UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Monitoring the Quality of Information (QoI) for low-cost sensor networks

Albert Cerezo Llavero

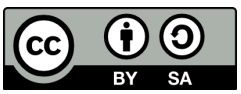
Supervisors: José M. Barceló Ordinas
Jorge García Vidal

Departament d'Arquitectura de Computadors
Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Master Thesis submitted for the degree of
Master in Research and Innovation in Informatics
Computer Networks and Distributed Systems (CNDS)

25th April 2019

Albert Cerezo Llavero
ORCID: 0000-0003-3392-7809



I would like to dedicate this thesis to my loving parents and to my loving Lara

Acknowledgements

I would like to thank Jose María Barceló Ordinas and Jorge García Vidal for letting me be part of their research group for the last three years. I also wanted to thank them for their trust in me to be part of a European research project during this period.

Abstract

Sensing devices have improved a lot during the past having low-cost, smart and portable sensors that can be placed everywhere forming low-cost sensor networks. Having low-cost sensor networks also enables to perform citizen science using sensing devices, where people from different backgrounds collaborate with researchers. The present work is part from an European citizen science project named Captor. This project is mainly focused on making people aware of current pollution problems. For that purpose, many sensing devices are placed in volunteers houses to monitor air quality. Therefore, maintaining reliable results from low-cost sensor networks is a must. If such requirements would not be met, population would not take into account the results obtained. Over time, sensors change the way they measure some phenomenas compromising the accuracy of the measurements. When that occurs, sensors need to be replaced or calibrated in order to get accurate measurements again. Then, the scope of this work is to investigate how to achieve good data quality and how to maintain it. First of all, different scoring statistical methods are applied to synthetic data. Some scoring errors patterns are extracted from comparing comparing co-located. The results from this experiments are then applied to a real scenario to validate the statements set before. From these empirical experimentation, some policy rules are defined to tag sensors as unreliable. Finally, a real time event driven solution is proposed which implements the *Quality of Information* rules defined previously.

Table of contents

List of figures

List of tables

Nomenclature

Greek Symbols

η	Continuous signal measurement
η_e	Estimated continuous signal measurement
η_r	Real continuous signal measurement
$\Phi^{x,y}$	Set of spatially and temporally close pairs of measurements
β	Coefficient
ε	Error
σ^2	Variance
μ	Mean
σ	Standard deviation
ρ	Pearson correlation

Superscripts

$0-4$	Ozone sensor number
t	Temperature sensor
rh	Relative humidity sensor

Subscripts

n	Sensor number
m	Measurement number

i Window number

Other Symbols

y_k Measurement vector y_k

$\hat{y}_k$ Estimated measurement vector $\hat{y}_k$

U Nodes set

u_n Node n

N Total number of nodes

S_n Set of sensors from node u_n

x_{mk} Feature

s_n Sensor from node u_n

T Time domain

L Location domain

D Measurement domain

α_s Calibration constant for a given sensor

β_s Calibration coefficient for a given sensor

v_s Measurement from a given sensor

ε_s Calibration error from a given sensor

Δt Maximum temporal difference

Δl Maximum spatial difference

v_m Measurement of a sensor in measure m

a Mean of the signal

b Amplitude of the signal

f Frequency

D_{KL} Kullback-Leibler divergence

w_i Window i

w_s Window size

w_h Window hop

I Total number of windows

Acronyms / Abbreviations

ANN Artificial Neural Network

LSTM Long Short-Term Memory

MLR Multiple Linear Regression

NRT Near Real Time

QoI Quality of Information

WSNs Wireless Sensor Networks

RMSE Root Mean Square Error

VoI Value of Information

Chapter 1

Introduction

1.1 Context and motivation

Sensors get information from the medium in order to represent real world phenomena or events. Sensors have been used for many different applications: habitat monitoring, environment observation, healthcare and other comercial applications [1]. Data coming from these sensors is normally used to take decisions, meaning that having results describing those phenomenas or events accurately is critical.

The data from a sensor can not be taken as an absolute truth, it has an inherit error. In order to have more precise measurements, sensors are deployed forming sensor networks, normally Wireless Sensor Networks (*WSNs*). There have been many approaches regarding deployment and calibration of such networks. However, maintenance has been left as a second order problem.

Maintenance of wireless sensor networks could be seen as a simple task from the network or failure point of view but it is not. Sensors characteristics can change over time, in other words, the way a sensor is measuring a phenomena can change. Therefore, some parameters should be changed in order to perform more reliable measurements. It could also be the case that a change made a sensor malfunction and it has to be replaced. Such changes on the behaviour or malfunctions are not easy to notice, neither a simple software nor human observation are capable to detect them.

In both cases, it can be denoted as a data fault. A data fault can be understood as some data which is not consistent with the phenomena being measured. Such faults erroneously interfere in the ability of scientists to make meaningful conclusions. Sometimes, having data failures from sensor networks can lead to serious problems, for example, inaccurate data from methane sensors in mines can lead to harm the miners' health.

In order to avoid and predict such data faults it is important to measure the Quality of Information (*QoI*). Quality of Information could be understood as the perceived information quality by the user in terms of accuracy, completeness, reliability and certainty. There are other aspects which could be taken in this definition, but for the purpose of this thesis such terms have been extracted from Sachidananda et al. [2].

A more concrete definition of *QoI* can be done by extending the previous terms. Quality of Information is the perceived information quality having into account the correctness of the data respect to the real world, the quantity of data, freedom from changes or variations and the confidence of the delivered data. Therefore, it raises the need to use different statistics applied to sensor networks to quantify the overall *QoI* in order to improve and maintain them in the desired *QoI*.

QoI could be measured in different sensor networks but nevertheless low-cost sensor network is used due for the present work. Low-cost sensor network is a special scenario where the sensors are characterised by its deployment forming a network, having some spacial and time advantages, and by the fact that being low-cost sensors it will be easier to have many of them. However, such sensors sometimes are related to less-quality sensors in comparison to those that are more expensive. Therefore, to have good *QoI*, spacial, time, replication and redundancy advantages should be taken.

1.2 Objectives

The present work is focused on researching good *QoI* indicators for a low-cost sensor network deployed in Barcelona in the contest of a European Horizon 2020 project. First of all, hypothesis will be tested with some synthetic data, evaluating different possible behaviours. In addition, some injected faults will be added in the data to tune and validate the indicators. Later, the data from the low-cost sensor network will be compared with some other high-cost sensors in order to validate the model.

Once all hypothesis are tested, an event based real time monitoring tool is developed. This tool will allow to show historical sensor information combined with real time notifications based on *QoI* indicators. For that purpose, novel open source, distributed and real time frameworks are used. It will combine an event queue based framework with a real time processor. Additionally, a near real time (*NRT*) dashboard is used to show the results.

Combining the indicators with an event based processor is not an easy task. Due to the fact that it will be compared in real time, the model will need to summarize using sliding windows. Sliding windows could be defined as small continuous time portions. As a result, the windowing protocol has to be optimum sin terms of size for each indicator. An optimum

one would be a window whose size does not delay failure detection and it also does not notice false faults.

1.3 Thesis structure

The document is structured in seven sections. The present one contains some introductory context and motivation to the project, setting the main objectives. The second one, includes related work done by other researchers in terms of assessing *QoI* and some other fault analysis. The third section describes the European Captor Project.

The fourth section includes statement of the theoretical scenario, the scope of the project and the procedure which will be used. The fifth one covers common faults and ways to test them. The sixth section is the more extensive one, as it contains all the experiments performed to test such faults, both with synthetic data and later with real data.

Next, the seventh section describes the resulting software tool from the experimental analysis and the recommended deployment architecture for larger scenarios. Finally, some conclusions are extracted from the work. All the work which is not from the author is referenced at the end of the document. Data generation and tests are scripted using Python. These scripts are available in an attached file, *code.zip*. Additionally, software and hardware used for the present work are described in *Appendix A*.

Chapter 2

Prior and related work

In this section, works related with sensors, calibration, data faults, *QoI* assessment and monitoring are covered. First of all, works related with sensors calibration are presented differentiated by calibration methods. In the survey from Barcelo et al. [3] most calibration methods are presented and classified using different criteria such as if they are on-line or off-line methods or if they have a ground truth to compare with or not among others.

One approach of performing sensor calibration is to use linear functions. Whitehouse and Culler [4] use Multiple Linear Regression (*MLR*) for calibrating a localization system by assigning multiple to each positioning sensor a parameter to increase the system's performance. Balzano and Nowak [5] calibrate temperature and humidity sensors without having an accurate ground truth, which is known as blind calibration. Hasenfratz et al. [6] and Barceló et al. [7] use *MLR* and *MLR* fusion to fit pollution sensors having reference sensors which are used as ground truth.

There are other works that use non-linear methods to calibrate sensors. Moses and Patterson [8] calibrate sensor networks using a Bayesian approach where maximum posteriori estimation is used. Non-linear methods also include supervised learning algorithms. Spinelle et al. [9] [10] use an Artificial Neural Network (*ANN*) to calibrate pollution sensors. Fujino and Honda [11] use another non-linear method for temperature method based on Gaussian processes regression.

Additionally, calibration can be classified by other characteristics as being a consensus or distributed calibration. Bolognani et al. [12] perform a consensus calibration algorithm for a localization and target tracking application. The mentioned algorithm does not need to know the overall system topology. Having only partial view consensus is able to achieve reasonable results.

Once different calibrations methods have been covered, it is important to assess different *QoI* of *WSNs* works. They go from more theoretical scenarios as setting general terminology of *QoI* or common data faults to more concise solutions for concrete scenarios.

Sachidananda et al. [2] define *QoI* of *WSNs* using different attributes, giving a common starting framework for later works, as the present one. It also describes four functional blocks: data collection, processing, transport and sinks or applications. Therefore, when dealing with *QoI* of *WSNs* many phases can be assessed. Bisdikian et al. [13] not only introduce a *QoI* definition with many attributes, but also include another concept, Value of Information (*VoI*). *VoI* is defined as "an assessment of the utility of an information product when used in a specific usage context".

Ni et al. [14] cover common data faults that occur in deployed sensor networks. It classifies the faulty behaviours by the type of features: environment, system and data. Such classification gives a set of possible common features to asses data faults modelling from a general point of view.

Su et al. [15] describe a *QoI* based data selection in wireless sensor networks. The proposed method is based on consensus, setting the sensors' contribution. The solution is based on a optimization problem maximizing the reliability of sensory data while eliminating their redundancies under the constraint of network resources.

L. Ma, X. Gu and B. Wang [16] propose a correction outlier algorithm based on sliding window prediction. The method is based on supervised learning, meaning that historical data is used to train a predictive model. Such model gives a prediction of the next possible measurement based on the last window, if the measurement is greater than an threshold, the measurement is labeled as an outlier, which is corrected.

Guo et al. [17] propose a windowed based framework focusing on data accuracy in a multi-hop wireless sensor network. The data from the sensor nodes is sent to a central sink where data is aggregated and evaluated using sliding windows. Always the data is quantified without a ground truth thanks to information fusion theory.

Joslyn and Lipor [18] deal with water quality sensors which seem to be regularly corrupted due to sensor faults. Therefore, based on manually tagged datasets, they performed supervised learning. The machine learning methods used are support vector machine and gradient boosting for regression.

Chapter 3

CAPTOR

CAPTOR is an European H2020 research project whose main aim is to collect and measure air quality [7] [19]. For that purpose low-cost sensor nodes are deployed in different test beds. The present thesis is developed within CAPTOR project.

One important objective of this project is to raise people's awareness which is assessed by being a citizen science project. Many different volunteers and researchers collaborate to make people aware of current pollution problems. This fact makes people change from being informed to being part of it, having a greater impact.

During the project 170 sensoring devices have been deployed in Spain, Austria and Italy. Although there have been two main different CAPTOR devices, both have measured Ozone(O_3), nitrogen dioxide (NO_2), temperature and relative humidity. One of them, Captors, has been built using Arduino, printed boards, electro-chemical O_3 and NO_2 sensors and a temperature and relative humidity sensor. The other ones, Raptors, where built using Raspberry boards. Another difference between them is that Captors are directly plugged to electrical power supply and to a Wi-Fi network whereas Raptors use a battery and are connected using a mobile network.

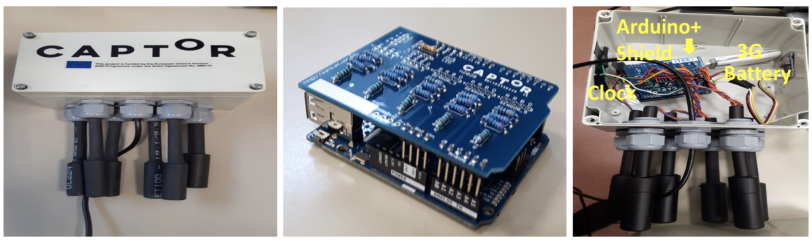


Fig. 3.1 Captor sensing nodes

Captor nodes have been build by UPC Barcelona, with a Do It Yourself (*DIY*) philosophy, using hardware which is affordable and easy to build from a written guide. Additionally,

a video was recorded to allow volunteers to build their own nodes and to add them to the network. Figure 3.1 shows some pictures taken to the nodes.

Captor nodes were calibrated using a reference station, which was designed as ground truth. The reference station is placed in Barcelona and maintained by the Spanish National Research Council (*CSIC*) [20]. Then, the nodes were collocated next to the reference station during 4 weeks. Then, in order to calibrate them, in an off-line manner, hourly measurements of each sensor and the reference station were compared. Coefficients for the different sensors are obtained from the calibration and set into the nodes in order to enable them to send calibrated value.

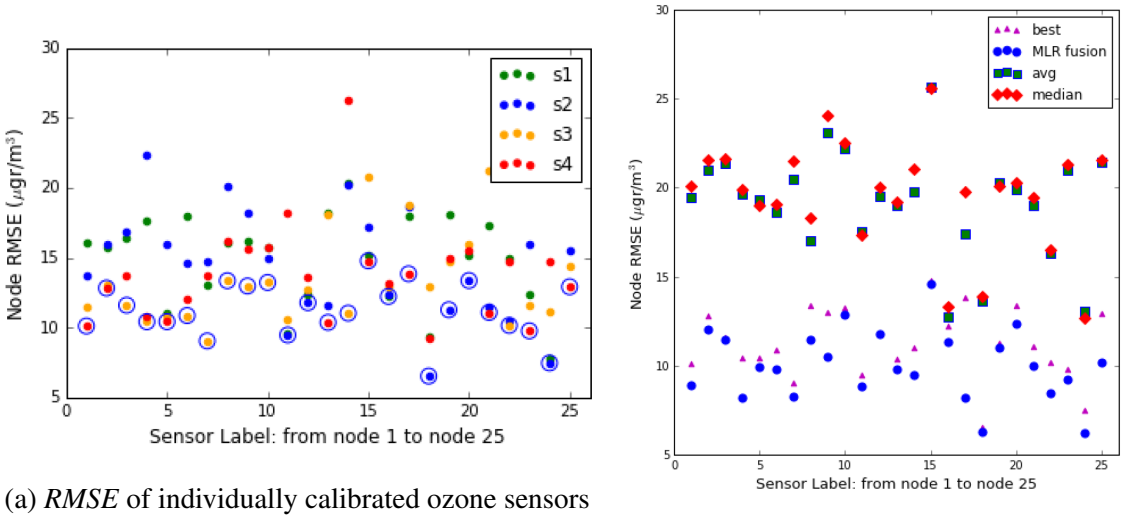
Coefficients for calibrated values were calculated in two different ways. One was to perform a regression for each sensor individually plus the temperature and humidity and to take the one with less *RMSE*. The second one was to combine all sensors, humidity and temperature. The *RMSE* formula used to compare estimated measurement with the reference one is shown below.

$$RMSE = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(\hat{y}_k - y_k)^2} \tag{3.1}$$

Considering K as the number of measurements, where $K \subseteq \mathbb{N}$.
Having y_k as the measurement k from the reference station and $\hat{y}_k$ as the estimated measurement k by a given node or sensor.

Once the different coefficients were calculated, the approaches were compared. The figure 3.2a shows best calibrated sensor selection based on the lowest *RMSE*. Figure 3.2b shows node calibration using *MLR* fusion, best sensor (presented before), average between sensors and mean between sensors. There is some improvement by adding more sensors to the calibration method using fusion than just taking the best individually calibrated sensor. However, many of the nodes does not reduce substantially the *RMSE*. From the figure it can also be extracted that using other fusion methods such as average or mean most of the times increases the *RMSE* between 70% and 90%. The nodes were also reviewed by Ripoll et al. [21].

Once all nodes were calibrated, they were distributed among volunteers' houses around Catalonia during summer period (June-September). The campaign period is during summer due to Ozone increases when the temperature is high. Therefore, measurements which could be harmful and alerting are normally only during the mentioned period.



(a) *RMSE* of individually calibrated ozone sensors

(b) *RMSE* of all sensors from a node

Fig. 3.2 *RMSE* of 25 calibrated ozone nodes using different methods

Additionally, within the CAPTOR project a web app and a mobile app were developed in order to show to volunteers and to the rest of the population, measurements taken by Captors and Raptors. It was developed within a final project by the same author of the present work [22]. The application is available on a website [23] and in Google Play [24]. Figure 3.3 shows captorAIR map, where all the different nodes are positioned on a map and colored according to air quality measurements. By clicking on a node, further information is given on the right side. By clicking on a concrete sensor link from a node, historical information with different recommended and legal thresholds as shown in the figure 3.4.

It is important not to create an alarm neither to have inaccurate data or the results will not be considered by population. Figure 3.5 which shows a Captor which is misbehaving. It is taking measurements over $600\mu g/m^3$! It is fairly impossible to have that concentrations. Having them would imply that no one would be able to live in that environment.

After the campaign, the nodes were returned to the reference station in order to check if the way sensors measure ozone is still the same or the coefficients are no longer properly calibrated. However, during the campaign period, checking sensors' coefficients was a manual task. This fact leads to undetected misbehaving sensors or detected misbehaviours with days or even weeks later. Therefore, there is a high motivation to detect them automatically when the ground truth or reference station is no longer comparable with the low-cost sensors.

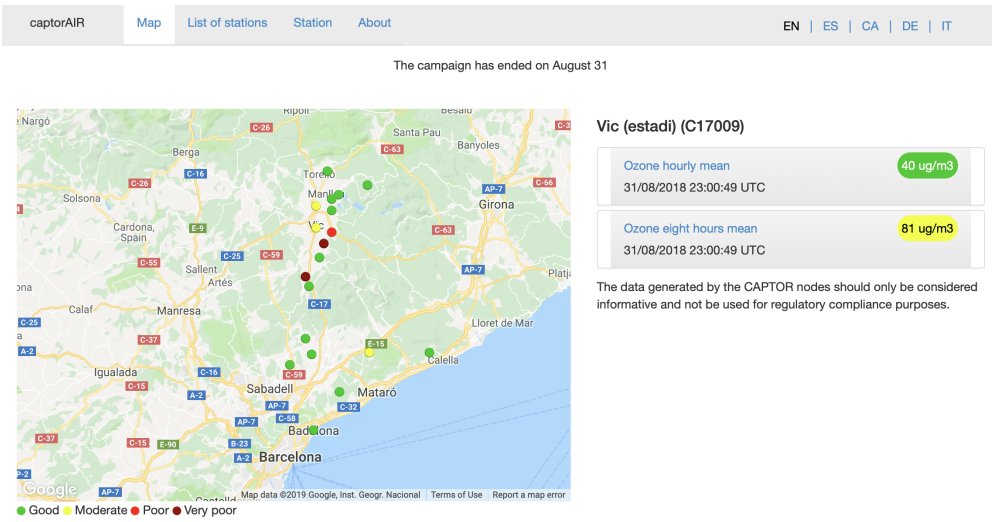


Fig. 3.3 CaptorAIR map [23]

Fig. 3.4 CaptorAIR historical data [23]

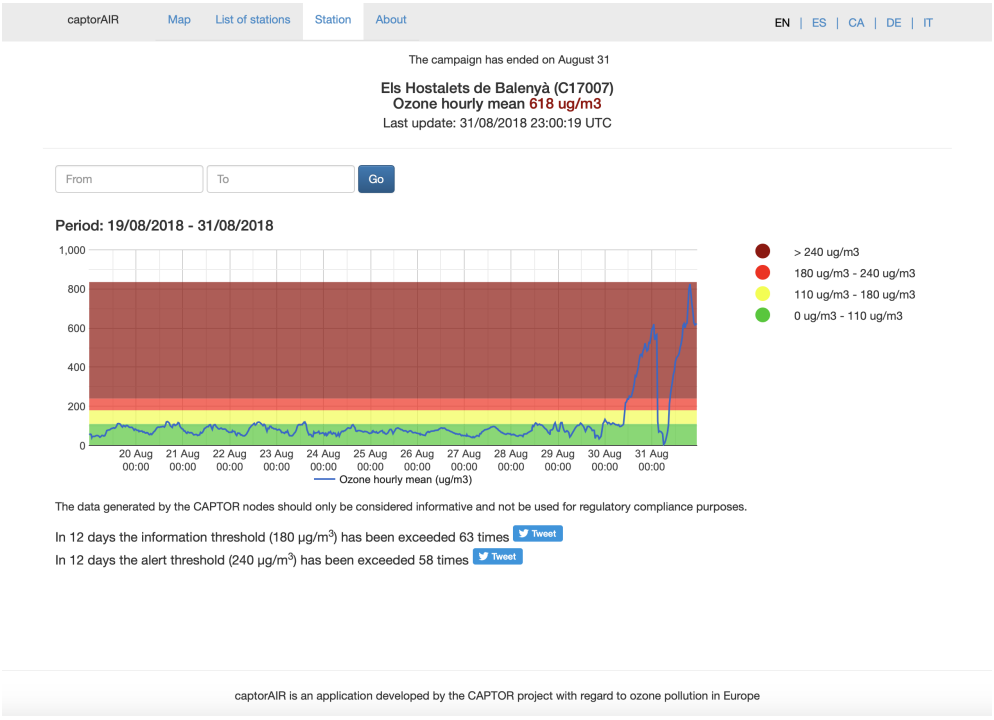


Fig. 3.5 CaptorAIR historical data error [23]

Chapter 4

Scope and statements

4.1 Scope

The scope of the present work is to assess the *QoI* of a *WSN*. This *WSN* is monitoring environment phenomenas such as temperature, humidity and ozone (O_3). Although the tests will focus the current scenario, the method to quantify such quality could be used in many other scenarios.

The nodes used are built using low-cost hardware, not only the sensors but also the boards and rest of hardware. Therefore, computing capabilities of these nodes are very limited, meaning that it will not be considered that any *QoI* assessment will be done in the nodes. Neither monitoring other hardware nor the connection is part of the present work, only data sent by sensors.

The nodes are connected to internet via Wi-Fi or 3G. There are not direct connection between them, as there is in other *WSNs*, for example, multi-hop networks. As a result, all data will be sent to a central point, enabling the system to work with the whole information and not only with part of it, as it is usual in multi-hop networks.

Another point to consider is that there is no ground truth, there is no other data that shows concrete and accurate information about the measured phenomena. Therefore, information fusion theory (*MLR fusion* is used to generate better output data from the system. However, compared data will not be the processed one, as due to the fact that processed data has the same or less information than before being processed, which is demonstrated by information therory.

Finally, according to the scenario presented in *section 4.2.1*, nodes have multiple sensors, having the opportunity to compare them according to the rendezvous between sensors described in *section 4.2.3*. The nodes presented have been deployed over different locations in Catalonia, Spain. This fact enables to extend the models used with other spacial modelling.

Those errors that will be monitorized are described in *section 5.1*. The presented errors will be monitorized by a set of statistic indexes such as correlation and also using information theory with mutual information and entropy in *section 5.2*.

4.2 Statements and assumptions

4.2.1 Scenario

The current scenario is defined by a set of nodes U, being u_n the node n having $u \in U$ and being N the total number of nodes. Each node has a set of sensors $S_n \in u_n$.

There are three type of sensors: temperature, humidity and ozone, the last ones are the ones that are replicated, having two different node types, those with three ozone sensors or those with five. Then, two sets are possible describing the sensors of a node:

$$\{s_n^1, s_n^2, ..., s_n^k\} \in S_n \quad k = 1, \ \ldots, \ 4, \ t, \ rh \tag{4.1}$$

Each superscript denotes the type of sensors, when it is a number it sets the number of ozone sensor and t and rh are the temperature and the relative humidity sensores respectively.

4.2.2 Calibration process

According to Barceló et al. [7] the Captor nodes are calibrated using a *MLR* in order to have accurate measurements. The formula is described as follows:

$$y_m \sim f(\beta, x_m) = \beta_0 + \sum_{k=1}^{K} \beta_k x_{mk} + \varepsilon_m \qquad 0 \leq m < M \tag{4.2}$$

Where y is a vector with the measurements from the ground truth (reference station). Being ε_m a random error term, Gaussian distributed with zero mean and variance σ^2. Coefficients are denoted by β , where β_0 is the constant and β_k the coefficient k which multiplies the values for a given feature k. Then, x_{mk} is the measurement for the sensor k on measurement m. Having a total number of measurements M.

In order to obtain the coefficients, the problem is solved by a least-squares minimization problem. For that purpose, four weeks historic data is splitted into a training and test dataset

with a ratio of 65% and 35% respectively. Once, the nodes are validated with an error below 15% are ready to be deployed for the summer campaigns.

4.2.3 Rendezvous between sensors

Accordint to O. Saukh, D. Hasenfratz and L. Thiele [25] the phenomena being measured can be described as a continuos signal $\eta : T \times L \rightarrow D$ with time domain $T \subseteq \mathbb{R}^+$, location domain $L \subseteq \mathbb{N} \times \mathbb{N}$ having latitude and longitude and a measurement domain $D \subseteq \mathbb{R}$.

The phenomena measurement follow the next linear formula relating sensor measurement:

$$\eta = \alpha_s + \beta_s \cdot v_s + \varepsilon_s \tag{4.3}$$

They define α_s, β_s and ε_s as the calibration coefficients and the calibration error. These variables are trained using historical data and they are used to approximate the measurement of a sensor, v_s, with the real phenomena being measured. Therefore, if the measurement characteristics of a sensor change over time, the estimated η_e would not be accurate with the real η_r.

For a given time $t \in T$, a given location $l \in L$ and a given sensor $s_n^1 \in u_n \in U$, the estimated η_e would be as follows:

$$\eta_e(t,l) = \alpha_{s_n^1} + \beta_{s_n^1} \cdot v_{s_n^1}(t,l) + \varepsilon_{s_n^1} \tag{4.4}$$

According to O. Saukh et al. [26], it is important to define which measurements can be directly compared due to the fact that are supposed to be measuring the same phenomena or not. Those which can be directly compared are also called collocated. Then, they define a set of spatially and temporally close pairs of measurements, $\Phi^{x,y}$ between sensors y and x:

$$\Phi^{x,y} = \{ \ (v_x(t_i, l_i), \ v_y(t_j, l_j)) \mid (|t_i - t_j| \ \leq \ \Delta t) \ \wedge \ (|l_i - l_j| \ \leq \ \Delta l) \ \} \tag{4.5}$$

Δt and Δl denote the maximum temporal and spacial difference between a given sensors in order to say wether they are measuring the same phenomena event or not. For ozone sensors only those placed in the same box are considered to measure the same phenomena event for a given measurement with a maximum difference of time of ten minutes, as the

measurements are averaged every half hour. However, according to O. Saukh et al. [26], they claim that averaged ozone measurements are highly correlated. Therefore, the present work will also take into account those which are placed with a maximum difference of 1km. The next figure extracted from the cited paper show such behaviour, using the Pearson correlation.

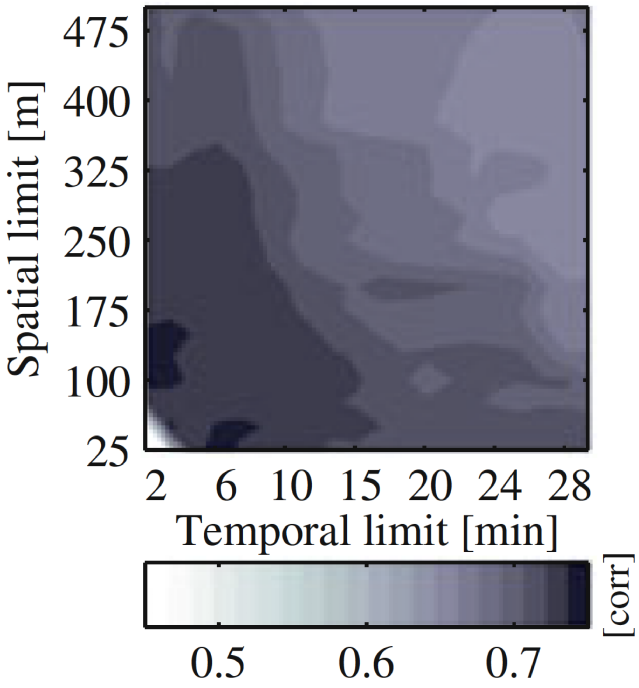


Fig. 4.1 Rendezvous between ozone sensors extracted from Saukh et al. [26]

Chapter 5

Types of faults and tests

5.1 Fault types

This section will cover a set of detected errors for Ozone sensors. Ozone measurements normally follow sinusoidal function, as it increases when temperature does. During day increases ozone concentration and during night decreases. Figure 5.1 shows real data from a reference station during august 2018 which is placed in Palau Reial, Barcelona. In order to show such errors, some data has been generated following the next sine wave formula:

$$v_m = a + b \cdot sen(2 \cdot \pi \cdot m \cdot f) + \varepsilon \qquad 0 \leq m < M \tag{5.1}$$

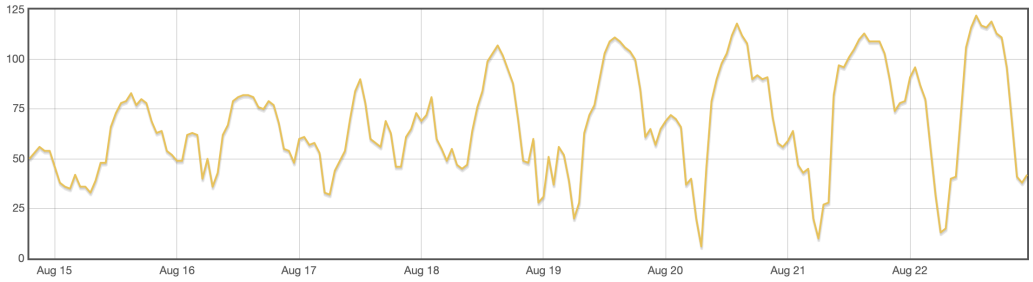


Fig. 5.1 Palau Reial Ozone measurement during August 2018

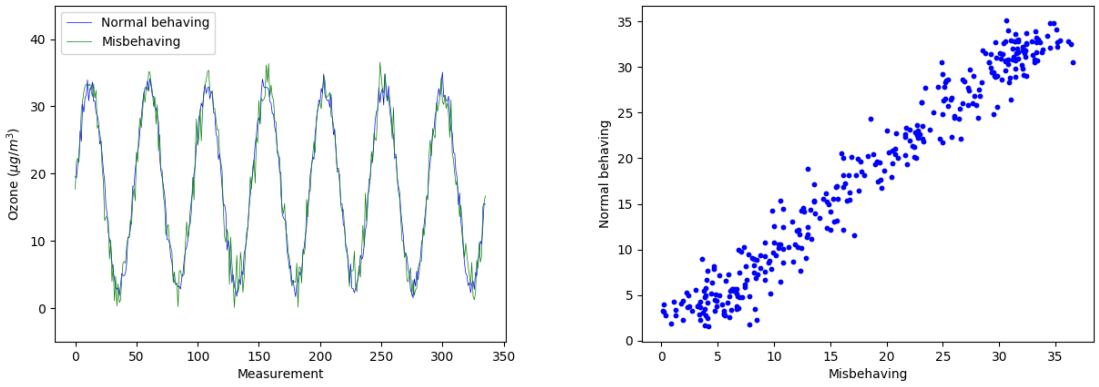
For the synthetical generated experiments, it is used a total time of 7 days with 24 hours and a sampling rate of 2 samples per hour, a total of 336 samples. Therefore m goes from $0 < m \leq M$ where M is 336. Frequency is denoted as f, being $f = \frac{1}{48}$ as every day 48 measurements are taken. Having v_m as the measurement m for a given sensor. Being a the

mean of the signal and b the amplitude. It has a random error ε which is described by a normal of mean 0 and variance σ^2.

All faults will be exemplified using two generated signals: one showing the normal behaviour of a sensor (blue) and the other one showing the misbehaving one (green). All of the faults will include a table showing the parameters taken and a figure including time series plot of both signals (the blue would be the normal behaving and the green the misbehaving) and a scatter plot where each v_t of each signal is used as an axis. The next example shows two signals behaving similarly, from now on these two signals will be called *basic example*.

Table 5.1 Table with two similar synthetic signals

	Normal behaving	Misbehaving
a	18	18
b	15	15
σ^2	1	2



(a) Time line plot

(b) Scatter plot

Fig. 5.2 Basic plots of two similar synthetic signals

Having both plots helps to understand better the behaviour of each signal. In this case, both behave similarly. From the time line it can be seen that both signals are almost the same having slightly differences on each point. This is due to the fact that the normal distribution used to generate noise is a random distribution, giving some randomness to the experiment. The second plot, can help to see if both are correlated or not. If they were not correlated, it would look like disperse points without following a line. Thanks to the fact that one is set as an accurate or normal behaving signal, the one that is misbehaving can be easily noticed.

The next subsections keep the same normal behaving signal. However, each subsection changes the misbehaving signal according to different errors which has been noticed over time. All of them will follow the same structure, setting the attributes of the function, plotting the signals and later detailed information.

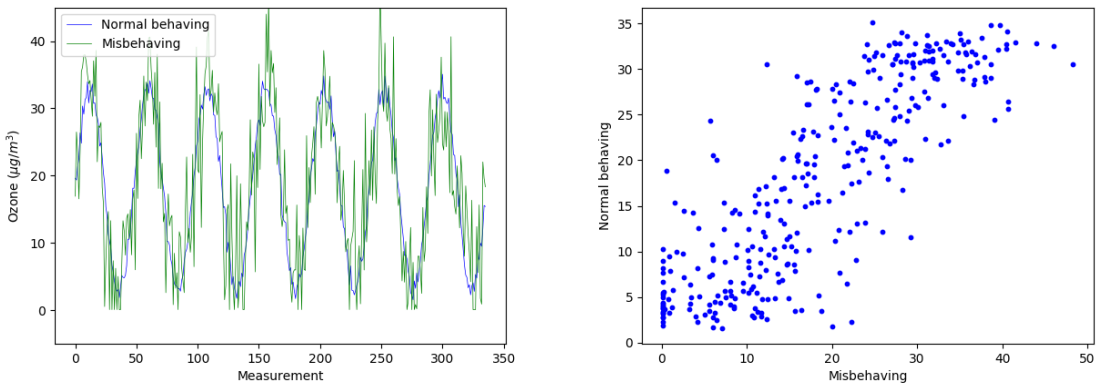
5.1.1 Big error

This fault is a common one, which could be understood as a sensor that is performing bad and giving different values for a certain expected value. For example, from the scatter plot, if we look into the y axis which reflect a good approximation of the expected value, values that are supposed to reflect 30 units are between [20, 40] in the misbehaving generated signal (x axis).

Sensors having this fault are considered to malfunction due factory process or a bad manipulation of the sensor. Such fault can not be solved with calibration methods, as in the end some information is missing or wrong and it can not be converted into an accurate output. Therefore, normally they are labeled as sensors to be replaced.

Table 5.2 Table with two synthetic signals, one with a big error

	Normal behaving	Misbehaving
a	18	18
b	15	15
σ^2	1	7



(a) Time line plot

(b) Scatter plot

Fig. 5.3 Plots of two similar synthetic signals, one with a big error

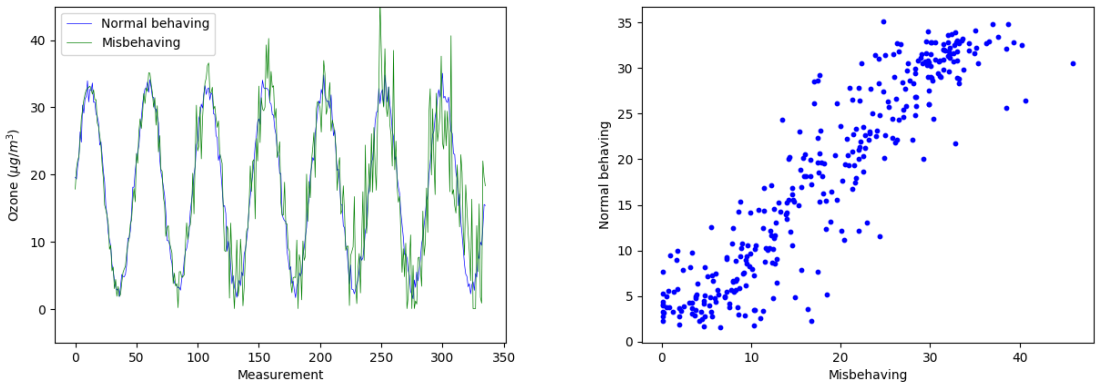
5.1.2 Increasing error

This fault is characterised by a gradual degradation where a sensor over time starts having more and more error in the measurements, which is translated into worse quality data over time. Although, looking into *Figure 5.4b* could be similar to *Figure 5.3b*, they are not the same fault. It is better to compare them using a time line plot, in which plot for a given period the sensor is performing considerably good until more or less from day four til the end (from measure 192 to 336) in which the error becomes unacceptable.

When assessing faults with different statistic tests, the difference between a measurement with a big error and one which degrades over time will be easier to notice. This is due to the fact that working with time windows makes easier to see that one is maintained error and the other ones is increasing.

Table 5.3 Table with two synthetic signals, one with an increasing error

	Normal behaving	Misbehaving
a	18	18
b	15	15
σ^2	1	day $\cdot$ 1



(a) Time line plot

(b) Scatter plot

Fig. 5.4 Plots of two similar synthetic signals, one with an increasing error

5.1.3 High/low values

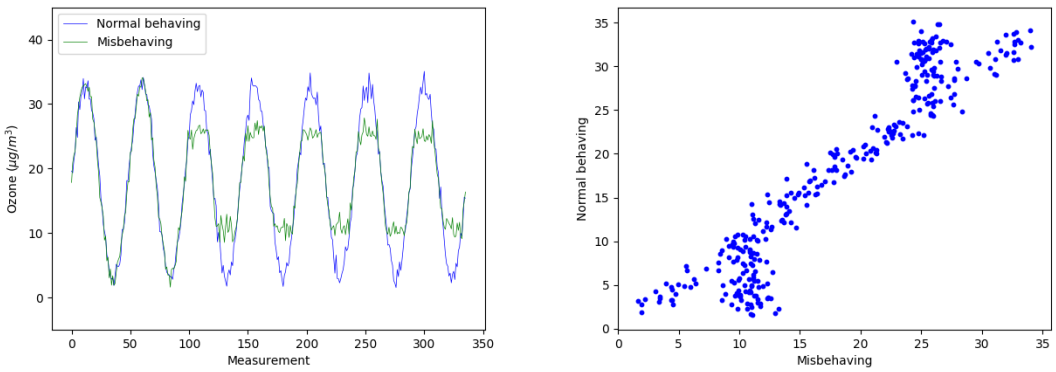
The assessed error has to be with the fact that some sensors are not capable of measure some high/low values. In *Figure 5.5a*, the misbehaving signal has a maximum and a minimum which can be easily identified. In addition, this behaviour is also identificable in *Figure 5.5b* where the dots are more sparse in both extremes of the plot (high and low values).

The second signal remains as a normal behaving one until on day three, it changes the behaviour being incapable of taking high and low values. There is an extra parameter c which is used to generate the misbehaving signal. A c value of 0.5 means that signal measured will be in range $[a-b\cdot 0.5, a+b\cdot 0.5]$ plus error. The formula used to exemplify the error is below:

$$\begin{cases} v_m = min[(a+b*(1-c)),(a+b\cdot sen(2\cdot\pi\cdot m\cdot f)] + \varepsilon \\ v_m = max[(a-b*(1-c)),(a+b\cdot sen(2\cdot\pi\cdot m\cdot f)] + \varepsilon \end{cases} \tag{5.2}$$

Table 5.4 Table with two synthetic signals, one with bad performance for high/low values

	Normal behaving	Misbehaving (day 1 to 2)	Misbehaving (day 3 to 7)
a	18	18	18
b	15	15	15
σ^2	1	1	1
c	0	0	0.5



(a) Time line plot

(b) Scatter plot

Fig. 5.5 Plots of two similar synthetic signals, one with bad performance for high/low values

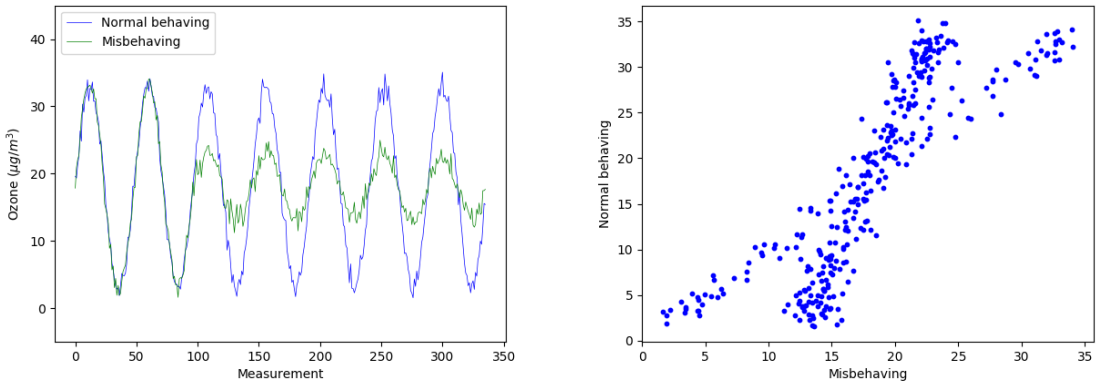
5.1.4 Amplitude

It is important to differentiate high/low values error from change of amplitude error. The main difference between them is that for the high/low one until a certain threshold the values are right whereas amplitude change requires a complete change on calibration coefficients. Therefore, noticing scale changes is a must.

As there are two clear lines that cross the points from the *Figure 5.6b*, calibration coefficients will need to be changed in order to recalibrate the node. This is due to the fact that the way a sensor is measuring has changed. This test shows a decrease on amplitude, if it had been an increase on the amplitude, the multiplier coefficient could be sending very high values which do not represent the real phenomena.

Table 5.5 Table with two synthetic signals, one changed its amplitude from day three

	Normal behaving	Misbehaving (day 1 to 2)	Misbehaving (day 3 to 7)
a	18	18	18
b	15	15	5
σ^2	1	1	1



(a) Time line plot

(b) Scatter plot

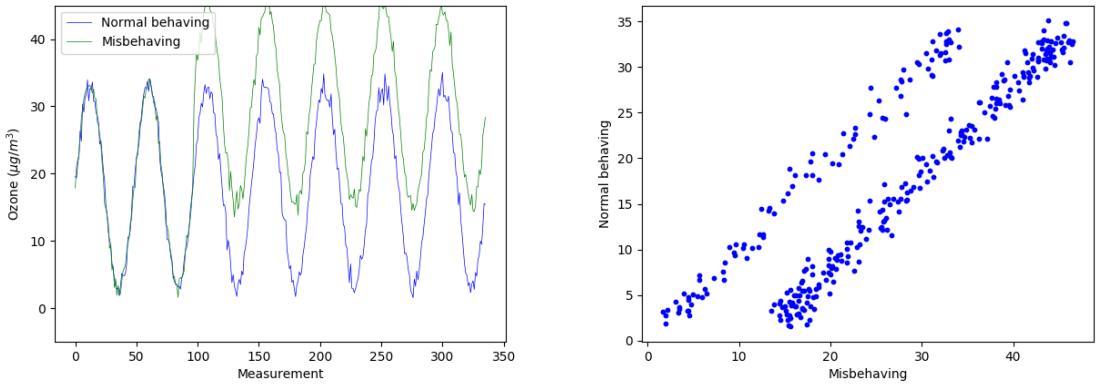
Fig. 5.6 Plots of two similar synthetic signals, one changed its amplitude from day three

5.1.5 Mean

Not only the amplitude changes but also the mean. The error has been generated by increasing the offset from the sine function from day three. It is clearly identificable in the *Figure 5.7b* due to the fact that there are drawn two parallel lines in the scatter plot. If they were not that parallel, we would be talking of mix of errors between the mean and amplitude.

Table 5.6 Plots of two similar synthetic signals, one changed its mean from day three

	Normal behaving	Misbehaving (day 1 to 2)	Misbehaving (day 3 to 7)
a	18	18	30
b	15	15	15
σ^2	1	1	1



(a) Time line plot

(b) Scatter plot

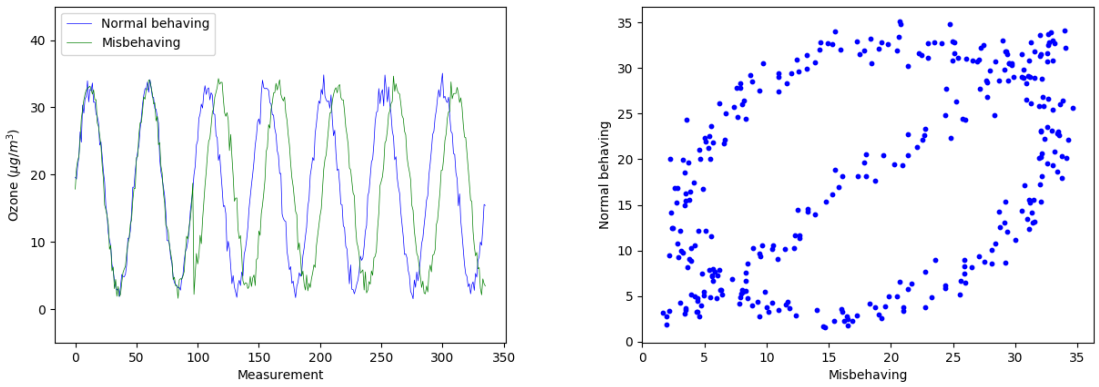
Fig. 5.7 Plots of two similar synthetic signals, one changed its mean from day three

5.1.6 Time gap

This error is normally caused by a clock error. It is not a common error neither easy to notice due to the fact that all sensors share the same clock. However, it is the easiest to solve as it does not need to be recalibrated, it only needs to set correctly the clock time. Therefore, it is recommended to check it by the Captors software by asking to some clock service on-line. It is characterized in the scatter plot (*figure 5.8b*) by drawing a circular or ellipsoidal shape when comparing with a collocated sensor using a clock with the correct time.

Table 5.7 Plots of two similar synthetic signals, one changed its mean from day three

	Normal behaving	Misbehaving (day 1 to 2)	Misbehaving (day 3 to 7)
a	18	18	18
b	15	15	15
σ^2	1	1	1
g	0	0	-10



(a) Time line plot

(b) Scatter plot

Fig. 5.8 Plots of two similar synthetic signals, one changed its clock from day three

5.2 Comparison tests

Once different errors particularities are described, it is import to set the tests that could notice their behaviours. For that purpose different statistical methods are used. First of all, a very simple way to compare them is using mean and variance. Then sensors are correlated using Pearson correlation. Finally, information theory is used as scoring method to compare both signals.

5.2.1 Basic statistics

The first test consists on calculating separately mean and variance for each sensor. Mean (*equation 5.3*) is calculated by adding each individual measurement x_m and dividing by the total number of measures M. Variance (*equation 5.3*) is calculated by adding the difference between the measurement x_m and the mean μ_x to the square, all this summation is divided by the total number of measurements M.

As the scale of a measurement can vary drastically between sensors, the best way to compare them is to perform ratios. Therefore, a mean ratio is expressed as $\mu_x : \mu_y$ bing the result mean of x divided by the mean of y. It also applies for the variance ratio, the variance from one sensor divided by the variance from the other one.

$$\mu_x = \frac{\sum_{t=0}^{M} x_m}{M} \qquad 0 \leq m < M \tag{5.3}$$

$$\sigma_x^2 = \frac{\sum_{t=0}^{M} (x_m - \mu_x)^2}{M} \qquad 0 \leq m < M \tag{5.4}$$

$$\mu_x : \mu_y = \frac{\mu_x}{\mu_y} \qquad \sigma_x^2 : \sigma_y^2 = \frac{\sigma_x^2}{\sigma_y^2} \tag{5.5}$$

5.2.2 Pearson correlation coefficient

Pearson correlation coefficient measures the linear correlation between two variables X and Y. It takes values from -1 to 1. Where -1 means that they are totally negative linear correlated and where 1 denotes positive total linear correlation. Then, values close to 0 would mean that there is no correlation between them. *Figure 5.9*) shows different examples with its ρ.

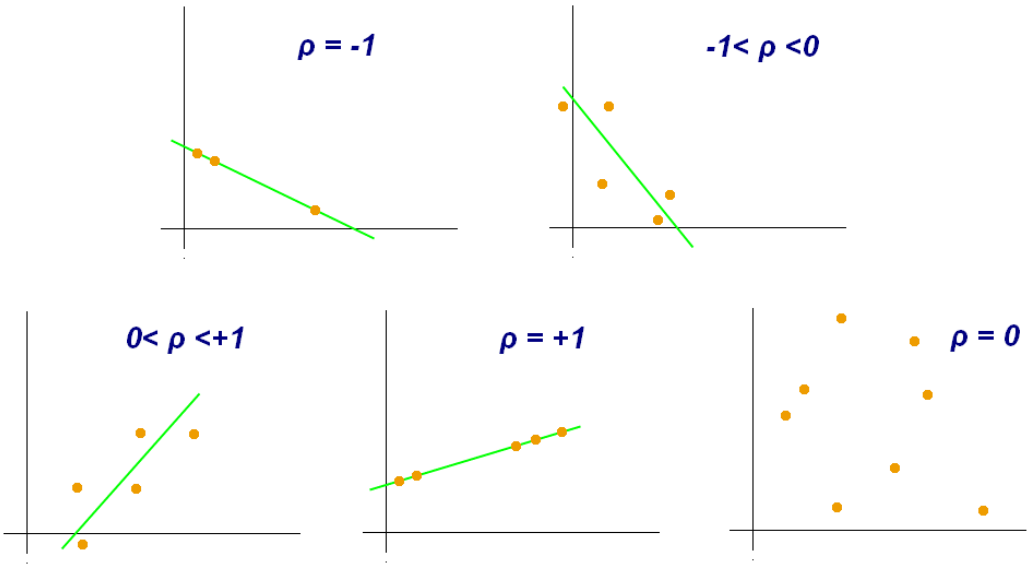


Fig. 5.9 Pearson correlation coefficients [27]

Being $\rho_{X,Y}$ the Pearson correlation between two variables X and Y. It is calculated by the covariance of the variables divided by the product of both individual standard deviations σ_x and σ_y. It can also be expressed with the expectation as shown in the next equation.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_x \sigma_y} = \frac{E[(X-\mu_Y)(Y-\mu_Y)]}{\sigma_x \sigma_y} \tag{5.6}$$

5.2.3 Information theory

Information theory is normally used for signal processing and compression and for channel coding but it can also be used to compare information. Information theory has many measurement information measurements such as entropy, mutual information and information gain. For the current work, two main scoring indicators are used: Kullback-Leibler divergence and mutual information.

Kullback-Leibler divergence also known as relative entropy or information gain is used to compare how one probability distribution is different from a second one. A Kullback–Leibler divergence of 0 indicates that there is no different between the two distributions. Then, for the present work, values different than 0 will quantify how divergent are the two signals. The *equation 5.7* formulates the Kullback-Leibler divergence between two probability distributions P and Q.

$$D_{KL}(P,Q) = -\sum_{x \in X} P(x)\ log\left(\frac{Q(x)}{P(x)}\right) \tag{5.7}$$

Another interesting measurement is mutual information which compares the mutual dependence between two variables. Basically, it compares how similar is the joint the joint distribution of the pair (X,Y) is to the product of the marginal distributions as shown in *equation 5.8*.

$$I(X,Y) = \sum_{x \in X} \sum_{y \in Y} P_{XY}(x,y)\ log\left(\frac{P_{XY}(x,y)}{P_X(x) \cdot P_Y(y)}\right) \tag{5.8}$$

Chapter 6

Experiments

6.1 Methodology

Once faults and tests are defined, they have to be combined, faulty data needs to perform those tests in order to check the behaviour against the previously defined indicators. Therefore, first of all synthetically generated errors are tested to get some results enabling generalising indicators' values for some errors. Then, having this information, real captors data from 2017 campaign is used to validate if the previously learnt misbehaving values of the indicators also apply.

The tests can be passed against the whole summer data and against subsets of the data. In order to select the subsets of data, a windowing selection is defined. Each window w_i takes an ordered subset of measurements with size w_s. The first window starts at measurement 0 and the consecutive window starts is incremented by the window hop w_h. The number of windows is determined by I, which is the number of windows of size w_s with an overlap of w_h that the total number of measurements m can have starting at measurement 0. *Equation 6.2* defines how to calculate I.

$$w_i = [i \cdot w_h,\ \ i \cdot w_h + w_s - 1] \qquad 0 \leq i < I \tag{6.1}$$

$$I = \left\lfloor \frac{M - w_s}{w_h} \right\rfloor + 1 \qquad 0 \leq m < M \tag{6.2}$$

Figure 6.1 exemplifies a hopping window with a window size of 8 and a window hop of 4. Therefore, there is an overlap of 4 measurements. Having a total number of 40 measurements, the number of total windows I is 9.

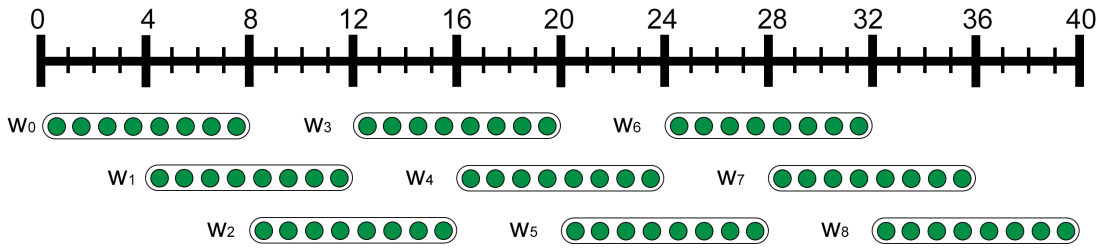


Fig. 6.1 Hopping window with w_s of 8 and w_h of 4

For the experimentation phase, a hopping window of size 48 and a a hop of 24 is used. It has been selected in order to see a full day (48 measurements per day) avoiding false error detections due to single small differences between sensors. It is not bigger to have a predicted error as soon as possible and avoiding being hidden by the scoring system. They could be hidden due to have many correlated data and a small portion of uncorrelated. *Figure 6.2* shows a daily window with 48 measurements and an overlap of a midday.

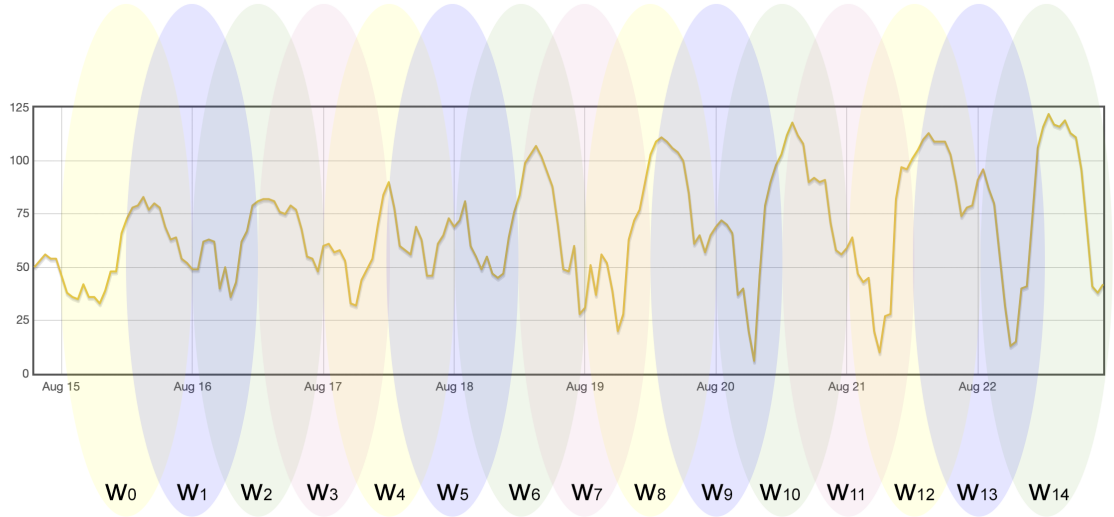


Fig. 6.2 Windowed ozone data

6.2 Synthetic data

Data from *section 5.1* has been used as examples of possible faults. All tests defined in *section 5.2* have been developed in python scripts and processed against synthetical data. As could be extracted from *table 6.1* and *table 6.2* seems that taking all the dataset as a whole is hiding the errors in the scoring results. Therefore, a further study with the window methodology is presented in the next subsections.

Table 6.1 Overall tests for synthetic data, part 1

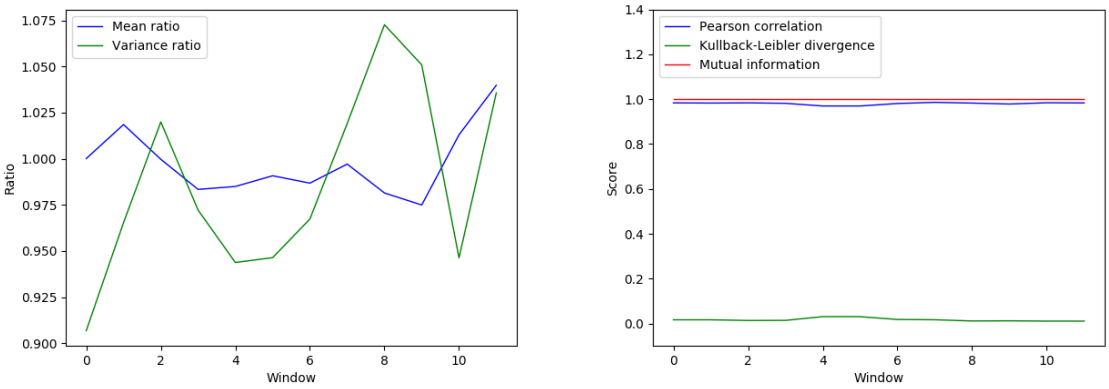
	Normal behaving	Basic	Big error	Increasing error	High/low values
$\overline{v}$	18.088	18.165	18.828	18.347	18.163
σ^2	112.218	113.073	134.783	116.769	71.769
Correlation	1.0	0.979	0.837	0.92	0.959
Entropy	0.0	0.016	0.138	0.058	0.029
MI	1.0	1.0	0.983	0.996	1.0

Table 6.2 Overall tests for synthetic data, part 2

	Normal behaving	Basic	Amplitude	Mean	Time Gap
$\overline{v}$	18.088	18.165	18.081	26.617	18.125
σ^2	112.218	113.073	41.772	141.769	113.889
Correlation	1.0	0.979	0.847	0.882	0.463
Entropy	0.0	0.016	0.073	0.056	0.241
MI	1.0	1.0	1.0	1.0	1.0

6.2.1 Basic

The basic example is the own that both signals are almost the same. As expected all scoring tests show very correlated data as can be extracted from *figure 6.3*. Mean and variance ratios are centered on 1 and they do not change more than a 10% above or below that value. Correlation and mutual information are always over 0.95 whereas divergence is below 0.05.

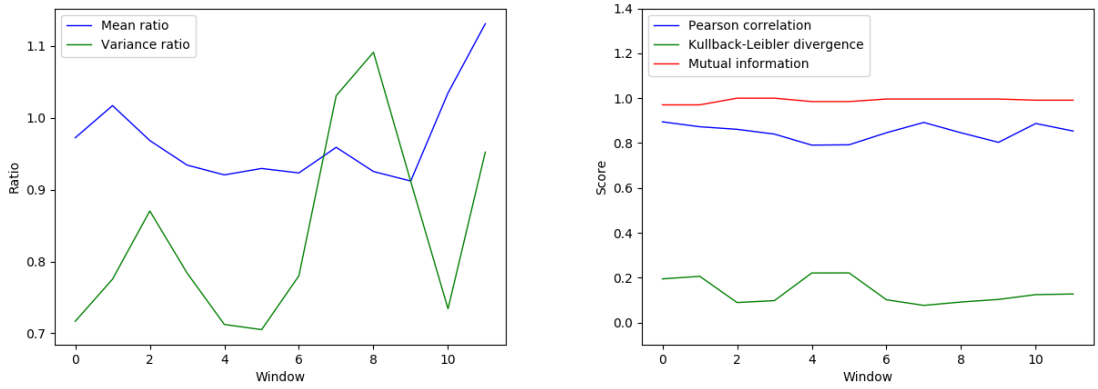


(a) Mean and variance ratios (b) Correlation, divergence and mutual information

Fig. 6.3 Comparing two similar synthetic signals

6.2.2 Big error

The misbehaving signal is characterized by having a big ε. Although, looking into the mean from *figure 6.4a* seams to be correct, the variance ratio stays more unstable. Correlation and divergence are the ones that notice the fault, they are above 0.8 and over 0.2 respectively. Finally, mutual information is almost 1, meaning that it is not noticing difference among them.

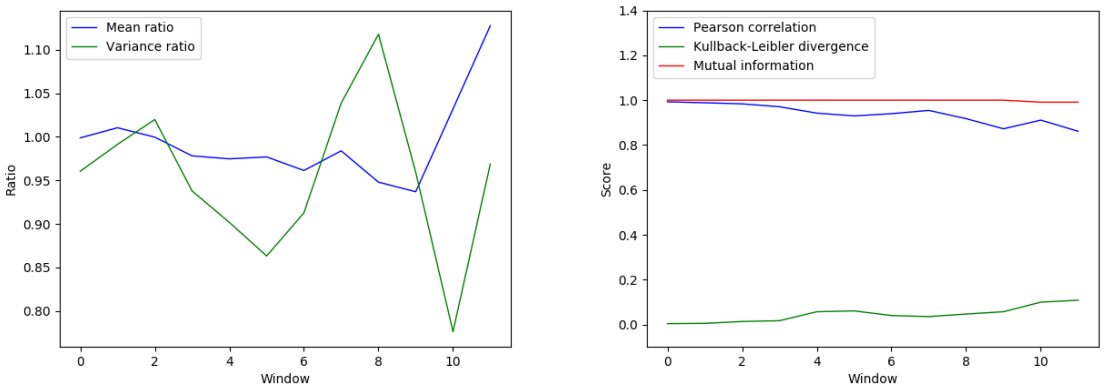


(a) Mean and variance ratios (b) Correlation, divergence and mutual information

Fig. 6.4 Comparing two synthetic signals, one with a big error

6.2.3 Increasing error

This fault is highly comparable with the previous one as both have to be with ε. However, the last day is the one that reaches the same ε than in the previous experiment. It is really visible how on *figure 6.5b* correlation and divergence degrade reaching 0.8 and 0.2 again.

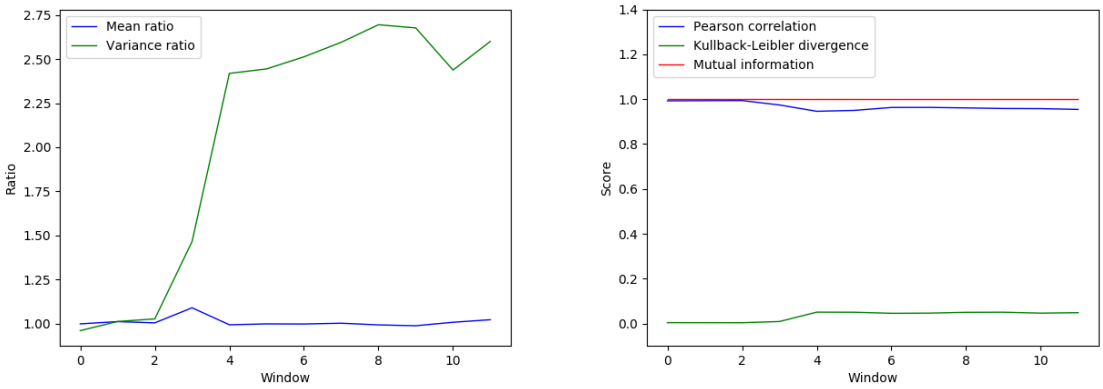


(a) Mean and variance ratios (b) Correlation, divergence and mutual information

Fig. 6.5 Comparing two synthetic signals, one with an increasing error

6.2.4 High/low values

This faulty sensor changes on day 3 by being incapable of measuring high and low values. It is similar to an amplitude change as the only test that notices it is the variance ratio which has been multiplied by 2.5 in only three windows of difference. Other scoring results are acceptable as could be seen in *figure 6.6*.

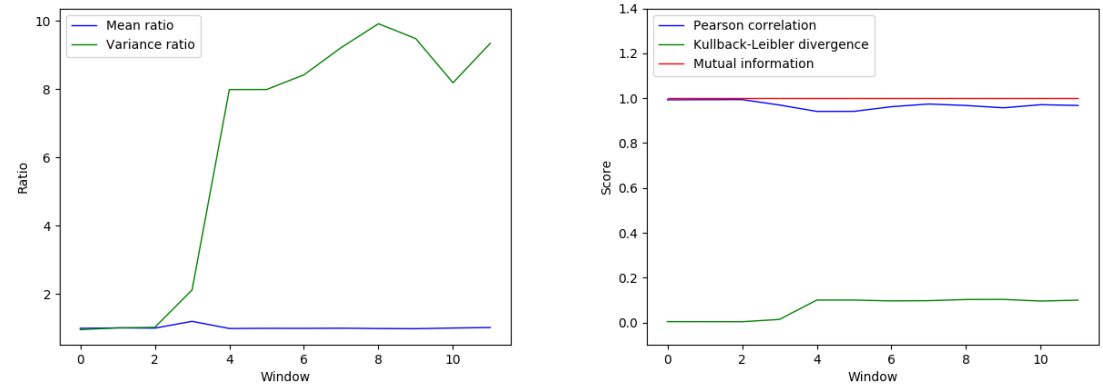


(a) Mean and variance ratios (b) Correlation, divergence and mutual information

Fig. 6.6 Comparing two synthetic signals, one with a threshold

6.2.5 Amplitude

As in the previous subsection, this is directly related with the amplitude. Looking into *figure 6.7a*, variance ratio multiplies by 8. By having a greater change on amplitude, another interesting result can be extracted from *figure 6.7b*. While the divergence degrades until almost 0.2, the correlation decreases for a period of time but return to values near 1. It means that correlation return to high values once the signal returns to stable state.

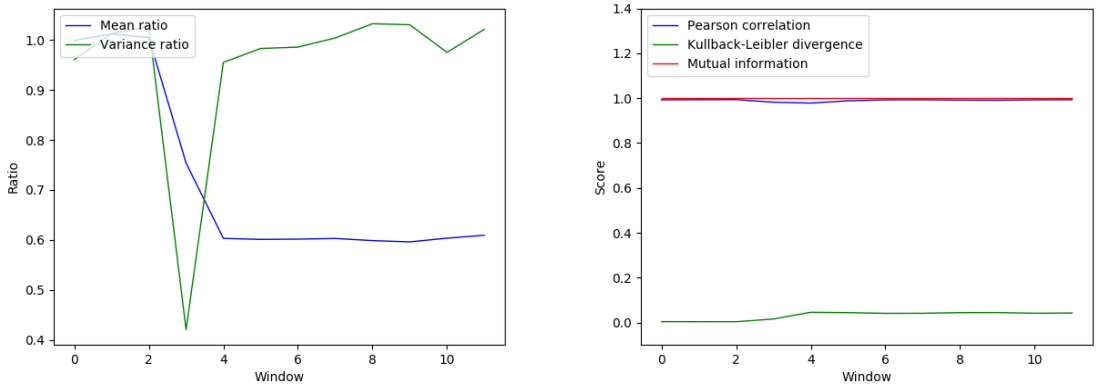


(a) Mean and variance ratios (b) Correlation, divergence and mutual information

Fig. 6.7 Comparing two synthetic signals, one changes its amplitude

6.2.6 Mean

The scoring results of a signal that changed the mean on the third day as shown in *figure* **??**, variance and mean divided by 2 in less than three windows. The difference among them is that, as the amplitude is always the same, the variance ratio turns back to 1. Correlation, divergence and mutual information does not notice it.

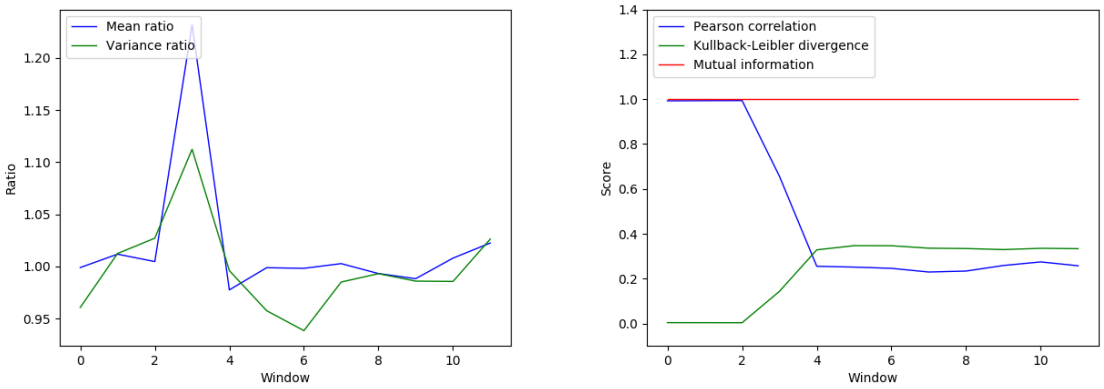


(a) Mean and variance ratios (b) Correlation, divergence and mutual information

Fig. 6.8 Comparing two synthetic signals, one changes its mean

6.2.7 Time gap

The last experiment consists on testing a clock error. While mean and variance ratio are more or less stable and mutual information remains in 1, divergence and correlation show very bad scores.



(a) Mean and variance ratios (b) Correlation, divergence and mutual information

Fig. 6.9 Comparing two synthetic signals, one with a clock error

6.2.8 Partial results

All faults set in the definition section were correctly noticed using ratios and scoring equations such as correlation and divergence. However, mutual information seams that do not apply for the current scenario. Therefore, it will be discarded for the next experiments.

As a summary, all previous experiments are reviewed. Changes in ε degrade correlation and divergence scoring results. However, from mean and variance ratios these errors are not noticeable. Changes in amplitude are clearly noticeable in variance ratios. Changes in mean not only changed drastically mean ratio but also variance ratio. Clock errors drastically degrade correlation and divergence.

This section has also validated the windowing protocol. Having only some general scores with a big data set is not enough, it could hide some low scoring values by only taking the scoring summary. Therefore, real scenario will be also studied using the same time window procol.

6.3 Real data

After all fault and tests are validated by using synthetically generated data against the scoring equations, real data is scored. According to the partial results presented in *section 6.2.8*, all drastic changes and degradations of the scores will be studied empirically and possible causes for each error. Next four subsections shows a good behaving node and three which at some point one or more sensors misbehaved. See that each subsection will have the same structure, a zoom on some part of the time line and twelve comparison charts. These charts are six pairs of sensor vs sensor, having one for ratios and the other for correlation and divergence. Finally, the results are generalized into some rules which will be used to detect errors during next summer campaigns.

6.3.1 Captor 4

Captor number 4 is one of the best of the campaign, it remained good behaving during the whole summer. Although comparing s_4^1 agains the other three it is not perfect behaving, it remains as a good sensor. It can be seen that the mean and variance ratios remained stable for s_4^1. However, all other three sensors compared together did not have any degrade neither drastic changes. By looking into *figure 6.10*, s_4^1 looks like its amplitude was small, which made more inaccurate results but behaved correctly.

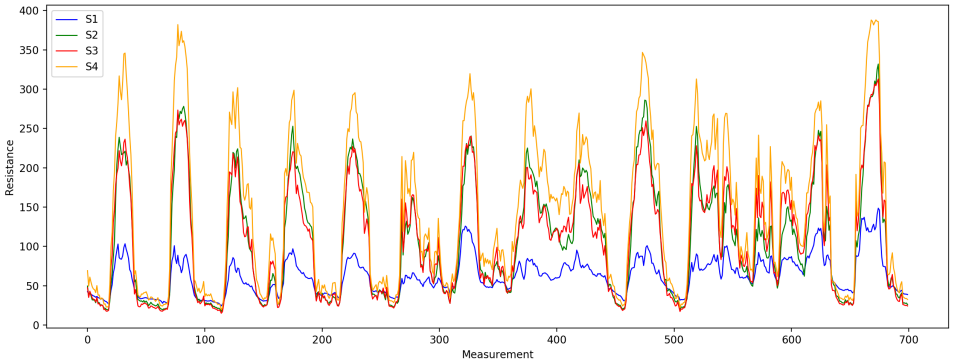
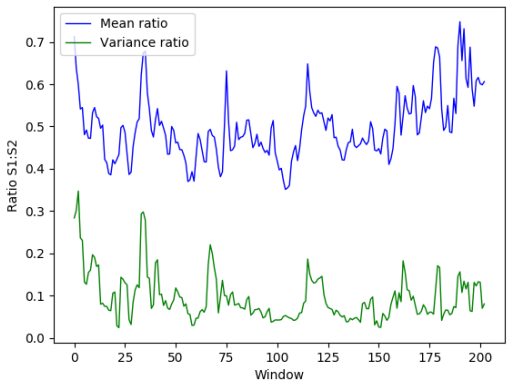
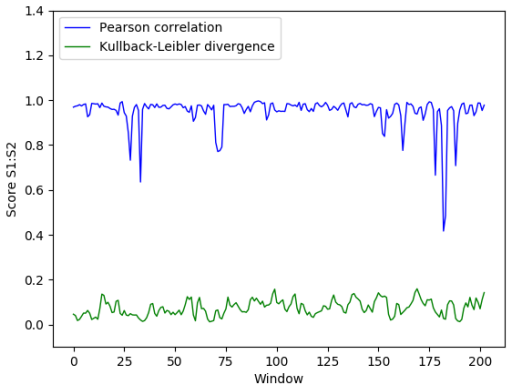


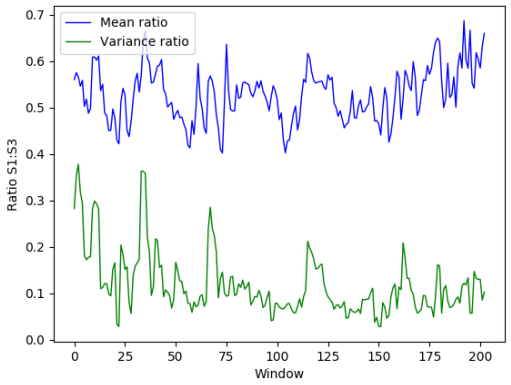
Fig. 6.10 Captor 4 timeline, from measurement 4000 to 4700 (w_{166} to w_{195})



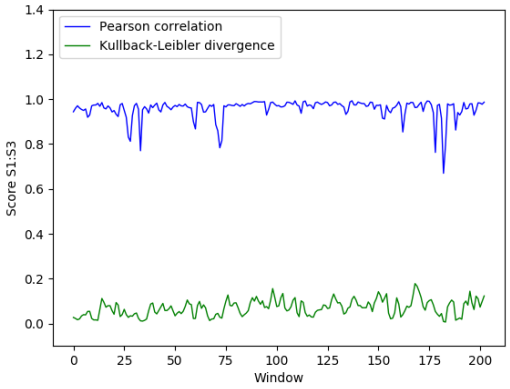
(a) Ratios for s_4^1 vs s_4^2



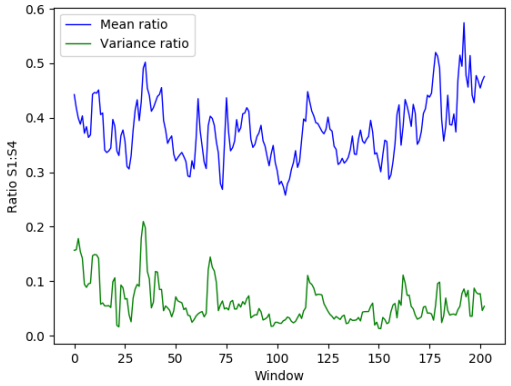
(b) Scoring for s_4^1 vs s_4^2



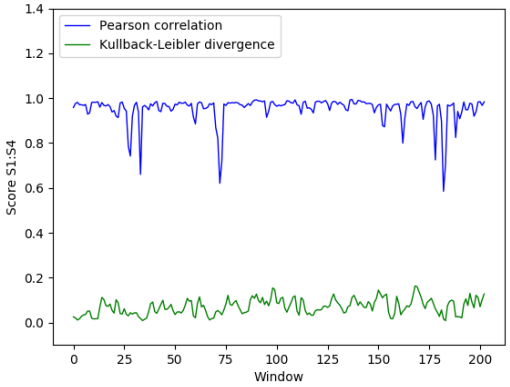
(c) Ratios for s_4^1 vs s_4^3



(d) Scoring for s_4^1 vs s_4^3

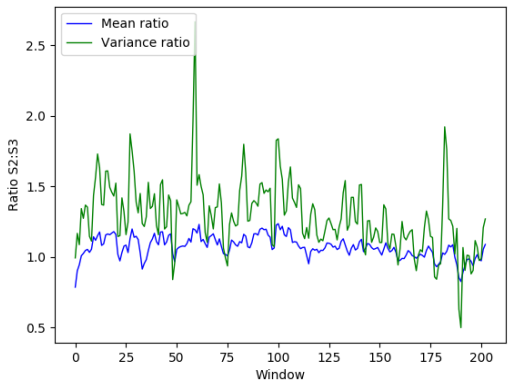


(e) Ratios for s_4^1 vs s_4^4

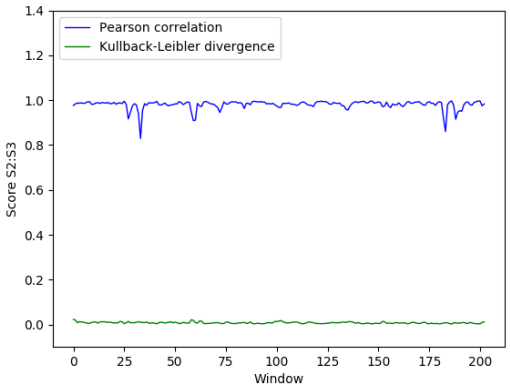


(f) Scoring for s_4^1 vs s_4^4

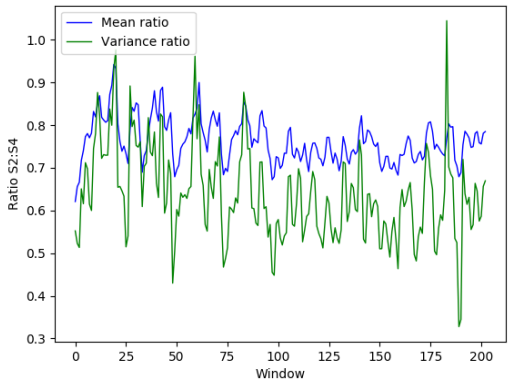
Fig. 6.11 Comparing sensors from captor node 4, part 1



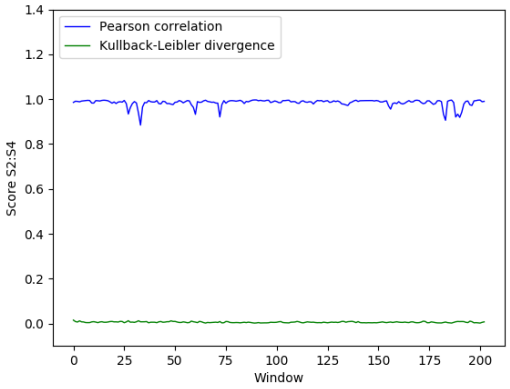
(a) Ratios for s_4^2 vs s_4^3



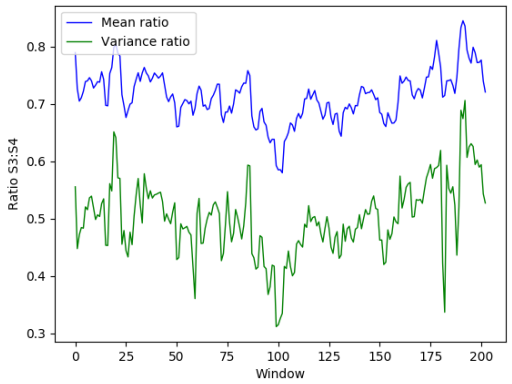
(b) Scoring for s_4^2 vs s_4^3



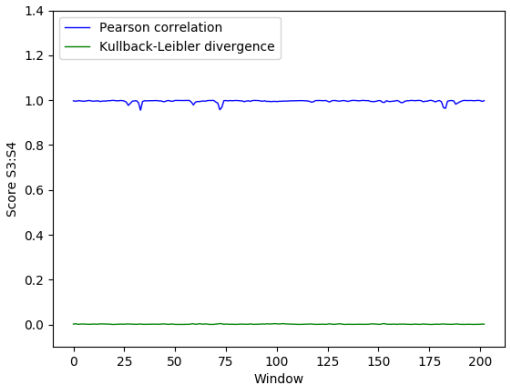
(c) Ratios for s_4^2 vs s_4^4



(d) Scoring for s_4^2 vs s_4^4



(e) Ratios for s_4^3 vs s_4^4



(f) Scoring for s_4^3 vs s_4^4

Fig. 6.12 Comparing sensors from captor node 4, part 2

6.3.2 Captor 6

Captor number 6 behave all the time more or less correctly. However, during the first week of campaign, sensor s_6^3 had high decrease on the correlation. Making a zoom on that data in *figure 6.14*, shows that s_6^3 that had more or less the same values than s_6^2 since from measurement 90 until the 200 their distance change considerably. The same error seams to happen when comparing s_6^1 and s_6^4.

An important insight to take from this experiment is that once it is marked as misbehaving it is hard to know if the scoring goes normal again, in other words, if the sensors is behaving in the same way as it did before. As in this example it returns to its normal behaviour, a good indicator could be how stable divergence has remained.

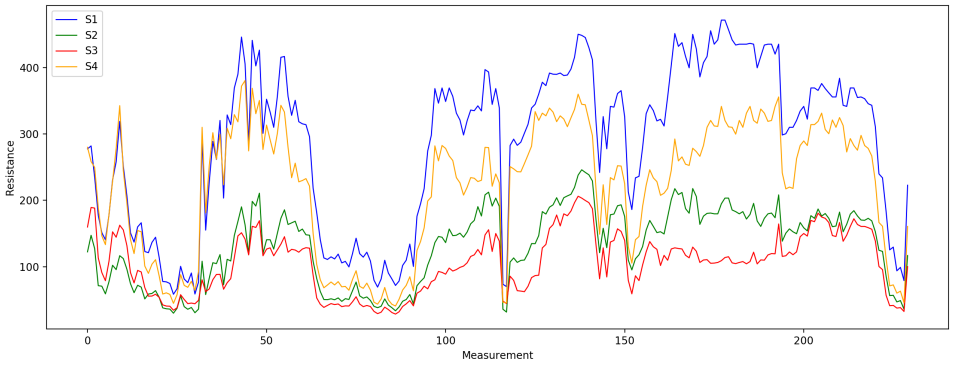
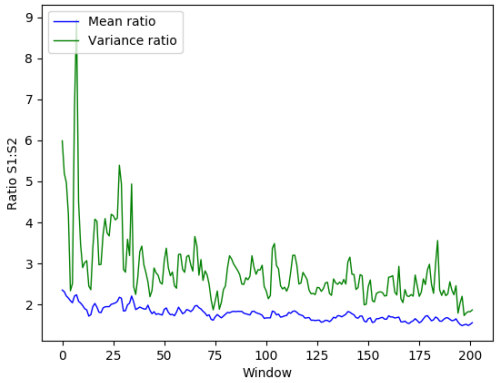
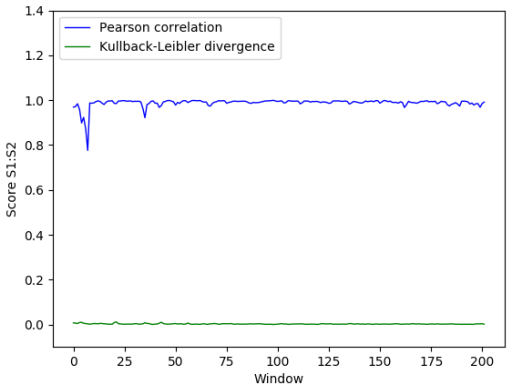


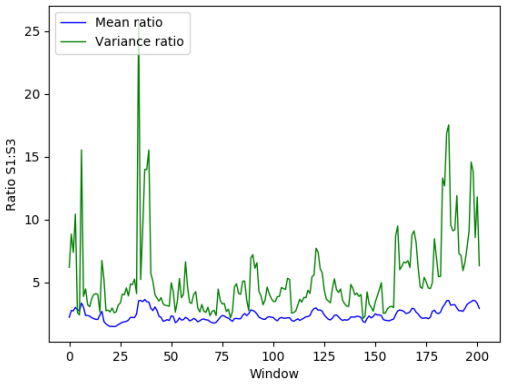
Fig. 6.13 Captor 6 timeline, from measurement 0 to 250 (w_0 to w_{11})



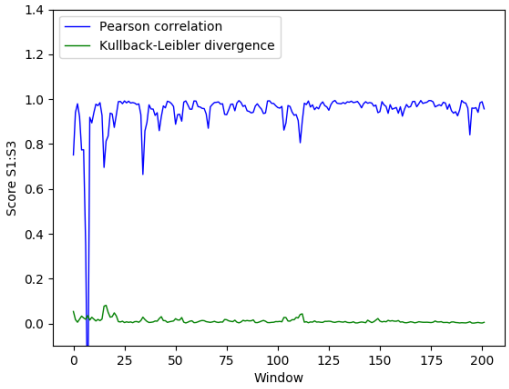
(a) Ratios for s_6^1 vs s_6^2



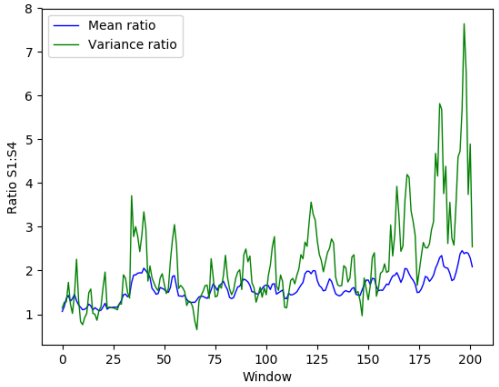
(b) Scoring for s_6^1 vs s_6^2



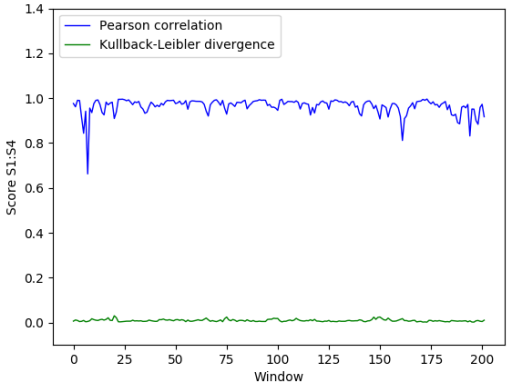
(c) Ratios for s_6^1 vs s_6^3



(d) Scoring for s_6^1 vs s_6^3

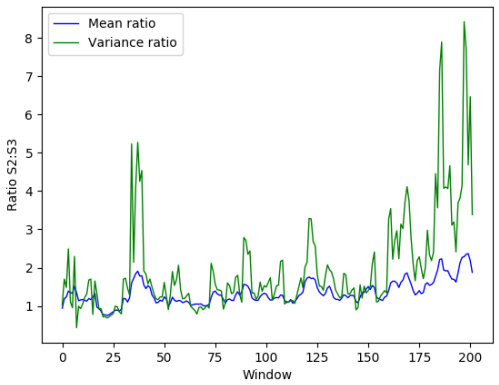


(e) Ratios for s_6^1 vs s_6^4

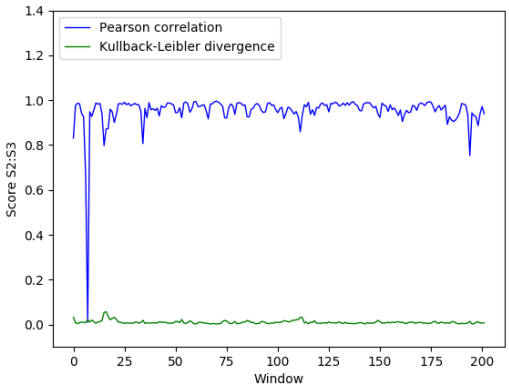


(f) Scoring for s_6^1 vs s_6^4

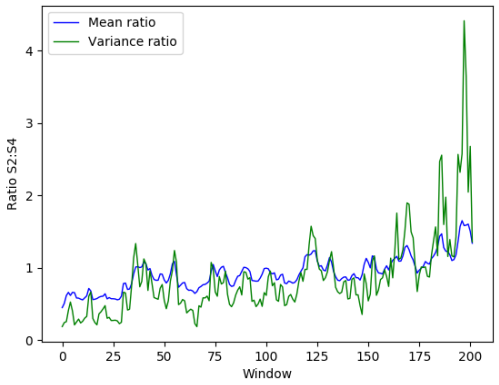
Fig. 6.14 Comparing sensors from captor node 6, part 1



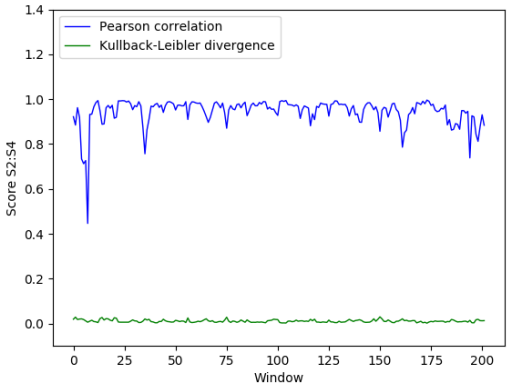
(a) Ratios for s_6^2 vs s_6^3



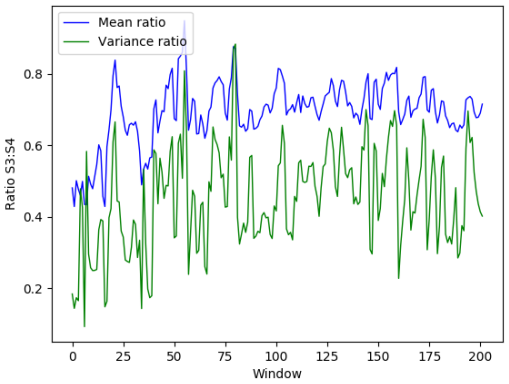
(b) Scoring for s_6^2 vs s_6^3



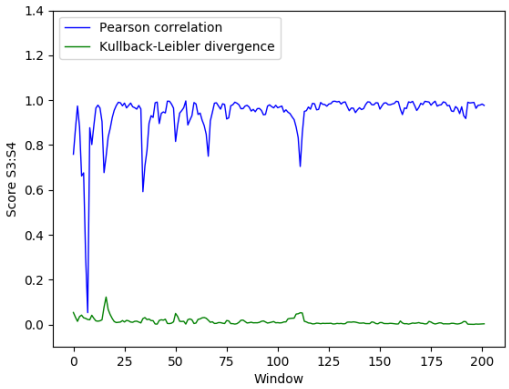
(c) Ratios for s_6^2 vs s_6^4



(d) Scoring for s_6^2 vs s_6^4



(e) Ratios for s_6^3 vs s_6^4



(f) Scoring for s_6^3 vs s_6^4

Fig. 6.15 Comparing sensors from captor node 6, part 2

6.3.3 Captor 8

Captor 8 has a very identifiable error by looking into *figure 6.17*, sensor s_8^1 against all other sensors has a big degrade in correlation from w_{75} until w_{100} and it has a peak of 20 times its variance ratio. However, looking into *figure 6.18*, all other 3 sensors compared together were very good during the whole campaing, stable ratios and good scorings in correlation and divergence.

Studying it more in deep, doing a zoom into the mentioned period, seems that sensor s_8^1 has a recurrent error where sometimes it takes very low values. All other three sensors always follow the same signal without that peaks seen in sensor s_8^1. Therefore, it can be said that it is being detected as an amplitude error.

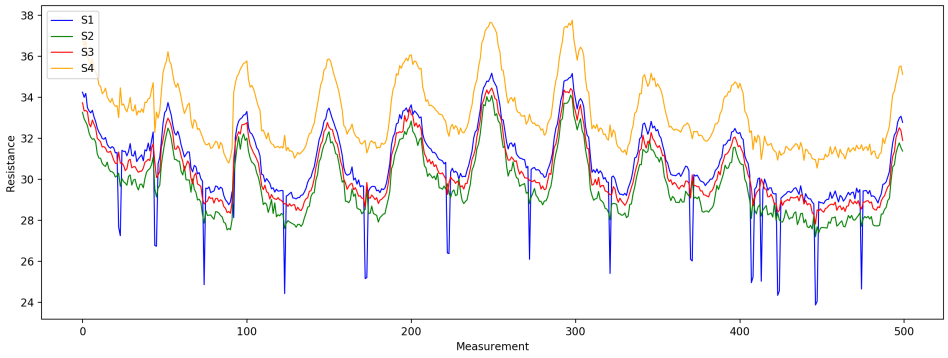
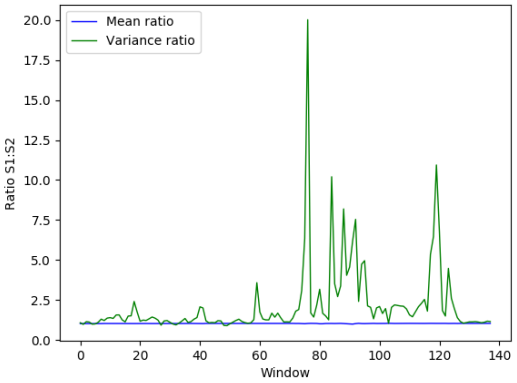
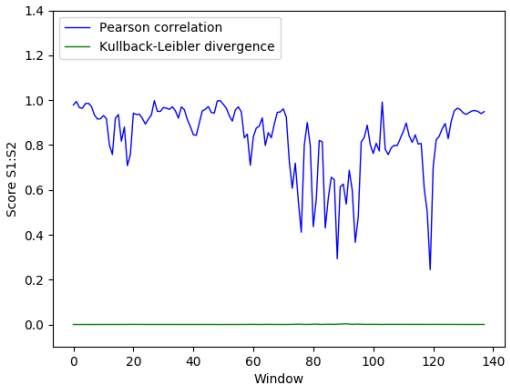


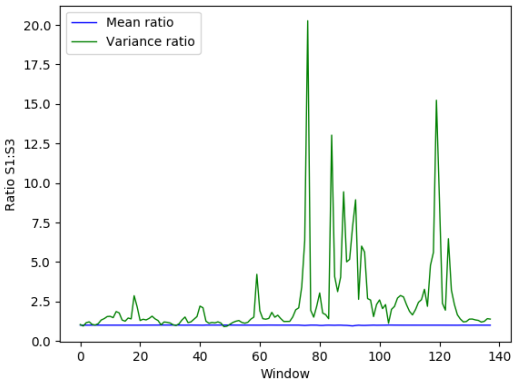
Fig. 6.16 Captor 8 timeline, from measurement 1400 to 1900 (w_{58} to w_{79})



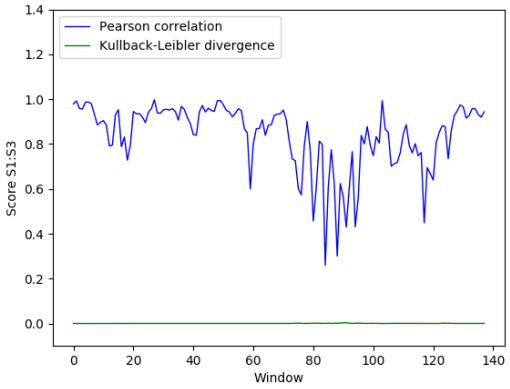
(a) Ratios for s_8^1 vs s_8^2



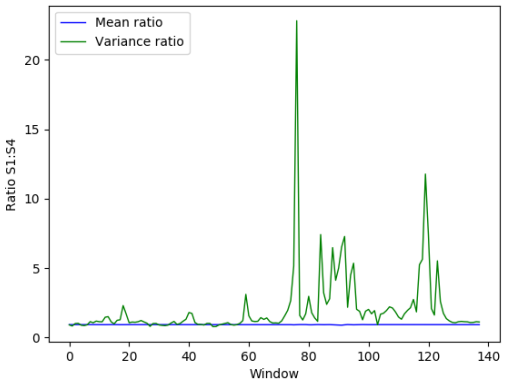
(b) Scoring for s_8^1 vs s_8^2



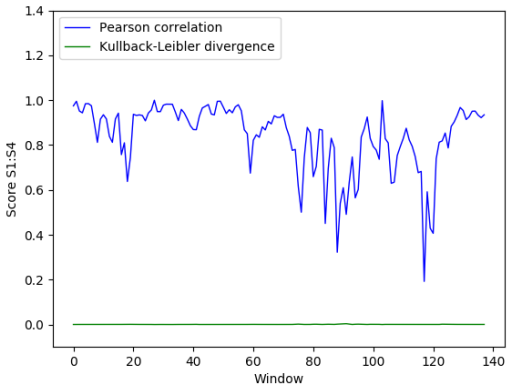
(c) Ratios for s_8^1 vs s_8^3



(d) Scoring for s_8^1 vs s_8^3

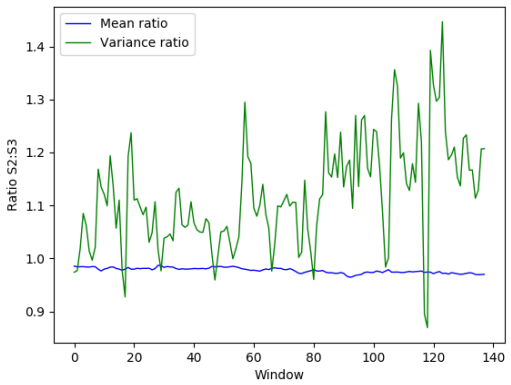


(e) Ratios for s_8^1 vs s_8^4

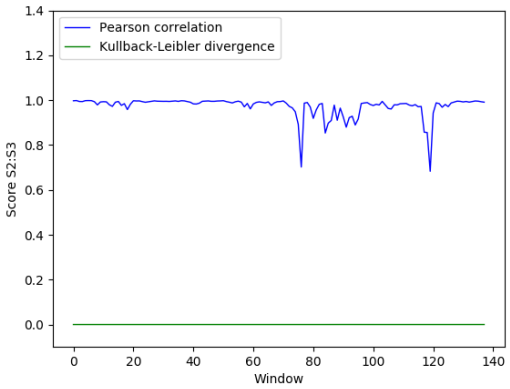


(f) Scoring for s_8^1 vs s_8^4

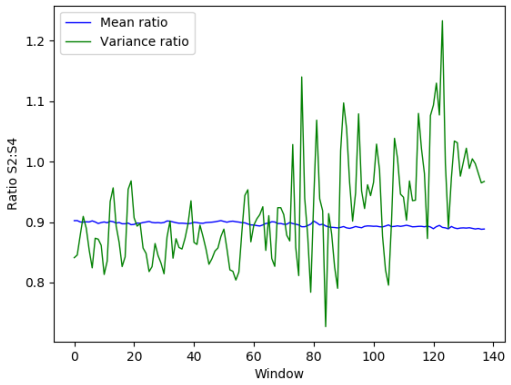
Fig. 6.17 Comparing sensors from captor node 8, part 1



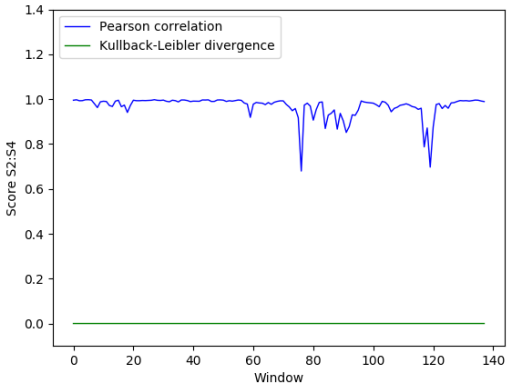
(a) Ratios for s_8^2 vs s_8^3



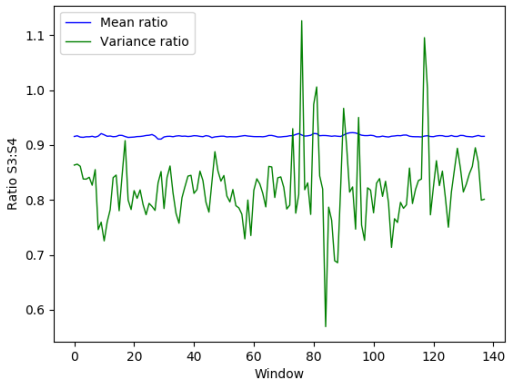
(b) Scoring for s_8^2 vs s_8^3



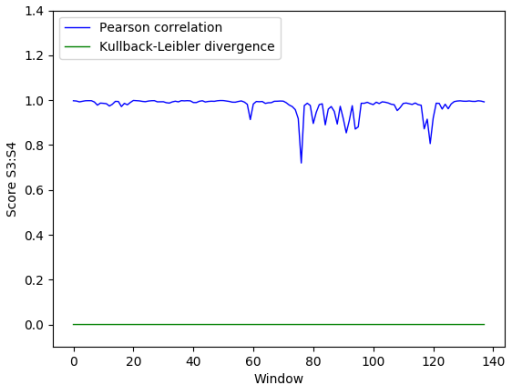
(c) Ratios for s_8^2 vs s_8^4



(d) Scoring for s_8^2 vs s_8^4



(e) Ratios for s_8^3 vs s_8^4



(f) Scoring for s_8^3 vs s_8^4

Fig. 6.18 Comparing sensors from captor node 8, part 2

6.3.4 Captor 10

Captor 10 had one sensor wrong during the whole summer. *Figure 6.20* and *figure 6.21* show that s_{10}^4 against the other three was performing very bag, very unstable ratios, high divergence values and low correlation values. Looking into captor 10 timeline in *figure 6.19*, this sensor was almost a line due to it had small amplitude, being unable to measure ozone.

It was not only sensor s_{10}^4 which was performing bad, during a certain period sensor s_{10}^1 also had some peak faults. From the comparison charts, it is noticeable an error near w_{10} where correlation reached values below 0.4 and divergence above 0.4. Additionally, variance ratio had a peak of 10 times greater than previous values. By checking *figure 6.19*, it is clearly identifiable that s_{10}^1 which remains more or less with the same amplitude than s_{10}^2 and s_{10}^3, it starts giving some higher values in comparison with the other two, having again an error due to amplitude changes.

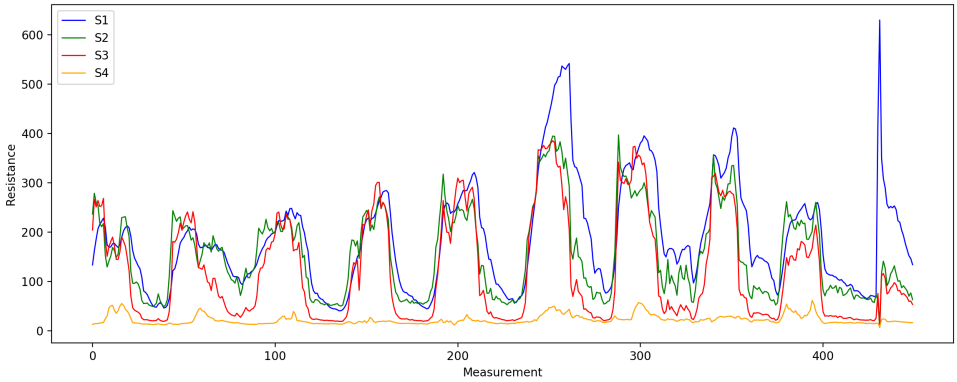
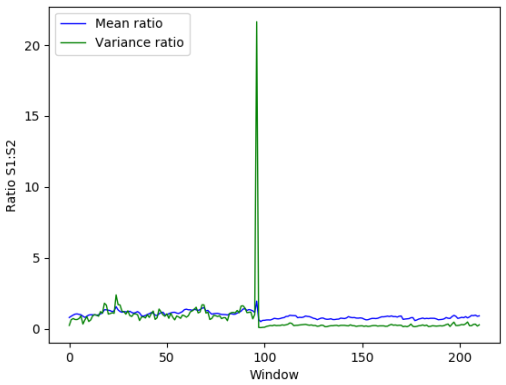
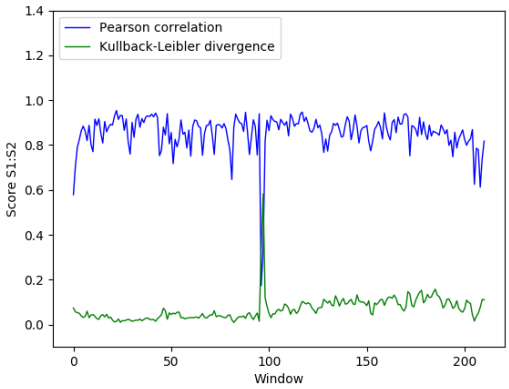


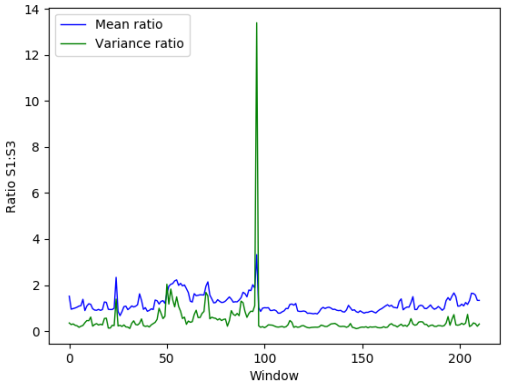
Fig. 6.19 Captor 10 timeline, from measurement 1900 to 2350 (w_{80} to w_{100})



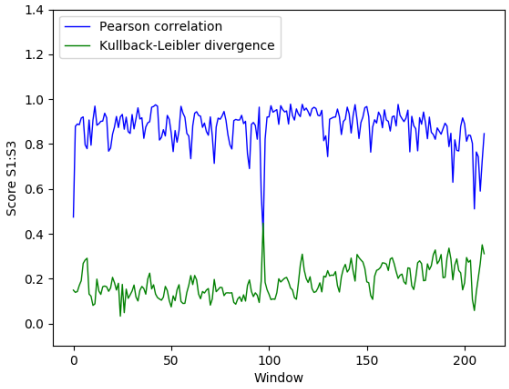
(a) Ratios for s_{10}^1 vs s_{10}^2



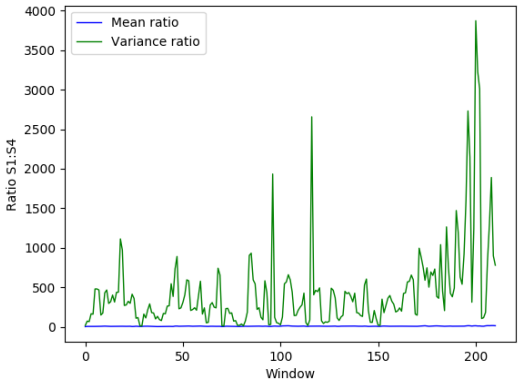
(b) Scoring for s_{10}^1 vs s_{10}^2



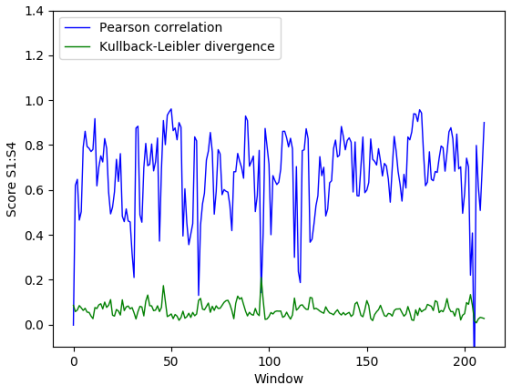
(c) Ratios for s_{10}^1 vs s_{10}^3



(d) Scoring for s_{10}^1 vs s_{10}^3

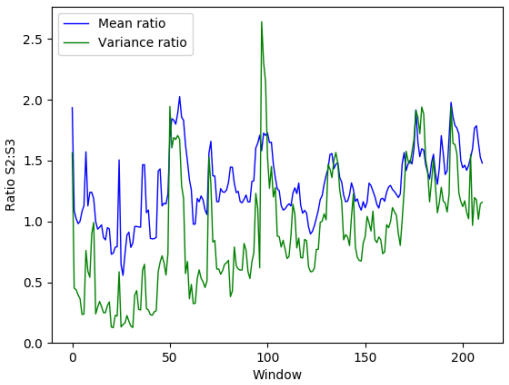


(e) Ratios for s_{10}^1 vs s_{10}^4

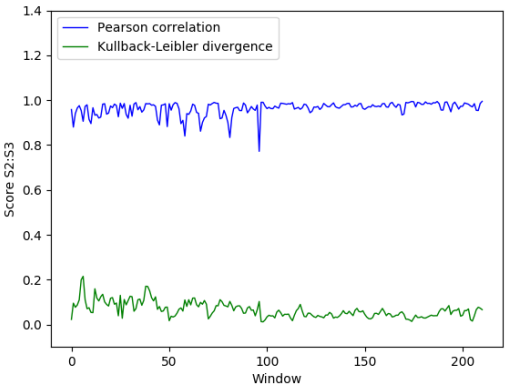


(f) Scoring for s_{10}^1 vs s_{10}^4

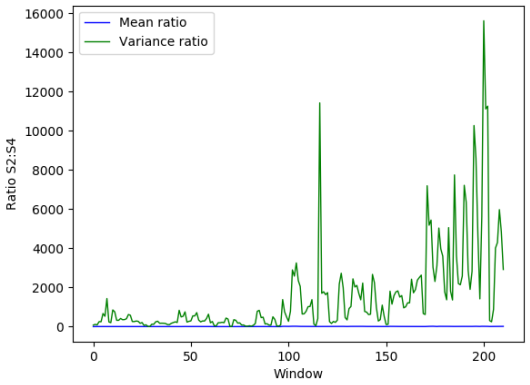
Fig. 6.20 Comparing sensors from captor node 10, part 1



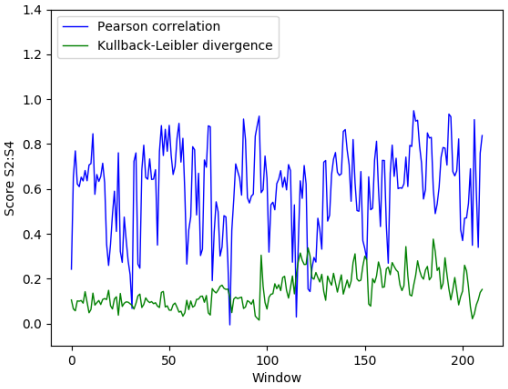
(a) Ratios for s_{10}^2 vs s_{10}^3



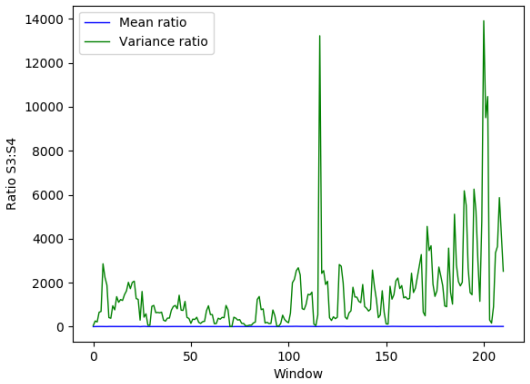
(b) Scoring for s_{10}^2 vs s_{10}^3



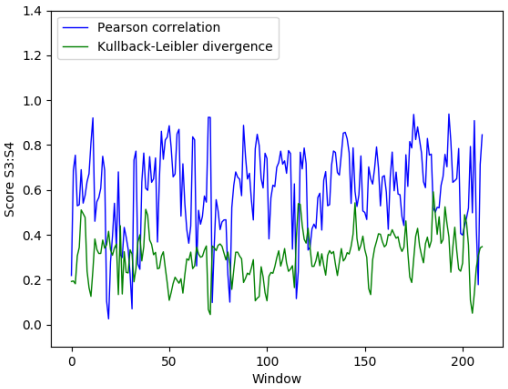
(c) Ratios for s_{10}^2 vs s_{10}^4



(d) Scoring for s_{10}^2 vs s_{10}^4



(e) Ratios for s_{10}^3 vs s_{10}^4



(f) Scoring for s_{10}^3 vs s_{10}^4

Fig. 6.21 Comparing sensors from captor node 10, part 2

6.3.5 Results

The previous subsections has shown that errors can be found by applying some statistical scorings. Most of them were due to amplitude changes over time. This behaviour was clearly identifiable mainly by its changes in correlation and variance ratio peaks. Other sensors, such as s_{10}^4 was detected as a sensor which should be replaced due to it was incapable of measuring ozone.

It is important to meet some agreement in order to quantify *QoI* for each sensor to discard them from showing results to the population. Therefore, a set of generalized rules from the empirical experimentation have been extracted. Given one of the next rules applies for one sensor against the other sensors, it will be marked as a faulty sensor:

- Pearson correlation below 0.5.
- Kullback-Leibler divergence above 0.4.
- Mean ratio multiplies or divides by 3 in less than 5.
- Variance ratio multiplies by 5 in less than 5 windows.

It has been defined a set of very conservative rules avoiding false error detection and allowing to detect those errors that can generate alarms on population or distrust. Once a sensor is market as a faulty, the system should be able to change to the next sensor with the lowest *RMSE* and use its *MLR* calibration coefficients. In case that here are only two sensors and comparison between them shows a faulty sensor, the whole node will be invalidated.

Another result from the experimental phase was to get rid of those sensors that only misbehaved for a certain time, they cannot be considered as false errors but they became tagged as faulty and then recovered. Although it is fairly difficult to know, a good indicator is to check if ratios after an unstable phase, stabilized in the same values before the misbehaving period. This would be an improvement to mark them again as reliable sensors.

Chapter 7

Real time pipeline

Once the solution has been mathematically designed, it is important to design the architecture and data and processing flows. As data is coming in real time, the proposed solution follows a real time event driven architecture. The next two sections presents the proposed architecture solution and the implementation flow solution respectively.

7.1 Architecture

The proposed architecture should be a real time event driven architecture and it should also be reliable, scalable, maintainable and upgradeable. Reliable by being fault tolerant, capable of recover from system faults. Capable to scale if the number of monitorized sensors increases. It should also be easy to maintain, having a solution that allows to check logs in a centralized manner. Last but not least, it is very important to have a system that allows to deploy new software versions incrementally and without downtimes. The solution architecture is shown in *figure 7.1*.

First of all, the overall solution is a containerized architecture, where Docker, Kubernetes and Prometheus play a key role. Docker [28] is the reference containerization open source framework, which allows to build containers capable of running issolated applications without the need of external library installations. Kubernetes [29] is the leader open source container orchestrator. It allows to orchestrate containers on top of a cluster. Kubernetes is capable to deploy and mantain containers, scale them and upgrade the containers version smoothly. Prometheus [30] will be used to monitorize all traces from containers in order to autoscale them using some threshold rules based on load traces. Finally, continuous integration and continuous deployment will be done using Jenkins [31] pipelines.

All the previous technologies cover reliability, scalability, maintainability and upgradability. However, they do not satisfy the core of the solution, the real time event based solution.

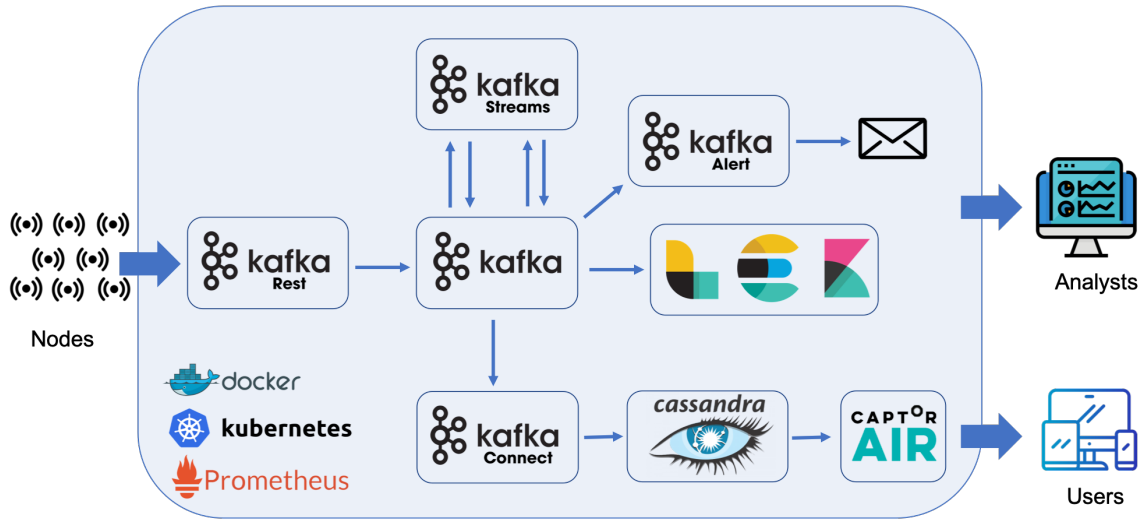


Fig. 7.1 Real time event based architecture

For that purpose, Kafka ecosystem is used. Apache Kafka [32] is a streaming technology which allows to orchestrate distributed event data flows. Kafka differentiates uses named topics to designate each data flow. This technology is characterized by working in a publish subscribe manner, meaning that a set of producers will be publishing data into a topic and one or more consumers could subscribe to the events from that topic.

Additionally, there are many other technologies from the Kafka ecosystem as Kafka Rest Proxy, Kafka Streams, Kafka Connect and many other Kafka client libraries. Kafka Rest Proxy [33] will be used as a Restful API to Post measurements and to publish them into a Kafka topic. Kafka Streams [34] is the processing technology which will compute calibration and sensor classification defined in the previous section. Kafka Connect [35] will persist events in a Cassandra database.

There are some more technologies used in the solution. Apache Cassandra [36] is a NoSQL key-value distributed database and it is used to store all calibrated historical data which is later consumed by end users using CaptorAIR application. Analysts also need to monitorize sensors behaviour, it is done by receiving alerts using a Kafka client whice sends notifications given faulty sensors and using the Elastic stack. Elastic stack (Logstash, Elasticsearch, Kibana) [37] are a set of tools that allow to show dashboards in near real time. Then, analysts will be able to check and compare sensors and scoring metrics as soon as they are computed. Find an example of a Kibana dashboard [38] in *figure 7.2*.

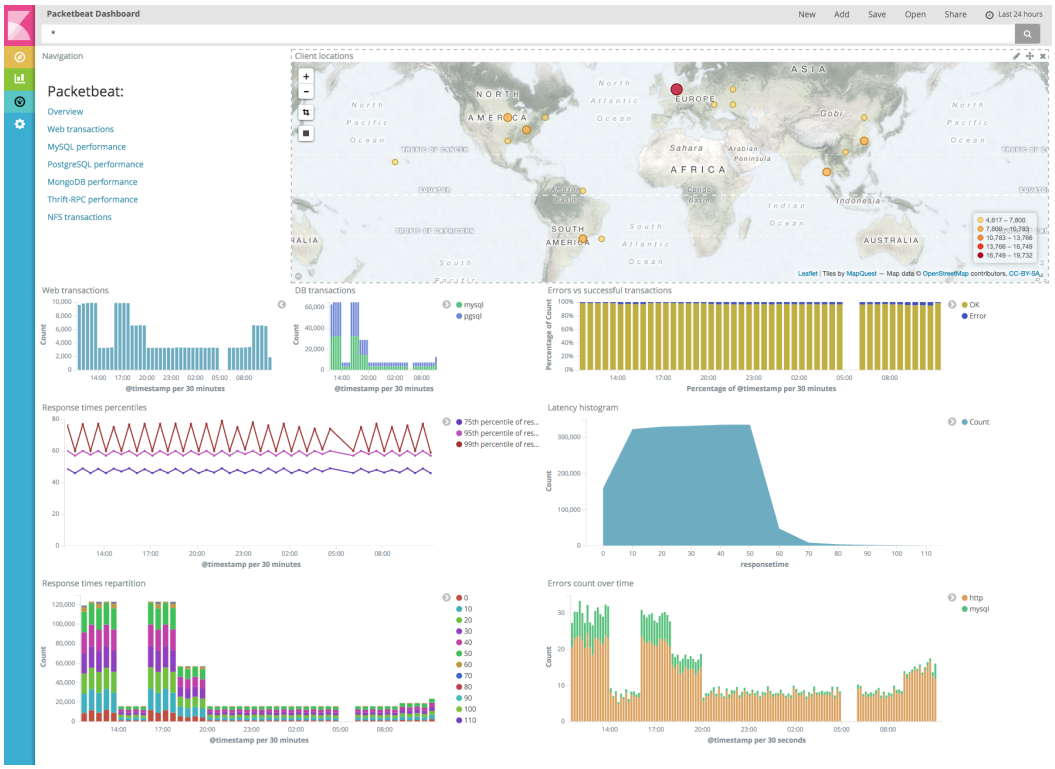


Fig. 7.2 Kibana example dashboard

7.2 Data and processing flows

Once all technologies are selected, it is important to design the flow patterns that events will follow. For that purpose, four topics and a Kafka table are used and four streaming applications are created. All data and processing flows are shown in *figure 7.3*. Pipe objects are Kafka topics, the table is a Kafka table and the wide arrows are stream processing applications.

Before going intro processing flow, it is important to define Kafka topics and Kafka tables. First of all, Kafka Rest Proxy will publish events from nodes in an input *Raw Topic*, where each event would a measurement of all features from a node for a given time. See that they are identified by the node name, which allows later on to group they by id. Then there is *Calibrated Topic* which has the same measurements from *Raw Topic* but with an extra feature which would be the calibrated ozone measurement. The third topic, is the *Scoring Topic* whose events are scoring and ratios from 48 measurements identified again by node. The last one is the *Error Topic* which will only contain events when a sensors has to be tagged as unreliable. The last element is a Kafka table, which allows to store static data and join it

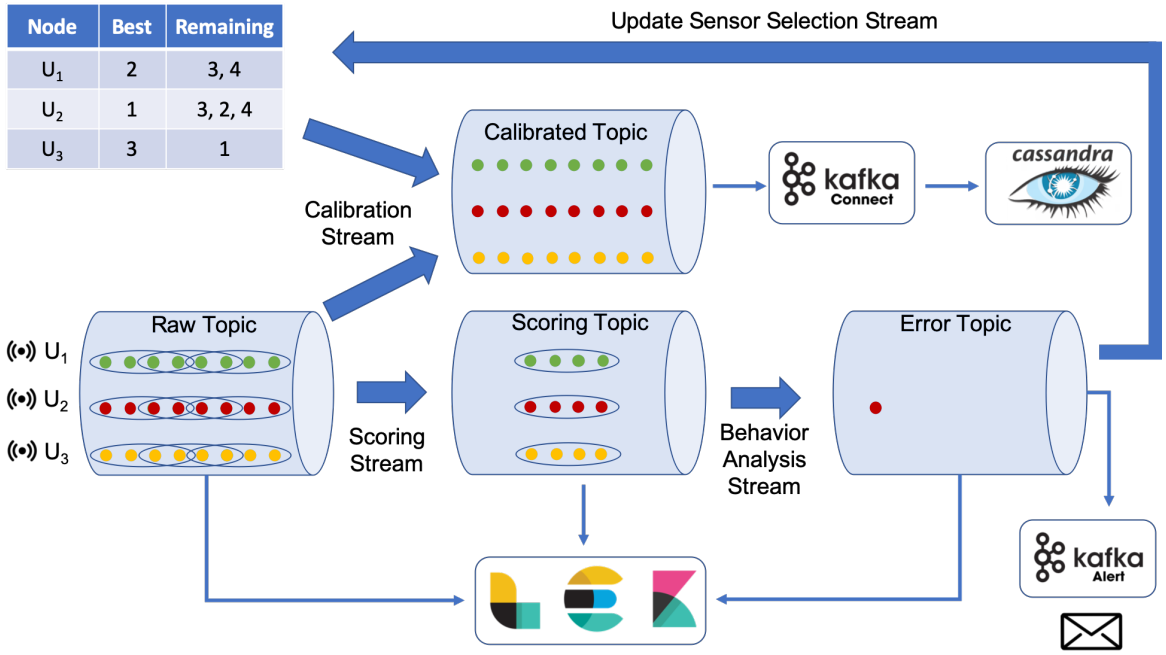


Fig. 7.3 Real time event based data and processing flows

with real time events. It will contain for each node the sensor with lowest calibration *RMSE* which remains reliable.

Once the topics and tables are defined, processing stream applications follow two main flows: calibration flow and monitoring flow. The simplest one would be the calibration flow. It only have one stream applications which takes event by event from the *Raw Topic*, joining it with the current calibration coefficients and publishing it to *Calibrated Topic*. Then a Kafka Connect agent will persist the events in Cassandra to serve them to CaptorAIR application.

The second flow is the one that monitors the sensors' reliability. It has three main stream processing applications. Scoring stream takes events from *Raw Topic* grouped by id and windowed with last 48 measurements. It calculates mean and variance ratios, correlation and divergence. The results are published into the *Scoring Topic*. Then behaviour analysis stream application takes this scores grouped by node and again windowed to compare the last scores from the same node. Given the rules from *section 6.3.5*, if any of them is met, an event will be sent to the *Error Topic*. The las streaming application is the one that takes messages from *Error Topic* and updates the Kafka table to always maintain the sensor with the lowest calibration *RMSE* which remains reliable.

Finally, see that the content from the topics is consumed by Logstash agents which index all the events, scores and error in Elasticsearch. Once they are indexed, they can be easily

analysed using Kibana dashboards. Not only they are being indexed but also the errors are being consumed by a Kafka client which alerts analysts and monitoring staff of sensor faults.

Chapter 8

Conclusions and future work

The present work has assessed *QoI* for low-cost sensor networks to identify possible misbehaving sensors. It started defining a theoretical scenario in which faults and tests were set. The statistical tests were mean ratio, variance ratio, Pearson correlation and Kullback-Leibler divergence. Tests and synthetic generated errors were put all together. The results validates that the tests are capable to notice the errors.

From that partial results, the tests were then applied to a real scenario, which validated statements set in the synthetically generated scenario. Most of the detected errors from the Captor nodes are related with amplitude and unexpected high/low peaks. In both cases, they were clearly identifiable due to Pearson correlation scoring was degraded and the variance ratio changed significantly.

From these experimentation phase, some general rules were extracted empirically. The rules state that if one of them is met, a sensor would be tagged as unreliable. Those rules has been defined very conservative avoiding tagging sensors as unreliable erroneously. It is also proposed a method to untag sensors from being unreliable if some previous scorings were met.

Once all the rules are set, a real time event driven scenario is designed to perform the *QoI* assessment automatically and in a real time manner. Allowing to automatize sensor monitoring while performing proactive sensor selection to maintain always the nodes calibrated. Sensor selection consists on assure that data which is being shown to population is being taken from a reliable sensor with the lowest calibration *RMSE*.

There are some points that can be extended or added to the present work. One would be to move from basic statistical scoring equations to more complex ones or even though to use machine learning. Some unsupervised methods could be used to perform anomaly detection. For example, long short-term memory (*LSTM*) could be applied [39].

Another open point is the distance assessment. All the work has been done by comparing sensors placed in the same node. However, if all the sensors from a node change in the same way, the indicators will remain showing good results which should mean that the behaviour remains as before when it is not. Therefore, rendezvous presented in *section 4.2.3* could be used in order to correlate sensors from different nodes.

The solution proposed in *section 7* should be implemented as it has only been designed in the present work. Having the monitoring tool implemented, measurements will be more reliable as the proposed solution will discard at least those more noticeable errors.

References

[1] Ning Xu. A survey of sensor network applications. *IEEE communications magazine*, 40(8):102–114, 2002.

[2] Vinay Sachidananda, Abdelmajid Khelil, and Neeraj Suri. Quality of information in wireless sensor networks: A survey. *In Proceedings of the 15th International Conference on Information Quality (ICIQ'10)*, page 193–207, 2010.

[3] Jose M. Barcelo-Ordinas, Messaoud Doudou, Jorge Garcia-Vidal, and Nadjib Badache. Self-calibration methods for uncontrolled environments in sensor networks: A reference survey. *Ad Hoc Networks*, 88:142 – 159, 2019.

[4] Kamin Whitehouse and David Culler. Calibration as parameter estimation in sensor networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, WSNA '02, pages 59–67, New York, NY, USA, 2002. ACM.

[5] Laura Balzano and Robert Nowak. Blind calibration of sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, IPSN '07, pages 79–88, New York, NY, USA, 2007. ACM.

[6] David Hasenfratz, Olga Saukh, and Lothar Thiele. On-the-fly calibration of low-cost gas sensors. In *Wireless Sensor Networks*, pages 228–244, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[7] J. M. Barcelo-Ordinas, J. Garcia-Vidal, M. Doudou, S. Rodrigo-Muñoz, and A. Cerezo-Llavero. Calibrating low-cost air quality sensors using multiple arrays of sensors. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, April 2018.

[8] Randolph L. Moses and Robert Patterson. Self-calibration of sensor networks. In *AeroSense 2002*, volume 4743, pages 108–119. International Society for Optics and Photonics.

[9] Laurent Spinelle, Michel Gerboles, Maria Gabriella Villani, Manuel Aleixandre, and Fausto Bonavitacola. Field calibration of a cluster of low-cost available sensors for air quality monitoring. part a: Ozone and nitrogen dioxide. *Sensors and Actuators B: Chemical*, 215:249 – 257, 2015.

[10] Laurent Spinelle, Michel Gerboles, Maria Gabriella Villani, Manuel Aleixandre, and Fausto Bonavitacola. Field calibration of a cluster of low-cost commercially available

sensors for air quality monitoring. part b: No, co and co2. *Sensors and Actuators B: Chemical*, 238:706 – 715, 2017.

[11] Tomoyuki Fujino and Satoshi Honda. Automatic calibration of sensing systems for distributed physical fields. *SICE Journal of Control, Measurement, and System Integration*, 6(3):221–229, 2013.

[12] Saverio Bolognani, Simone Del Favero, Luca Schenato, and Damiano Varagnolo. Consensus-based distributed sensor calibration and least-square parameter identification in wsns. *International Journal of Robust and Nonlinear Control*, 20(2):176–193, 2010.

[13] Chatschik Bisdikian, Lance M. Kaplan, and Mani B. Srivastava. On the quality and value of information in sensor networks. *ACM Trans. Sen. Netw.*, 9(4):48:1–48:26, July 2013.

[14] Kevin Ni, Nithya Ramanathan, Mohamed Nabil Hajj Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Eddie Kohler, Greg Pottie, Mark Hansen, and Mani Srivastava. Sensor network data fault types. *ACM Trans. Sen. Netw.*, 5(3):25:1–25:29, June 2009.

[15] Lu Su, Shaohan Hu, Shen Li, Feng Liang, Jing Gao, Tarek F. Abdelzaher, and Jiawei Han. Quality of information based data selection and transmission in wireless sensor networks. In *2012 IEEE 33rd Real-Time Systems Symposium*, pages 327–338, Dec 2012.

[16] Li Ma, Xiaodu Gu, and Baowei Wang. Correction of outliers in temperature time series based on sliding window prediction in meteorological sensor network. *Information*, 8(2), 2017.

[17] Hao Guo, Zhongming Pan, Zhiping Huang, and Jing Zhou. A flexible framework for assessing the quality of information in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 11(10):485954, 2015.

[18] Kathleen Joslyn and John Lipor. A supervised learning approach to water quality parameter prediction and fault detection. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2511–2514, Dec 2018.

[19] CAPTOR Project. https://www.captor-project.eu/en/. Accessed: 2019-01-24.

[20] CSIC Spanish National Research Council. http://www.csic.es. Accessed: 2019-01-24.

[21] A. Ripoll, M. Viana, M. Padrosa, X. Querol, A. Minutolo, K.M. Hou, J.M. Barcelo-Ordinas, and J. Garcia-Vidal. Testing the performance of sensors for ozone pollution monitoring in a citizen science approach. *Science of The Total Environment*, 651:1166 – 1179, 2019.

[22] Albert Cerezo Llavero. Plataforma de captación de datos IoT: AirAct. Technical report, Universitat Politècnica de Catalunya, 2016. https://upcommons.upc.edu/bitstream/handle/2117/88743/118690.pdf.

[23] CaptorAIR. https://captorair.org. Accessed: 2019-01-24.

[24] CaptorAIR Android App. https://play.google.com/store/apps/details?id=org.airact.captorair. Accessed: 2019-01-24.

[25] Olga Saukh, David Hasenfratz, and Lothar Thiele. Reducing multi-hop calibration errors in large-scale mobile sensor networks. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, IPSN '15, pages 274–285, New York, NY, USA, 2015. ACM.

[26] Olga Saukh, David Hasenfratz, Christoph Walser, and Lothar Thiele. On rendezvous in mobile sensing networks. In Koen Langendoen, Wen Hu, Federico Ferrari, Marco Zimmerling, and Luca Mottola, editors, *Real-World Wireless Sensor Networks*, pages 29–42, Cham, 2014. Springer International Publishing.

[27] Kiatdd. Pearson correlation coefficient. https://commons.wikimedia.org/w/index.php?curid=37108966. License: Creative Commons BY-SA 3.0. Accessed: 2019-04-03.

[28] Docker. https://www.docker.com. Accessed: 2019-04-03.

[29] Kubernetes. https://kubernetes.io. Accessed: 2019-04-03.

[30] Prometheus. https://prometheus.io. Accessed: 2019-04-03.

[31] Jenkins. https://jenkins.io. Accessed: 2019-04-03.

[32] Apache Kafka. https://kafka.apache.org. Accessed: 2019-04-03.

[33] Kafka Rest Proxy. https://docs.confluent.io/3.0.0/kafka-rest/docs/index.html. Accessed: 2019-04-03.

[34] Kafka Streams. https://kafka.apache.org/documentation/streams/. Accessed: 2019-04-03.

[35] Kafka Connect. https://docs.confluent.io/current/connect/index.html. Accessed: 2019-04-03.

[36] Apache Cassandra. http://cassandra.apache.org. Accessed: 2019-04-03.

[37] Elasticsearch, Logstash and Kibana. https://www.elastic.co/en/. Accessed: 2019-04-03.

[38] Kibana example dashboard. https://www.elastic.co/guide/en/beats/packetbeat/5.4/packetbeat-sample-dashboards.html. Accessed: 2019-04-03.

[39] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, Oct 2017.

Appendix A

Hardware and software versions

Hardware

All scripts have run in the same computer. Its specification is described below:

- **Model**: Macbook Pro late 2016 Touch Bar version
- **CPU**: Intel i7-6820HQ 2,7GHz
- **RAM**: 16 GB 2133 MHz LPDDR3
- **GPU 1**: Intel HD Graphics 530 1536 MB
- **GPU 2**: Radeon Pro 460 4 GB

Software

All software used for the present work from text and code editors to language and package versions is listed below:

- **Text editor**: Texpad 1.8.9
- **Text interpreter**: pdfTex 3.14159265-2.6-1.40.19
- **IDE**: PyCharm Community 2019.1
- **Python**: 3.7
- **Python packages**:

- cycler 0.10.0
- kiwisolver 1.0.1
- matplotlib 3.0.3
- numpy 1.16.2
- pandas 0.24.2
- pyparsing 2.3.1
- python-dateutil 2.8.0
- pytz 2018.9
- scikit-learn 0.20.3
- scipy 1.2.1
- setuptools 39.1.0
- six 1.12.0