Self-Healing Spatio-Temporal Data Streams using Error Signatures

SHIGERU IMAI
RICHARD KLOCKOWSKI
CARLOS A. VARELA

WORLDWIDE COMPUTING LABORATORY
DEPARTMENT OF COMPUTER SCIENCE
RENSSELAER POLYTECHNIC INSTITUTE

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
DDDAS PROGRAM PI MEETING
PROGRAM DIRECTOR: DR. DAREMA
ARLINGTON, VA, OCTOBER 1, 2013
Outline

- **Motivation and goals**
  - Air France 447 accident
  - Dynamic Data Driven Avionics Systems

- **PILOTS**
  - Programming Language for spatio-temporal data streaming applications

- **Error detection & correction methods**
  - Error signatures

- **Experiments**
  - Cessna flight data (simulated errors)
  - AF447 flight data (actual data error)

- **Future work**
Air France Flight 447

- June 1\textsuperscript{st} 2009, Flight 447 from Rio de Janeiro to Paris
- Thunderstorm caused airspeed sensors (*pitot tubes*) to ice and fail
- Autopilot system not able to deal with data failures—disengaged
- Pilots unable to react to erroneous data in a timely manner, eventually stalling the plane into the Atlantic Ocean

![Figure 3: Pitot probe (with protection caps)](http://www.bea.aero/en/enquetes/flight.af.447/rapport.final.en.php)

![Map of Flight 447 path](http://upload.wikimedia.org/wikipedia/commons/4/4a/Air_France_Flight_447_path.png)
Primary cause of the AF447 accident: incorrect airspeed

Airspeed could have been recomputed from ground speed and wind speed

- Take advantage of *data redundancy* between independently produced inputs

\[ \text{ground speed} = \text{airspeed} + \text{wind speed} \]
Dynamic Data-Driven Avionics Systems

- Monitor input data streams; analyze for errors
  - Identify the *cause* of failures, and correct the data if possible
- Act as an assistant to pilots
  - Also for situations without humans in the loop: Autonomous UAVs, Mars rovers,…

**DDDAS Streaming Data Avionics Systems**

- **Input Data Streams** (spatio-temporal sensor data)
  - airspeed
  - wind speed
  - ground speed
  - GPS data
  - fuel levels etc…

- **Application**

- **Error Detection & Correction**

- **(Corrected) Input**

- **(Corrected) Output Data Streams**

- **Error Data and Mode (identified cause)**

- GPS is failing …
To facilitate development of smarter (flight) data streaming systems, we investigate:

1. Programming technology that can model spatio-temporal data streaming applications easily
   - *PILOTS* (Programming Language for spatio-Temporal data Streaming apps)

2. Error detection using *error signatures* and error correction based on *data redundancy*
PILOTS System Overview

PILOTS Program

Input Data Streams

\[ e = f(a,b) \]

\[ a' = g(b) \]

\[ b' = h(a) \]

Corrected Output Data Stream

\[ o = i(a',b') \]

Error Stream

Error Signatures

- No error
- Sig 1
- Sig M

Error Mode (0…M)
Outline

- Motivation and goals
  - Air France 447 accident
  - Dynamic Data Driven Avionics Systems
- PILOTS
  - Programming Language for spatio-Temporal data Streaming applications
- Error detection & correction methods
  - Error signatures
- Experiments
  - Cessna flight data (simulated errors)
  - AF447 flight data (actual data error)
- Future work
PILOTS Programming Language

- Designed for spatio-temporal data streaming applications
  - Data are related to points in space and time
    - Ex: temperatures, moving objects, traffic information, gas prices

- Highly declarative
  - Programmers specify *inputs, outputs, errors, signatures*, and *correction functions*

- First class support for input data selection
  - Interpolate (often sparse) existing data to application’s queries
    - Ex: *closest, euclidean, interpolate* methods

- Error detection and correction
  - Define error conditions as error signatures
**PILOTS: System Architecture**

- **Application Model**
  - Compute outputs and errors repeatedly

- **Data Selection**: from heterogeneous to homogeneous data
  - Selection operations to approximate data as a contiguous space

- **Error Analyzer**: error detection and correction
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;

end;
Running Example: *Twice*

```
$ glennis examples/twice 189 -4 pilots Twice -input=8888 -output=127.0.0.1:9999 -tau=0.6 -omega=10
```
Outline

- Motivation and goals
  - Air France 447 accident
  - Dynamic Data Driven Avionics Systems
- PILOTS
  Programming Language for spatiO-Temporal data Streaming applications
- Error detection & correction methods
  - Error signatures
- Experiments
  - Cessna flight data (simulated errors)
  - AF447 flight data (actual data error)
- Future work
Error Detection Algorithm Overview

1. **Error Function**
   - $e(t)$
   - $e(d_1', ..., d_n')$

2. **Measured error**
   - $t$

3. **Error Signatures**
   - No error
   - Sig 1
   - Sig M

4. **Likelihood vector**
   - $\delta = <20, 3, ..., 10>$
   - $L = <3/20, 3/3, ..., 3/10>$

5. **(1) Compute distances**
6. **(2) Convert the distances to a likelihood vector**
7. **(3) Choose the best matching signature (error mode)**

8. **Error mode (0, 1, ..., M)**
Error Function and Measurement

- Error function for speed data:

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

- Error measurement
  - Data generated by the application’s error function over a window of time \((\omega)\) most recent samples

\[e(t)\]
\[t\]
An error signature is a constrained mathematical function pattern defined as follows:

\[ S(\vec{K}, f(t), \vec{P}(\vec{K})) = \{ f(t) | p_1(\vec{K}) \land \cdots \land p_l(\vec{K}) \} \]

where,

- \( f \) : a function of time
- \( \vec{K} = \langle k_1, \ldots, k_m \rangle \) : a vector of constants
- \( \vec{P} = \{ p_1(\vec{K}), \ldots, p_l(\vec{K}) \} \) : a set of constraint predicates

An error signature sample is a particular function in an error signature

\[ s(t, \vec{K}) = f(t) \text{ s.t. } s(t, \vec{K}) \in S(\vec{K}, f(t), \vec{P}(\vec{K})) \]
Example: an interval error signature $S_I$

- $S_I(\bar{K}, f(t), \bar{I}(\bar{K}, \bar{A}, \bar{B})) = \{ f(t) | a_1 \leq k_1 \leq b_1, \ldots, a_m \leq k_m \leq b_m \}$, 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 in the green region below.

- An error signature sample:

$$s_I(t, \langle 3 \rangle) \text{ is } f(t) = t + 3$$
Mode Likelihood Vectors

- Calculate the distance between measured error $e$ and a signature $S_i$

$$\tilde{\delta}_i(t) = \min K \int_{t-\omega}^{t} |e(t) - s_i(t, \bar{K})| dt.$$

- Calculate the mode likelihood vector

$$L(t) = <l_0(t), l_1(t), \ldots, l_n(t)>$$

where each $l_i(t)$ is defined as:

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

If 2\textsuperscript{nd} greatest element of $L$ is greater than threshold $\tau$, error is \textit{unknown}, else greatest element of $L$ determines current error mode.

$$\tau = 0.70 \quad \tau = 0.80$$

$$L = <0.3, 0.75, 1.0, 0.05> \quad L = <0.3, 0.75, 1.0, 0.05>$$

\textit{error mode = unknown} \quad \textit{error mode = 2}
Error Correction and Error Mode Category

• It is application dependent if a detected error mode is *recoverable* or not

• Error mode category
  o No error
  o Known
    ✷ Recoverable
      o Correct the error using independently measured inputs
    ✷ Unrecoverable
      o If there is not enough data redundancy, we cannot correct the error
  o Unknown
    ✷ Also unrecoverable
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";
        S2(K): e = -2*t + K "B failure";
        S3(K): e = K, abs(K) > 20 "Out-of-sync";
    correct
        S1: a = b / 2;
        S2: b = a * 2;
end;

S3 is unrecoverable
Outline

- Motivation and goals
  - Air France 447 accident
  - Dynamic Data Driven Avionics Systems
- PILOTS
  Programming Language for spatio-Temporal data Streaming applications
- Error detection & correction methods
  - Error signatures
- Experiments
  - Cessna flight data (simulated errors)
  - AF447 flight data (actual data error)
- Future work
Experimental Settings

- **SpeedCheck PILOTS program**
  - Checks if $\text{airspeed}(v_a)$, $\text{ground speed}(v_g)$, and $\text{wind speed}(v_w)$ are consistent
  -Corrects $v_a$ or $v_g$ when possible

- **Input Data Streams**
  - Cessna flight data (simulated errors)
  - AF447 flight data (actual data error)

- **Define a general error signature set applicable to data from both flights**

- **Evaluate**
  - Error detection *accuracy*
  - Error detection *response time*
Derive a general error signature set for SpeedCheck

- Assuming three error scenarios
  - Pitot tube (airspeed) failure
  - GPS (ground speed) failure
  - Both (Pitot tube + GPS) failures

- Parameters
  - \( v_a \): airspeed
  - \( a \): wind to airspeed ratio (i.e., \( v_w = av_a \)) \( \in [0,1] \)
  - \( b_h, b_l \): higher/lower pitot tube clearance ratio \( \in [0,1] \)
    - 0: fully clogged (\( v_a = 0 \))
    - 1: fully clear (\( v_a = v_a \))
• Error signature set when $a = 0.1$, $b_h = 0.2$, $b_h = 0.33$

![Graph showing error signatures for SpeedCheck](image)

- $v_a = 162$ knots for the Cessna flight
- $v_a = 470$ knots for the AF447 flight

### Table: Error Signatures for SpeedCheck

<table>
<thead>
<tr>
<th>Mode</th>
<th>Function</th>
<th>Error Signature Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>$e = k$</td>
<td>$k \in [-av_a, av_a]$</td>
</tr>
<tr>
<td>Pitot tube failure</td>
<td>$e = k$</td>
<td>$k \in [(1-a-b_h)v_a, (1-a-b_l)v_a]$</td>
</tr>
<tr>
<td>GPS failure</td>
<td>$e = k$</td>
<td>$k \in [-(a+1)v_a, -</td>
</tr>
<tr>
<td>Both failures</td>
<td>$e = k$</td>
<td>$k \in [-(a+b_h)v_a, -</td>
</tr>
</tbody>
</table>
program SpeedCheck;
inputs
wind_speed, wind_angle (x,y,z) using euclidean(x,y), 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;
air_speed_out: air_speed at every 1 sec;
ground_speed_out: ground_speed at every 1 sec;
wind_speed_out: wind_speed 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 (for Cessna flight) */
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;
Performance Metrics

- **Accuracy**
  - *How accurately does program determine true error mode?*
  - Defined as average number of correct estimated mode $m'$ determinations versus true mode $m$ during a time range $T$.
  - If accuracy = 1 all mode determinations are correct, and if 0 all are incorrect.

- **Response Time**
  - *How quickly does program correctly react to mode changes?*
  - Defined as minimum/average/maximum times it takes to estimate correct mode over all true mode changes in a time range $T$. 
Experiment 1: Cessna Flight

- Data recorded on an actual flight on April 3rd, 2012
  - *airspeed, air angle*: manually collected by pilot during the flight
  - *ground speed, ground angle*: collected from online (radar) data
  - *wind speed, wind angle*: from weather forecast

Albany, NY departed at 14:04 on April 3rd, 2012

Tipton, MD arrived at 15:45 on April 3rd, 2012
Experiment 1 – Simulated Scenarios

1. Simulate pitot tube failure at flight minute 40
   - airspeed gradually drops to 50 knots in five seconds

2. Simulate a GPS failure at minute 40
   - ground speed drops to 0 knots suddenly

3. Simulate both pitot tube and GPS failures at minute 40
   - Scenarios 2 and 3 together
Experiment 1 – Error Detection Results

Error signature set

1. Pitot tube failure ($\tau = 0.8$, $\omega = 1$)

- Accuracy = 0.929, response = 4 sec

2. GPS failure ($\tau = 0.8$, $\omega = 1$)

- Accuracy = 0.935, response = 0 sec

3. Both failures ($\tau = 0.8$, $\omega = 1$)

- Accuracy = 0.942, response = 5 sec
Experiment 1 – Performance Results

- Higher the threshold $\tau$, better the accuracy & response time
  - Higher threshold is good for avoiding unknown error mode
- Lower the windows size $\omega$, better the accuracy & response time
  - Errors are simple and do not require past history data
Data extracted from the final report of Air France Flight 447

- **airspeed, air angle**: extracted from the graphs
  - Real pitot tube failure is recorded
- **ground speed, ground angle**: extracted from the graphs
- **wind speed, wind angle**: “the wind and temperature charts show that the average effective wind along the route can be estimated at approximately ten knots tail-wind.”
  - wind speed $\leftarrow$ 10 knots
  - wind angle $\leftarrow$ air angle

Experiment 2 – Demo
Experiment 2 – Error Detection Result

Error signature set

Real AF447 flight data ($\tau = 0.8, \omega = 1$)

- Pitot tube failure
- Normal
- Both failures
- GPS failure

Accuracy = 0.931, response = 2.5 sec
Experiment 2 – Performance Results

- Same tendency as the Cessna flight data
- The same error signature set works for both the Cessna and AF447 flights
Concluding Remarks

1. Error functions
   - Must make redundancy between independently sensed/measured/produced input streams explicit

2. Error signature set
   - Well-behaved: Under normal and known error conditions, must produce nearly orthogonal mode likelihood vectors
   - The error signature set defined for the Cessna and AF447 flights works pretty well (accuracy: about 93% for a Cessna flight, 96.31% for AF447)

3. Choices of threshold $\tau$ and window size $\omega$
   - Domain specific
     - for SpeedCheck, $\tau=0.8$ and $\omega=1$ give best results
   - Larger $\omega$ values lead to less responsive programs; however, for too small $\omega$, the system could enter unknown mode more frequently.
   - For well-behaved signature sets, $\tau$ has less effect on accuracy. Otherwise, for smaller $\tau$ values, unknown mode is entered more frequently, while too large $\tau$ values can produce more false positives.
(Short-term) Future Work

- **Applying error signatures**
  - Airplane weight vs. performance analysis (using data from Tunisian Airlines accident)
  - Non-linear error signatures
    - exp, log, sin, piecewise, ...
  - Other domains

- **Reducing running time for distance computation**
  - Search space increases as
    - The size of constants set $K$ increases
    - The number of error signatures increases
  - Use provably-equivalent incremental algorithms

- **Error signature discovery**
  - Mining data/learning from errors

- **External DDDAS software components**
  - Simulated input data
  - Output visualization
  - Pipelining components and feedback loops
A quantitative spatial and temporal logic as a formalism:
- To enable reasoning about data streams that associate values to specific points or intervals of space and time.
- To enable geometric reasoning capabilities, in particular, trigonometric formulae to calculate with aircraft speeds, headings, range, and endurance.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v$</td>
<td>Speed (horizontal)</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Direction</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Aircraft</td>
</tr>
<tr>
<td>$w,x$</td>
<td>Wind, crosswind</td>
</tr>
<tr>
<td>$r$</td>
<td>Runway</td>
</tr>
</tbody>
</table>

**Ground speed and crosswind as functions of airspeed, wind, and runway heading**

$$v_g = v_a + \sin(\alpha_w - \alpha_a) \times v_w$$

$$v_x = \cos(\alpha_w - \alpha_r) \times v_w$$
(Longer-term) Future Work (2/3)

- Extensions to logic programming to support *stochastic reasoning*.
  - Language extensions to standard Horn clause-based knowledge bases to incorporate probabilities.
  - Special language support for spatial and temporal data streams.
  - Incremental reasoning algorithms to dynamically re-compute logical queries efficiently as new data gets injected into the application.

<table>
<thead>
<tr>
<th>If...</th>
<th>then...</th>
</tr>
</thead>
<tbody>
<tr>
<td>New pilot report: icing en route</td>
<td>New route</td>
</tr>
<tr>
<td>New winds aloft</td>
<td>New altitude</td>
</tr>
<tr>
<td>New surface winds at destination</td>
<td>New airport</td>
</tr>
<tr>
<td>Imminent engine failure</td>
<td>Nearest airport</td>
</tr>
</tbody>
</table>

*Dynamic Data-Driven Flight Plan Adaptation Examples*
Data streaming analytics in real-time using cloud computing

- More and more data are expected to be available through the Internet
- Reason about spatial and temporal data in real-time
  - Give pilots better information to make more accurate judgments during crucial emergency moments
Publications

- **DDDAS 2012 ICCS**
  - PILOTS Programming Language Design; Error Signatures
- **ACM GeoSpatial 2012**
  - PILOTS Programming Language Implementation; Twice, Accuracy and Response Time performance metrics
- **ACM/IEEE UCC 2012**
  - Application-level Migration for Dynamic Reconfiguration in Cloud Computing
- **MIT Press 2013**
  - Programming Distributed Computing Systems, A Foundational Approach
- **DDDAS 2013 ICCS**
  - Autonomous Error Detection and Recovery; Mode likelihood vectors, PILOTS SpeedCheck Cessna example
- **IEEE BDSE 2013**
  - Self-Healing Spatio-Temporal Data Streams, PILOTS SpeedCheck AF447 example
- **ACM AGERE @ SPLASH 2013**
  - Structured Reasoning for Actor Systems
- **ACM/IEEE UCC 2013**
  - Accurate Resource Prediction on Hybrid Clouds using Workload-Tailored Elastic Compute Units
Open Source Software

- Download PILOTS 0.2.3 at:
  
  http://wcl.cs.rpi.edu/pilots

  Thanks! Questions?

  Partial support from:
  Air Force Office of Scientific Research
  DDDAS Program
  Dr. Frederica Darema
  (AFOSR Grant No. FA9550-11-1-0332)
  &
  Yamada Corporation Fellowship

New textbook:

PROGRAMMING DISTRIBUTED COMPUTING SYSTEMS
A Foundational Approach

CARLOS A. VARELA

MIT Press, June 2013