] that enables data correction and detection of *spatio-temporal* data streams based on data redundancy. Spatio-temporal data streams refer to data streams whose items include associated spatial and temporal coordinates, often viewed as meta data. Examples include temperature measurements, financial stock values, gas prices, surveillance camera imaging, and aircraft sensor readings. A PILOTS program may specify 1) how to view heterogeneous data stream sources as homogeneous spatio-temporal data streams, 2) how to correct the data streams based on *error signatures*, and 3) how to output values of interest based on the corrected data streams. Error signatures are mathematical function patterns with constraints and are used to stochastically identify and categorize errors. The PILOTS programming language enables high-level development of applications to handle spatio-temporal data streams and ultimately assist humans in making better decisions.

The PILOTS project has evolved gradually to date. First, the design of the PILOTS programming language and the concept of error signatures were proposed [2]. Next, a runtime implementation of PILOTS capable of data selection and error signatures computation was presented [3]. Thirdly, an error detection method and a runtime implementation of PILOTS with error detection and correction capability were presented [4]. In this paper, we overview PILOTS version 0.2.3 [5] and mathematically refine the error signature-based detection and data correction methods. Also, we evaluate error detection performance with real data from a private Cessna flight and from the AF447 flight.

The rest of the paper is organized as follows. Section II

describes technical background of the paper including methods and software for error detection and correction. Section III talks about error signatures for commonly used speed data in aviation and how to express these error signatures in PILOTS programs. Section IV shows performance metrics and results of error detection performance for a private Cessna flight and the AF447 flight data. Finally, we show related work in Section V and conclude the paper in Section VI with potential future directions.

II. Technical Background

A. Error Detection and Correction Methods

The error detection and correction methods [4] are refined and described in detail. The basic idea is that the algorithm recognizes the shape of an error function, identifies a type of error, and corrects associated data values if possible.

Error function An error function is an arbitrary 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.

A vector $\overrightarrow{v}$ can be defined by a tuple (v, α), where v is the length of $\overrightarrow{v}$ and α is the angle between $\overrightarrow{v}$ and a base vector. Following this expression, $\overrightarrow{v_g}, \overrightarrow{v_a}$, and $\overrightarrow{v_w}$ are defined as $(v_g, \alpha_g), (v_a, \alpha_a)$, and (v_w, α_w) respectively as shown in Figure 1. To examine the relationship in Equation (1), we can compute $\overrightarrow{v_g}$ by applying trigonometry to $\triangle ABC$. We can define an error function as the difference between measured v_g and computed v_g as follows:

$$\begin{aligned} e(\overrightarrow{v_g}, \overrightarrow{v_a}, \overrightarrow{v_w}) &= |\overrightarrow{v_g} - (\overrightarrow{v_a} + \overrightarrow{v_w})| \\ &= v_g - \sqrt{v_a^2 + 2v_a v_w \cos(\alpha_a - \alpha_w) + v_w^2}. \end{aligned} \tag{2}$$

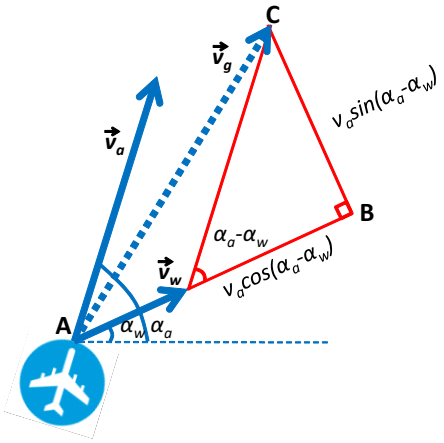

Fig. 1. Trigonometry applied to the ground speed, airspeed, and wind speed.

The values of input data are assumed to be sampled periodically from corresponding spatio-temporal data streams. Thus, an error function e changes its value as time proceeds and can also be represented as $e(t)$.

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

$$S(\bar{K}, f(t), \bar{P}(\bar{K})) = \{f(t) | p_1(\bar{K}) \wedge \cdots \wedge p_l(\bar{K})\}. \tag{3}$$

For example, an interval error signature can be defined as:

$$\begin{aligned} S_I(\bar{K}, f(t), \bar{I}(\bar{K}, \bar{A}, \bar{B})) \quad = \quad & \{f(t) | \\ & a_1 \leq k_1 \leq b_1, \ldots \\ & a_m \leq k_m \leq b_m\}, \end{aligned} \tag{4}$$

where $\bar{A} = \langle a_1, \ldots, a_m \rangle$ and $\bar{B} = \langle b_1, \ldots, b_m \rangle$. When $f(t) = t + k, \bar{K} = \langle k \rangle, \bar{A} = \langle 2 \rangle$, and $\bar{B} = \langle 5 \rangle$, the error signature S_I contains all linear functions with slope 1, and crossing the Y-axis at values $[2, 5]$ as shown in Figure 2. On the other hand, for $f(t) = 0$, S_I only contains the constant function $f(t) = 0$.

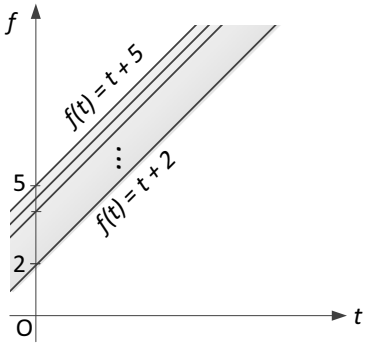

Fig. 2. Error signature S_I with a linear function $f(t) = t + k, 2 \leq k \leq 5$.

Given an error signature $S(\bar{K}, f(t), \bar{P}(\bar{K}))$, we enumerate its elements as *error signature samples*, *i.e.*,

$$s(t, \bar{K}) = f(t) \text{ s.t. } s(t, \bar{K}) \in S(\bar{K}, f(t), \bar{P}(\bar{K})). \tag{5}$$

An error signature sample is thus a particular function satisfying the constraints defined by an error signature. For the interval error signature S_I, a sample $s_I(t, \langle 3 \rangle)$ is $f(t) = t + 3$.

Mode likelihood vectors Given a set of error signatures $\{S_0, \ldots, S_n\}$, we calculate $\delta_i(t)$, the distance between the measured error function $e(t)$ and each error signature S_i by:

$$\delta_i(t) = \min_{\bar{K}} \int_{t-\omega}^{t} |e(t) - s_i(t, \bar{K})| dt. \tag{6}$$

where ω is the window size and $s_i(t, \bar{K}) \in S_i$. Note that our convention is to capture "normal" conditions as signature S_0. The smaller the distance $\delta_i(t)$, the closer the raw data is to the theoretical signature S_i. We define the *mode likelihood vector* as $L(t) = \langle l_0(t), l_1(t), \ldots, l_n(t) \rangle$ where each $l_i(t)$ is defined as:

$$l_i(t) = \begin{cases} 1, & \text{if } \delta_i(t) = 0 \\ \frac{\min\{\delta_0(t), \ldots, \delta_n(t)\}}{\delta_i(t)}, & \text{otherwise.} \end{cases} \tag{7}$$

Observe that for each $l_i \in L, 0 < l_i \leq 1$ where l_i represents the ratio of the likelihood of signature S_i being matched with respect to the likelihood of the best signature. At each time stamp, the maximum two elements l_i and l_j of the mode likelihood vector, where $l_i \geq l_j$, are inspected in order to determine the error mode. Because of the way $L(t)$ is created, the maximum entry l_i will always be equal to 1. Given a threshold $\tau \in (0, 1)$ we check for one likely candidate that is sufficiently more likely than its successor by ensuring that $l_j \leq \tau$. Thus we determine the correct mode by choosing the error signature, and error mode i, corresponding to l_i which is S_i. If $i = 0$ then the system is in normal mode. If $l_j > \tau$, then regardless of the value of j, *unknown error* mode (-1) is assumed.

Error correction 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 example used in Equations (1) and (2), if the ground speed v_g is detected as erroneous, its corrected value v_g^c can be computed by the airspeed and wind speed as follows:

$$v_g^c = \sqrt{v_a^2 + 2v_a v_w \cos(\alpha_a - \alpha_w) + v_w^2}. \quad (8)$$

B. Error Detection and Correction Software

PILOTS (**P**rogramm**I**ng **L**anguage for spati**O**-**T**emporal data **S**treaming applications) is a programming language specifically designed for analyzing data streams incorporating space and time. Using PILOTS, application developers can easily program an application that handles spatio-temporal data streams by writing a high-level (declarative) program specification. The PILOTS code includes an *inputs* section to specify the data streams and how data is to be extrapolated from incomplete data, typically using declarative geometric criteria (*e.g.*, *closest*, *interpolate*, *euclidean* keywords) [3]. It includes *outputs* and *errors* sections to specify the data streams to be produced by the application, as a function of the input streams with a given frequency. If a detected error is recoverable, output values are computed from corrected input data, otherwise original input data is used. The *signatures* and *correct* sections, enable PILOTS programmers to specify error signatures for known error conditions, as well as the function to use to correct the data automatically if such data errors are found.[1]

Figure 3 shows the architecture of the PILOTS runtime system, which implements the error detection and correction methods described in the previous section. It consists of three parts: the *Data Selection*, the *Error Analyzer*, and the *Application Model* modules.

The Application Model obtains homogeneous data streams $(d'_1, d'_2, \ldots, d'_N)$ from the Data Selection module, and then it generates outputs $(o_1, o_2, \ldots, o_M)$ and data errors $(e_1, e_2, \ldots, e_L)$. The Data Selection module takes heterogeneous incoming data streams $(d_1, d_2, \ldots, d_N)$ as inputs. Since this runtime is assumed to be working on moving objects, the Data Selection module is aware of the current location and time. Thus, it returns appropriate values to the Application Model by selecting or interpolating data in time and location depending on the data selection method specified in the PILOTS program.

The ErrorAnalyzer collects the latest ω error values from the Application Model and keeps analyzing errors based on the error signatures. If it detects a recoverable error, then it replaces an erroneous input with the corrected one by applying a corresponding error correction equation. The Application Model computes the outputs based on the corrected inputs produced from the Error Analyzer.

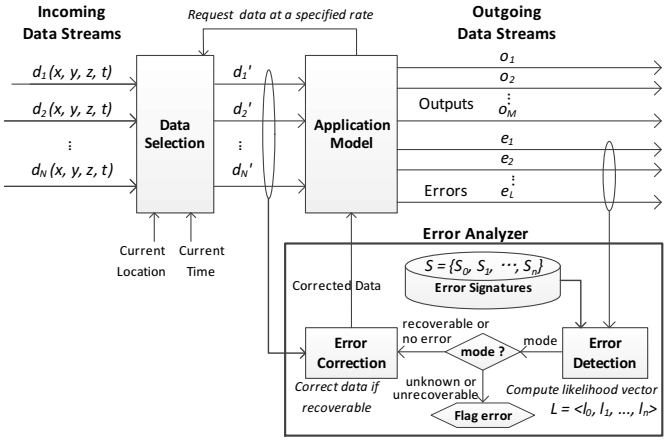

Fig. 3. Data streaming architecture with error detection and correction.

III. Error Signatures for Self-Healing Speed Data

In this section, we derive a set of error signatures for the speed example used in the previous sections. Also, we present a PILOTS program implementing the error signatures and corresponding error correction equations.

A. Error Signatures

We consider the following four error modes: 1) normal (no error), 2) pitot tube failure due to icing, 3) GPS failure, 4) both pitot tube and GPS failures. Suppose the airplane is flying at airspeed v_a. For computing error signatures for different error conditions, we will assume that other speeds as well as failed airspeed and ground speed can be expressed as follows.

- ground speed: $v_g \approx v_a$.
- wind speed: $v_w \leq a v_a$, where a is the wind to airspeed ratio.
- pitot tube failed airspeed: $b_l v_a \leq v_a^f \leq b_h v_a$, where b_l and b_h are the lower and higher values of pitot tube clearance ratio and $0 \leq b_l \leq b_h \leq 1$. 0 represents a fully clogged pitot tube, while 1 represents a fully clear pitot tube.
- GPS failed ground speed: $v_g^f = 0$.

[1]Parameters τ and ω—for specifying threshold and time window respectively—can be given in command-line options.

We assume that when a pitot tube icing occurs, it is gradually clogged and thus the airspeed data reported from the pitot tube also gradually drops and eventually remains at a constant speed while iced. This resulting constant speed is characterized by ratio b_l and b_h. On the other hand, when a GPS failure occurs, the ground speed suddenly drops to zero. This is why we model the failed ground speed as $v_g^f = 0$.

In the case of pitot tube failure, let the ground speed, wind speed, and airspeed be $v_g = v_a, v_w = av_a$, and $v_a^f = bv_a$. The error function (2) can be expressed as follows:

$$e = v_a - \sqrt{v_a^2(b^2 + 2ab\cos(\alpha_a - \alpha_w) + a^2)}.$$

Since $-1 \leq \cos(\alpha_a - \alpha_w) \leq 1$, the error is bounded by the following:

$$\begin{aligned} v_a - \sqrt{v_a^2(a+b)^2} \leq \quad & e \quad \leq v_a - \sqrt{v_a^2(a-b)^2} \\ (1-a-b)v_a \leq \quad & e \quad \leq (1-|a-b|)v_a. \end{aligned} \tag{9}$$

In the case of GPS failure, let the ground speed, wind speed, and airspeed be $v_g^f = 0, v_w = av_a$, and $v_a = v_a$. The error function (2) can be expressed as follows:

$$e = 0 - \sqrt{v_a^2(1 + 2a\cos(\alpha_a - \alpha_w) + a^2)}.$$

Similarly to the pitot tube failure, we can derive the following error bounds:

$$-(a+1)v_a \leq \quad e \quad \leq -|a-1|v_a. \tag{10}$$

We can derive error bounds for the normal and both failure cases similarly. Applying the wind to airspeed ratio a and the pitot tube clearance ratio $b_l \leq b \leq b_h$ to the constraints obtained in Inequations (9) and (10), we get the error signatures for each error mode as shown in Table I.

TABLE I
ERROR SIGNATURES FOR SPEED DATA.

Mode	Error Signature	
	Function	Constraints
Normal	$e = k$	$k \in [-av_a, av_a]$
Pitot tube failure	$e = k$	$k \in [(1-a-b_h)v_a, (1-\lvert a-b_l\rvert)v_a]$
GPS failure	$e = k$	$k \in [-(a+1)v_a, -\lvert a-1\rvert v_a]$
Both failures	$e = k$	$k \in [-(a+b_h)v_a, -\lvert a-b_l\rvert v_a]$

When $a = 0.1, b_l = 0.2$, and $b_h = 0.33$, the error signatures shown in Table I are visually depicted in Figure 4.

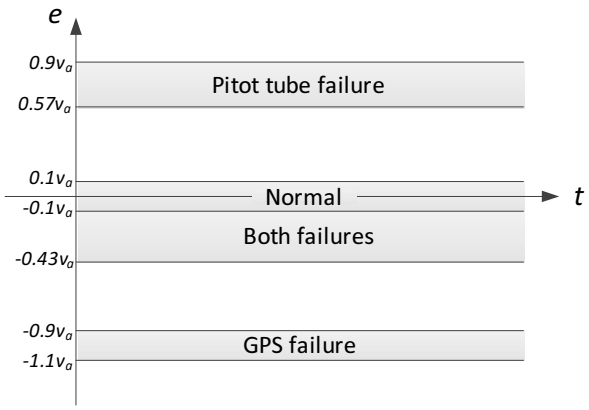

Fig. 4. Error Signatures for speed data ($a = 0.1, b_l = 0.2$, and $b_h = 0.33$).

B. PILOTS program

A PILOTS program called `speedcheck` implementing the error signatures shown in Table I is presented in Figure 5. This program checks if the wind speed, airspeed, and ground speed are correct or not, and computes a crab angle, which is used to adjust the direction of the aircraft to keep a desired ground track. For this program to be applicable to a Cessna 182-RG, we use a cruise speed of 162 knots as v_a. Each section of the program is explained in order:

- *inputs:* All the speed and angle data required to compute the error and crab angle are defined here with data selection methods. Since heterogeneous input data streams of `air_speed, air_angle, ground_speed` and `ground_angle` are defined for 2D regions and time, `euclidean(x,y)` and `closest(t)` select data which is closest to the current location in 2D euclidean space and then closest to the current time. For `wind_speed` and `wind_angle`, since they are defined for 3D regions and time, `interpolate(z,2)` is finally used to get linearly interpolated values in the Z-axis using two data points after `euclidean(x,y)` and `closest(t)` are applied.
- *outputs:* The crab angle is computed every second.
- *errors:* The error function `e` defined in Equation (2) is computed. The angle signs are reversed in the formulae, because in mathematics, angles increase counterclockwise (with 0° representing East) while in aviation, angles increase clockwise (with 0° representing North).
- *signatures:* There are four error signatures {`S0, S1, S2, S3`} associated with the error function `e`. They are all constrained by a constant `k` with lower and upper bounds based on the error signatures shown in Table I.
- *correct:* The error modes 1 and 2, which are identified by `S1` and `S2`, can be corrected using the equations defined for the airspeed and ground speed. If the error mode 3 corresponding to `S3` is detected, it is not possible to correct two variables at the same time, thus this error is unrecoverable.

IV. EVALUATION

We apply the error signatures defined in Section III to two sets of real flight data. The first one is a private flight using a Cessna 182-RG identified by N756VH [6] from Albany, NY to Fort Meade, MD on April 3rd, 2012. The other is the Air France flight 447 using an Airbus A330-203 which took off from Rio de Janeiro bound for Paris on June 1st, 2009. To simulate the failures mentioned in Section III, we added corresponding errors to the N756VH Cessna flight data; however, we used the real pitot tube failure data for the

```
program speedcheck;
  inputs
    wind_speed, wind_angle (x,y,z,t) using
      euclidean(x,y), closest(t), interpolate(z,2);
    air_speed, air_angle (x,y,t) using
      euclidean(x,y), closest(t);
    ground_speed, ground_angle (x,y,t) using
      euclidean(x,y), closest(t);

  outputs
    crab_angle:
      arcsin(wind_speed * sin(wind_angle - air_angle) /
        sqrt(air_speed^2 + 2 * air_speed * wind_speed *
          cos(wind_angle - air_angle) + wind_speed^2))
            at every 1 sec;

  errors
    e: ground_speed -
         sqrt(air_speed^2 + wind_speed^2 + 2 * air_speed *
           wind_speed * cos(wind_angle - air_angle));

  signatures
    /* v_a = 162 knots */
    S0(k): e=k,  -16.2<=k, k<=  16.2 "Normal";
    S1(k): e=k,   91.8<=k, k<= 145.8 "Pitot tube failure";
    S2(k): e=k, -178.2<=k, k<=-145.8 "GPS failure";
    S3(k): e=k,  -70.2<=k, k<= -16.2 "Both failures";

  correct
    S1: air_speed = sqrt(ground_speed^2 + wind_speed^2 +
          2 * ground_speed * wind_speed *
            cos(ground_angle - wind_angle));
    S2: ground_speed = sqrt(air_speed^2 + wind_speed^2 +
          2 * air_speed * wind_speed *
            cos(wind_angle - air_angle));
end
```

Fig. 5. A declarative specification of the speedcheck PILOTS program.

AF447 flight. PILOTS programs' error detection accuracy and response time to mode changes are evaluated.

A. Performance Metrics

- **Accuracy**: This metric is used to evaluate how accurately the algorithm determines the true mode. Assuming the true mode transition $m(t)$ is known for $t = 0, 1, 2, \ldots, T$, let $m'(t)$ for $t = 0, 1, 2, \ldots, T$ be the mode determined by the error detection algorithm. We define $accuracy(m, m') = \frac{1}{T}\sum_{t=0}^{T} p(t)$, where $p(t) = 1$ if $m(t) = m'(t)$ and $p(t) = 0$ otherwise.
- **Maximum/Minimum/Average Response Time**: This metric is used to evaluate how quickly the algorithm reacts to mode changes. Let a tuple (t_i, m_i) represent a mode change point, where the mode changes to m_i at time t_i. Let $M = \{(t_1, m_1), (t_2, m_2), \ldots, (t_N, m_N)\}$ and $M' = \{(t'_1, m'_1), (t'_2, m'_2), \ldots, (t'_{N'}, m'_{N'})\}$ be the sets of true mode changes and detected mode changes respectively. For each $i = 1 \ldots N$, we can find the smallest t'_j such that $(t_i \leq t'_j) \wedge (m_i = m'_j)$; if not found, let t'_j be t_{i+1}. The response time r_i for the true mode m_i is given by $t'_j - t_i$. We define the maximum, minimum, and average response times by $\max_{1 \leq i \leq N} r_i$, $\min_{1 \leq i \leq N} r_i$, and $\frac{1}{N}\sum_{i=1}^{N} r_i$ respectively.

B. Experiment 1: N756VH Cessna Flight

1) Flight data: Flight data is collected through the following independent sources:

- **ground speed**: Flight track log provided by FlightAware [6].
- **airspeed**: Manually recorded by the pilot.
- **wind speed**: Weather forecast information provided by National Weather Service [7].

The flight duration is 1 hour 41 minutes. The collected speed data and error computed by Equation (2) are shown in Figure 6. Notice that the airspeed data during take off and landing is not accurate due to the data collection mechanism.

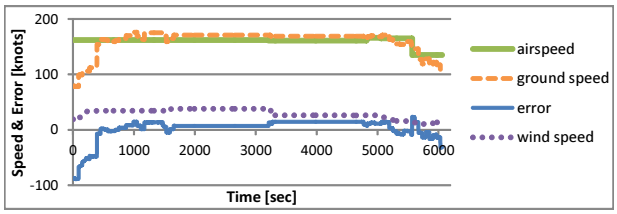

Fig. 6. Collected speeds and error for the N756VH 03-Apr-2012 KALB-KFME flight (normal).

2) Experimental Settings: Using the speedcheck PILOTS program shown in Figure 5, the 6060 seconds (=1 hour 41 minutes) of flight departing from Albany, NY and landing at Fort Meade, MD are recreated. Three types of error are simulated as shown below. In each case, all data streams except for erroneous one(s) are actual. Defined error modes are: -1 for unknown, 0 for normal, 1 for pitot tube failure, 2 for GPS failure, and 3 for both failures.

- **Pitot tube failure**: 2400 seconds after the departure, the airspeed drops from 162 knots to 50 knots within 10 seconds and stays at 50 knots until landing. The set of true mode changes is given by $M = \{(1, 0), (2401, 1)\}$.
- **GPS failure**: 2400 seconds after the departure, the ground speed drops from 171 knots to 0 knots immediately and stays at 0 knots until landing. The set of true mode changes is given by $M = \{(1, 0), (2401, 2)\}$.
- **Both pitot tube and GPS failures**: The above two speed changes happen simultaneously at 2400 seconds after the departure. Both speeds remain failed until landing. The set of true mode changes is given by $M = \{(1, 0), (2401, 3)\}$.

To find out the effect of the window size ω and threshold value τ on the accuracy and response time, we measure these metrics for window sizes $\omega \in \{1, 2, 4, 8, 16\}$ and threshold $\tau \in \{0.2, 0.4, 0.6, 0.8\}$. Note that since there is only one error mode change in each true mode changes set, we can get only one response time result for each simulated error case.

3) Results: Results of the accuracy and response time are are shown in Figure 7. For all the three cases, when $\omega = 1$ and $\tau = 0.8$, the best results are observed as follows: accuracy = 0.9294 and response time = 4 seconds for the pitot tube failure, accuracy = 0.935 and response time = 0 seconds for the GPS failure, and accuracy = 0.9342 and response time = 5 seconds for both failures. Accuracy is not even higher due to airspeed data during takeoff and landing which was not

collected because the pilot was busy operating the airplane, which makes the system incorrectly detect a both failure mode. Because the airspeed gradually drops, it takes a few seconds to detect it as a pitot tube failure; however, a GPS failure is immediately detected since the ground speed promptly drops to zero when it happens. This is why the response time for the GPS failure is better than the other two cases. Since the used error signature sets are non-overlapping constant functions (*i.e.*, $e = k$), even though smaller window sizes are normally noise-prone compared to bigger window sizes, past data is not necessary to determine the correct error modes. In this experiment, noise on the error is not big enough to jump out of the boundaries defined by error signature sets, therefore $\omega = 1$ gives the best results.

In Figure 7(b-1) for the GPS failure, when $\tau = 0.2$, accuracy is unusually low compared to the other two failure cases. This occurs because too low a threshold makes the normal and GPS failure modes compete against each other in the landing phase and thus the resulting mode falls into unknown mode for the last 600 seconds.

The transitions of the corrected speed and detected modes that show the best accuracy are shown in Figures 8 (pitot tube failure), 9 (GPS failure), and 10 (both failures) respectively. For the first 390 seconds, the error mode is detected wrongly in all three cases; the true modes are 0 (normal mode) whereas the detected modes are 3 (both failures) during this period. These wrong mode detections are originated from the erroneously recorded airspeed. Other than that, the error detection method works pretty well for all three cases.

Detected modes go into the unknown mode for a short period around 2401 seconds for both pitot tube failure and both failures. Since the airspeed takes a few seconds to drop, during that time, the normal and pitot tube failure modes are competing against each other for the pitot tube failure case. For the both failures case, the GPS failure and both failures modes are competing. Unlike the other two cases, the ground speed drops immediately for the GPS failure, and there is no conflict with other error modes, thus the GPS failure mode is correctly detected without going into the unknown mode.

C. Experiment 2: Air France Flight 447

1) Flight Data: The ground speed and airspeed are collected based on Appendix 3 in the final report of Air France flight 447 [1]. Note that the (true) airspeed was not recorded in the flight data recorder so that we computed it 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.

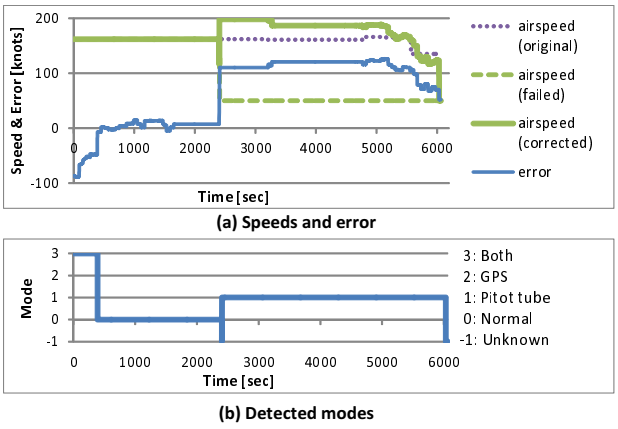

Fig. 8. Corrected airspeed and detected modes for the N756VH 03-Apr-2012 KALB-KFME flight (pitot tube failure, $\tau = 0.8, \omega = 1$).

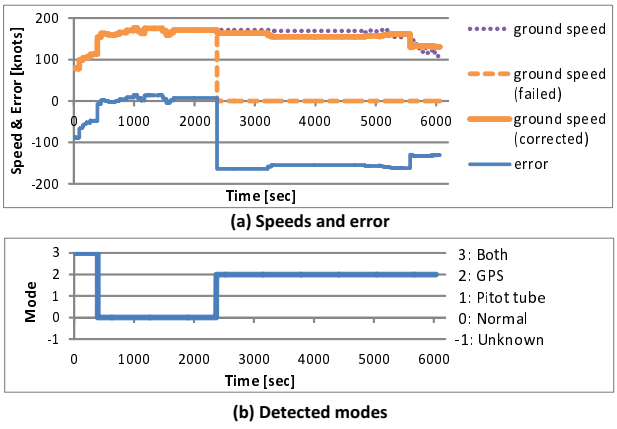

Fig. 9. Corrected ground speed and detected modes for the N756VH 03-Apr-2012 KALB-KFME flight (GPS failure, $\tau = 0.8, \omega = 1$).

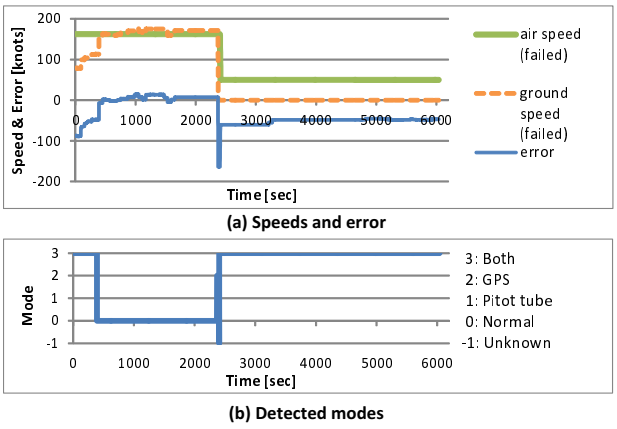

Fig. 10. Uncorrected speeds and detected modes for the N756VH 03-Apr-2012 KALB-KFME flight (pitot tube and GPS failure, $\tau = 0.8, \omega = 1$).

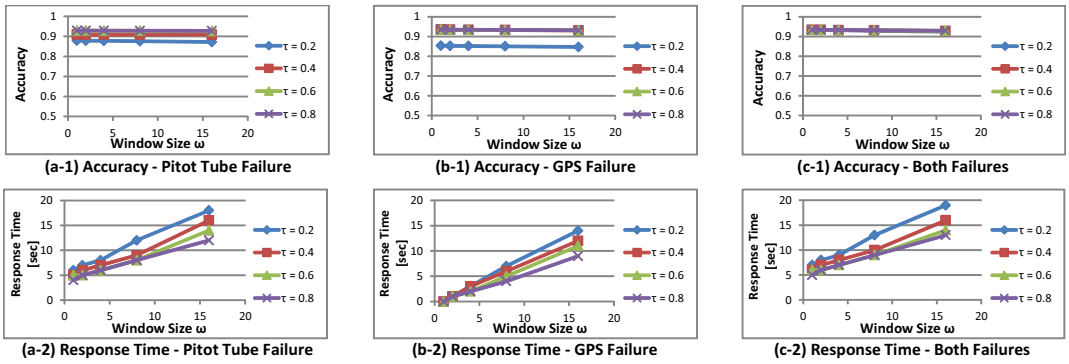

Fig. 7. Accuracy and response time for the N756VH 03-Apr-2012 KALB-KFME flight

2) Experimental Settings: According to the final report, 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 also use the `speedcheck` PILOTS program shown in Figure 5 except for constraints values in *signatures* which use $v_a = 470$ knots, the cruise airspeed of the AF447 flight. Defined error modes are the same as Experiment 1, so the set of true mode changes is defined as $M = \{(1,0),(64,1),(98,0)\}$. The accuracy and average response time are investigated for window sizes $\omega \in \{1,2,4,8,16\}$ and threshold $\tau \in \{0.2,0.4,0.6,0.8\}$.

3) Results: Results of the accuracy and maximum/minimum/average response times are shown in Figure 11. Same as Experiment 1, the best results, accuracy = 0.9631, maximum/minimum/average response times = 5/0/2.5 seconds, are observed when $\omega = 1$ and $\tau = 0.8$. Overall trends of the accuracy and response time are same as Experiment 1 because of the nature of the error signature set.

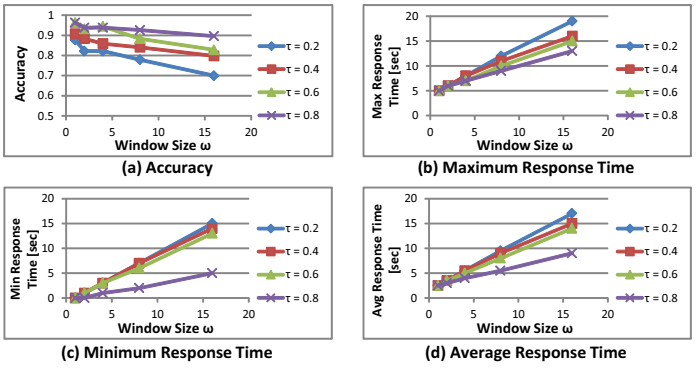

Fig. 11. Accuracy and response time for AF447 flight.

The transitions of the corrected speed and detected modes that show the best accuracy with $\omega = 1$ and $\tau = 0.8$ are shown in Figure 12. Looking at Figure 12(b), the pitot tube failure is successfully detected from 69 to 97 seconds except for the interval 64 to 69 seconds due to the slowly decreasing airspeed. The response time for the normal to pitot tube failure mode is 5 seconds and for the pitot tube failure to normal mode is 0 seconds (thus the average response time is 2.5 seconds). From Figure 12(a), the airspeed successfully starts to get corrected at 69 seconds and seamlessly transitions to the normal airspeed when it recovers at 98 seconds.

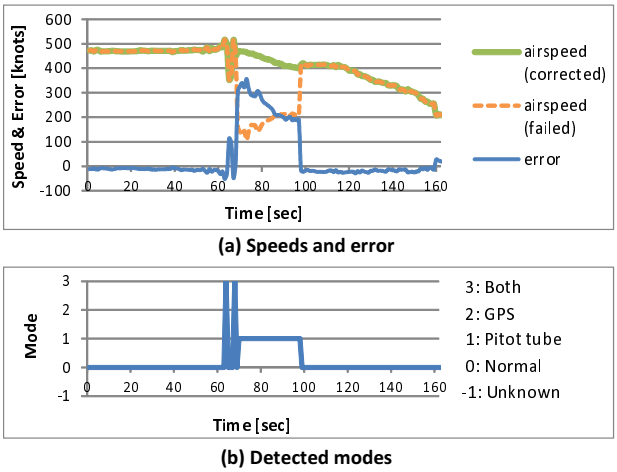

Fig. 12. Corrected airspeed and detected modes for AF447 flight.

V. Related Work

There are several systems that combine stream processing and data base management, *i.e.*, Data Stream Management Systems or DSMS, such as STREAM [8], Aurora [9], and TelegraphCQ [10]. They are designed to execute SQL-like queries to unbounded continuous incoming data streams and output events of interest. Microsoft StreamInsight is a DSMS-based system and has been extended to support spatio-temporal streams [11]. Also, the concept of the moving object data base (MODB) which adds support for spatio-temporal data streaming to DSMS is discussed in [12]. These DSMS-based spatio-temporal stream management systems support general continuous queries for multiple moving objects. Our streaming data analytics to detect errors based on signatures and correct data on the fly is beyond the scope of a

purely declarative SQL-based query approach. Furthermore, our domain-specific approach enables highly declarative description of input-output relationships between streams, error functions, error signatures, and data correction functions using the PILOTS programming language.

Distributed streaming systems have been studied in the context of cloud computing [13], [14]. Our data error correction methods could be useful for distributed settings as well by connecting multiple distributed PILOTS programs.

VI. Conclusion and Future Directions

We define a general error signature set for aviation speed data and evaluate error detection performance of PILOTS programs with real flight data. For this particular signature set, we find that the accuracy and response time improve as the threshold τ increases. The reason of this behavior is that there are some cases in which there are two competing modes whose likelihood values are close to each other, so the mode detection algorithm tends to regard it as an unknown error mode. Higher threshold values are more tolerant to multiple competing modes, thus give better results. Unsurprisingly, there is a positive correlation between the window size and response times for all the threshold values. This is an intuitive result because the less the error detection algorithm uses past data, the more responsive it becomes to mode changes. In addition, a faster average response time leads to a better accuracy result since the error detection algorithm cannot predict mode changes, but only react to them. That is, a smaller window size implies better accuracy. This is true because our designed error signature set produces nearly orthogonal mode likelihood vectors. Also, it is noteworthy that our error detection and data correction methods never correct data incorrectly.

When computing mode likelihood vectors, time to compute distances by Equation (6) can be significant due to the exponential growth of the search space as the size of the constants set $\bar{K}$ increases. To use the presented error detection and correction methods in larger-scale real-time systems, techniques to bound the running time must be devised.

Future research directions include applying the error signature-based error correction methods to other flight accidents, *e.g.*, those due to fuel sensor reading errors. Also, uncertainty quantification [15] is an important future direction to associate confidence to data and error estimations in support of decision making. More and more data are expected to be available in cockpits in the near future [16], and thus automated data analysis systems will become even more crucial to both manned and unmanned aerial vehicles. We envision scalable smarter flight systems processing massive data in real-time by dynamically creating and connecting multiple PILOTS program instances. Such systems need to reason about spatial and temporal data and constraints and give the pilots better information to make more accurate judgments during critical moments. The presented techniques and software can be used as a promising starting point to develop these flight systems.

Acknowledgments

This research is partially supported by the DDDAS program of the Air Force Office of Scientific Research, Grant No. FA9550-11-1-0332 and a Yamada Corporation Fellowship.

References

[1] Bureau d'Enquêtes et d'Analyses pour la Sécurité de l'Aviation Civile, "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," http://www.bea.aero/en/enquetes/flight.af.447/rapport.final.en.php, 2012.

[2] S. Imai and C. A. Varela, "A programming model for spatio-temporal data streaming applications," in *Dynamic Data-Driven Application Systems (DDDAS 2012)*, Omaha, Nebraska, June 2012, pp. 1139–1148.

[3] S. Imai and C. A. Varela, "Programming spatio-temporal data streaming applications with high-level specifications," in *3rd ACM SIGSPATIAL International Workshop on Querying and Mining Uncertain Spatio-Temporal Data (QUeST) 2012*, Redondo Beach, California, USA, November 2012.

[4] R. S. Klockowski, S. Imai, C. Rice, and C. A. Varela, "Autonomous data error detection and recovery in streaming applications," in *Dynamic Data-Driven Application Systems (DDDAS 2013) Workshop*, May 2013, pp. 2036–2045.

[5] Worldwide Computing Laboratory, Rensselaer Polytechnic Institute, "The PILOTS programming language," http://wcl.cs.rpi.edu/pilots/.

[6] FlightAware, "Flight track log for N756VH on 03-Apr-2012 (KALB-KFME)," http://flightaware.com/live/flight/N756VH/history/20120403/1800Z/KALB/KFME/tracklog.

[7] NOAA's National Weather Service, "Forecast winds and temps aloft," http://aviationweather.gov/products/nws/winds/.

[8] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, K. Ito, R. Motwani, U. Srivastava, and J. Widom, "Stream: The Stanford data stream management system," in *ACM SIGMOD Conference*. Springer, 2004.

[9] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, "Aurora: a new model and architecture for data stream management," *The VLDB JournalThe International Journal on Very Large Data Bases*, vol. 12, no. 2, pp. 120–139, 2003.

[10] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, F. Reiss, and M. A. Shah, "TelegraphCQ: continuous dataflow processing," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM, 2003, pp. 668–668.

[11] M. H. Ali, B. Chandramouli, B. S. Raman, and E. Katibah, "Spatio-temporal stream processing in Microsoft StreamInsight," *IEEE Data Eng. Bull.*, pp. 69–74, 2010.

[12] K. An and J. Kim, "Moving objects management system supporting location data stream," in *Proceedings of the 4th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics*, ser. CIMMACS'05. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2005, pp. 99–104.

[13] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: Distributed stream computing platform," in *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*. IEEE, 2010, pp. 170–177.

[14] V. Gulisano, R. Jimnez-Peris, M. Patio-Martnez, C. Soriente, and P. Valduriez, "StreamCloud: An elastic and scalable data streaming system," *IEEE Trans. Parallel Distrib. Syst.*, pp. 2351–2365, 2012.

[15] D. Allaire and K. Willcox, "Surrogate modeling for uncertainty assessment with application to aviation environmental system models," *AIAA journal*, vol. 48, no. 8, pp. 1791–1803, 2010.

[16] U.S. Department of Transportation Federal Aviation Administration, "Code of federal regulations part 91.225: Automatic dependent surveillance-broadcast (ADS-B) out performance requirements to support air traffic control (ATC) service; final rule," http://www.faa.gov/regulations_policies/faa_regulations/, July 2013.