WSNs Self-Calibration Approach for Smart City Applications Leveraging Incremental Machine Learning Techniques

Conference Paper · November 2016

DOI: 10.1109/NTMS.2016.7792490

CITATIONS
8

READS
217

4 authors:

Rosaria Rossini

19 PUBLICATIONS **97** CITATIONS

SEE PROFILE

Enrico Ferrera

LINKS Foundation (LINKS)

25 PUBLICATIONS **139** CITATIONS

SEE PROFILE

Davide Conzon

LINKS Foundation

23 PUBLICATIONS **275** CITATIONS

SEE PROFILE

Claudio Pastrone

Istituto Superiore Mario Boella (ISMB)

65 PUBLICATIONS **718** CITATIONS

SEE PROFILE

WSNs Self-Calibration approach for Smart City applications leveraging Incremental Machine Learning Techniques

Rosaria Rossini, Enrico Ferrera, Davide Conzon, Claudio Pastrone
Pervasive Technologies (PerT) research area
Istituto Superiore Mario Boella (ISMB)
Via Pier Carlo Boggio, 61, 10138 Torino, Italy
Email: {surname}@ismb.it

***Abstract*—The diffusion of the Internet of Things paradigm, in the last few years, has led to the need of deploying and managing large-scale Wireless Sensor Networks (WSNs), composed by a multitude of geographically distributed sensors, like the ones needed for Smart City applications. The traditional way to manage WSNs is not suitable for this type of applications, because manually managing and monitoring every single sensor would be too expensive, time consuming and error prone. Moreover, unattended sensors may suffer of several issues that progressively make their measures unreliable and consequently useless. For this reason, several automatically techniques have been studied and implemented for the detection and correction of measurements from sensors which are affected by errors caused by aging and/or drift. These methods are grouped under the name of self-calibration techniques. This paper presents a distributed system, which combines an incremental machine learning technique with a non-linear Kalman Filter estimator, which allows to automatically re-calibrate sensors leveraging the correlation with measurements made by neighbor sensors. After the description of the used model and the system implementation details, the paper describes also the proof-of-concept prototype that has been built for testing the proposed solution.**

Keywords: Incremental Machine Learning; Self-Calibration; WSNs; Smart City

I. INTRODUCTION

In the last decades, WSNs have gained more and more importance in many research fields, which have many potential applications in many different facets of our lives [1]. Usually, WSNs are composed by cheap sensors, having limited hardware and energy resources, which can achieve complex sensing tasks working together, rather than expensive standalone traditional sensors. Furthermore, WSNs consist of a large number of sensors that are often deployed in remote or not-easily accessible areas and typically, a WSN node is left unattended for a long period. This is a very annoying issue in Smart City scenarios. This conditions make WSN nodes more prone to errors due to the lack of energy or the harsh conditions of the environment, where they are deployed [1]. For this reasons, WSN nodes can suffer from bias and gain in their measurements and also from drift due to aging [3]. These issues make measurements progressively useless and so the results of the final applications, which leverage on the WSN measures, will be strongly affected by this behavior.

The task of detection and correction of wrong measurements from sensors is known as *calibration*. The relationship between the measured and the real physical signal is known as calibration function. Traditional techniques for sensor calibration consist in a manual adjustment, performed leveraging on a calibrated sensor used as a reference for re-calibration. This technique is too time and effort consuming for a Smart City scenario, where we deal with a big number of sensors geographically deployed. For this reason, there is a significant need for the implementation of sensors auto-calibration techniques [2], more specifically for sensor networks in Smart City environments.

This paper presents a distributed calibration solution, based on a Kalman filter combined with an incremental machine learning algorithm, which constitute a real-time estimator suitable for large-scale scenarios. Furthermore, the authors describe the results obtained by testing this solution in a proof-of-concept environment, in order to validate the proposed solution.

The rest of the paper is organized as follows: section II introduces related works of self-calibration techniques. Section III presents the proposed solution, describing the system model and its actual implementation. Section IV describes the test-bed used to apply the solution in a concrete scenario and the obtained results. Finally, in section V the authors summarize their conclusions about the work.

II. RELATED WORKS

The traditional way for sensor re-calibration consists in a manual operation performed in-situ, using a calibrated sensor as a reference for non-calibrated sensor adjustment. Such calibration technique is too time-consuming and error-prone in a modern, distributed, dense deployed Smart City scenario. For this reason, an automatic calibration technique is needed. This type of techniques are known as auto-calibration or self-calibration, and can be classified into two main categories [4]: Non-blind and Blind calibration. The following subsections describe in detail the two different techniques, analyzing strengths and weaknesses of each solution.

A. Non-Blind Calibration

Non-blind calibration techniques calibrate sensors, leveraging on some reference information [7], [8], [9], [10]. These approaches are based on an a-priori knowledge of some information that can be useful for detecting and correcting non-calibrated sensors. This information is known as ground-truth, which is any high-fidelity measurement of an observed quantity that can be used to validate or verify a new measurement or technique. One simple non-blind technique consists in applying a controlled stimulus to the sensor and monitoring its response. Using ground-truth, it is possible to compare the response with the expected values and then adjust the calibration parameters accordingly [5]. In [6] is described another type of non-blind calibration, based on a manual calibration of a set of sensors, followed by a phase of re-calibration of non-calibrated sensors based on the calibrated set.

Non-blind calibration is used in many different works such as [11],[12], but it is not practical to use them in many scenarios because it needs a lot of effort in large-scale sensors deployments such as Smart City scenarios.

B. Blind Calibration

Blind calibration techniques perform sensor calibration leveraging on redundant information instead of reference data [18], [19], [20]. Blind calibration is a different approach that do not use ground-truth for sensors calibration. Blind calibration scheme is preferable in large-scale deployments, because sensor nodes are hardly accessible and cannot be manually tuned. In literature, blind calibration is mainly based on the assumption that in a very dense sensors deployment scenario, the measures from neighboring sensors are at least slightly correlated among each other. Then, in order to calibrate a sensor, it is possible to leverage on this correlation in order to determine a calibration curve that describes the relation among neighboring sensors, i.e. the model of the physical observation that is measured by the sensors.

The simplest approach for blind sensor network calibration is to assume that the deployment is very dense, so that neighboring sensors have mutual correlated readings [13], [14].

The approach used in [4] assumes that sensors measurements differ linearly from the correct values, this means that they suffer for a gain and an offset. These values can be calculated through a system of linear equations, if the sensors are spaced so close to "oversample" the signal of the physical phenomenon.

Other methods for blind calibration use a-priori knowledge, such as physical models and constraints, of the sensed phenomenon. In [15] geometrical and physical constraints on the behavior of a light source are used to calibrate light sensors with no need of comparing the measurement with a ground truth. The paper assumes that the considered sensors suffer a constant bias with time. In [16] is shown a technique for blind calibration of moisture sensors. It describes the use of a Kalman filter, in order to correct measures sensed by sensors estimating the correct moisture for each sensor. The algorithm leverages on a physical model of moisture used in environmental and civil engineering. Measurements are assumed to be affected by offset and gain.

In [2] is discussed the calibration problem dealing with the concept of drift, i.e. a non-linear systematic error that can affect sensors while aging.

In [27] is described the calibration algorithm depicted in Figure 1.

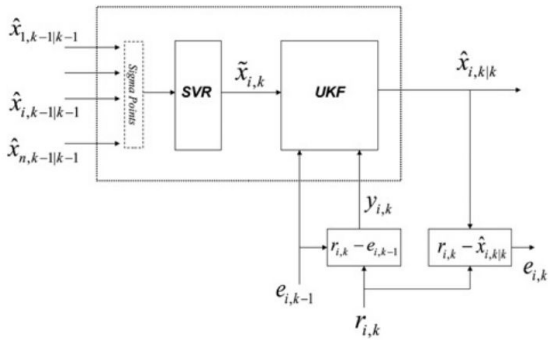

Fig. 1. Calibration framework at single sensor node [27].

The estimation stage is performed using a Support Vector Regression (SVR) algorithms which learn the physical model of the observed physical phenomenon. SVR is quite good as a non-linear regression tool. The estimation stage consists of two phases:

1) *Training phase*: which consists in the physical model learning phase, based on historical data;
2) *Running phase*: which consists in the estimation of the next physical phenomenon measure, based on the model learned at training phase.

The correction stage consists in the Unscented Kalman Filter (UKF) that performs drift and bias detection and correction.

The drawback of this technique is that SVR needs to be properly trained, leveraging on historical data collected during an appropriate period of time.

This approach has several drawbacks, such as the mandatory deployment of a data collection infrastructure and the lack of dynamic adaptability to temporal variations of monitored physical phenomenon, which leads to have a low estimation accuracy. To cope with the latter issue, it is necessary to perform periodical re-training which are time and resource consuming. To overcome such a problem, the authors propose the use of an incremental machine learning technique which is able to adapt its estimation according to the latest sensed data.

After the introduction of the current techniques of auto-calibration, the following sections will focus on describing the approach followed by the authors to design and implement the proposed solution.

III. Proposed Solution

The aim of this work consists in the autonomous recognition of sensor drifts, for instance due to the hardware aging, and correction, leveraging existing correlation among neighbor sensors, which are measuring the same physical phenomenon.

The approach proposed is based on two assumptions: the observed physical phenomenon, e.g. temperature, do not vary quickly in time and space. The neighboring sensors in a network observe correlated data, i.e. the measurements of one sensor are somehow spatially and temporally related to the measurements of its neighbors. In this way, it is possible to have an estimation of data measured from one sensor, using data from other sensors, which are spatially close to it.

In the next subsections we present the model used and the test environment created.

A. Model

The proposed model relies on an incremental machine learning technique and an UKF Kalman filter. The reference architecture is shown in Figure 1 and it is composed of two phases: firstly the model is extracted by a *Training Phase* and then, the status of the sensor (i.e., the measurement of the sensor) is updated, in the *Correction Phase*.

More specifically, given a set of neighboring sensors $S = s_1, s_2, ..., s_n$ and the set of the related observations, regarding a specific phenomenon $\mathbf{O} = o_1, o_2, ...o_n$, the sensor s_x will collect all the observations $\mathbf{O}^t$ at time t and compute the prediction $\mathbf{x}_t$ of the next measurement. The predicted value $\mathbf{x}_t$ and the current value read o_t^x from the sensor s_x will be used to refine the prediction and provide the final value of the phenomenon $\tilde{\mathbf{x}}_t$ and the error value computed between the observed value o_t^x and the predicted value $\tilde{\mathbf{x}}_t$.

As stated above, the first phase, *Training Phase*, allows the sensor to collect the neighboring observations and to learn incrementally the model of the phenomenon, by means of Incremental SVR algorithm [24].

The SVR is a machine learning algorithm, which classifies data by means of regression analysis. The algorithm takes a set of input data and it is able to predict and create a model of such data. The resulting model, after been trained from a series of examples, can successfully predict the output at an unseen given input. The aim is to identify a regression function $y = f(x)$ that, given a set of inputs (x_i, x_j), predicts the outputs y_i with minimum error. The regression function can be described by the following equation:

$$\mathbf{y} = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot K(x_i, x_j) + b \tag{1}$$

where α_i and α_i^* are the Lagrangian multipliers associated with the constraints, the $K(x_i, x_j)$ is the kernel function computed on two vectors x_i and x_j in the feature space $\phi(x_i)$ and $\phi(x_j)$, such that $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$. A typical example of kernel function is the Gaussian kernel [28].

As observed, increasing the training samples, the rate of SVM algorithm decreases and the memory space required, massively increases [23]. Then, for our purpose, in order to overcome this limitation, an incremental algorithm is needed to increase the speed of training and reduce the memory space, specifically, the "Incremental Support Vector machine" [29].

The incremental version of SVR relies on the same regression function, i.e., equation (1), but differs from the classical SVR in the way in which the coefficients α_i and α_i^* are computed. In particular, for each new sample data x_i these coefficients are adjusted in a finite number of discrete steps until the Karush-Kuhn-Tuker (KKT) conditions are satisfied [25], [26].

The second phase of the algorithm is the *Correction Phase*, which leverages a Kalman filter to perform the correction. This technique is massively used with neural networks [21], [30] and, in general, it is useful to obtain estimation of the states of a dynamic process [22].

Such a filter is a well-known method to estimate the state $\mathbf{x}_t$ of a dynamic process at each time t. The estimation $\tilde{x}_t$ is obtained, balancing prior estimations and measurements of the process $\mathbf{x}_t$ by means of the Kalman gain matrix. This matrix is implied to minimize the mean–square–error

$$E[(\tilde{\mathbf{x}}_t - \mathbf{x}_t)(\tilde{\mathbf{x}}_t - \mathbf{x}_t)^T]$$

.

Let the dynamic of the process be described by the following equation:

$$\mathbf{x}_{t+1} = A_t \mathbf{s}_t + B_t \mathbf{u}_t + \mathbf{p}_t \tag{2}$$

where $\mathbf{u}_t$ is the optional control input and $\mathbf{p}_t$ is the white noise. Matrices A_t, B_t are used to relate the process state at the step $t+1$ to the t–th process state and to the t–th control input, respectively.

Introducing the direct measurement values of the process $\mathbf{m}_t$ as:

$$\mathbf{m}_t = \mathbf{x}_t + \mathbf{r}_t$$

where $\mathbf{r}_t$ represents measurements uncertainty. Given that, the estimation $\tilde{\mathbf{x}}_t$ produced by the Kalman filter can be written as:

$$\tilde{\mathbf{x}}_t = \mathbf{s}_t^- + K_t(\mathbf{m}_t - \mathbf{x}_t^-) \tag{3}$$

where K_t is the Kalman gain matrix. The motivation about using Kalman filter relies on its widespread application in several fields to optimize performances, in fact it produces optimal estimations under diverse and well-known criteria. An exhaustive explanation of the filter is outside the scope of the paper and it can be found here [21].

B. Calibration framework

In this section, the use of the filter in combination with incremental machine learning algorithm to create a real-time estimator is described. As stated above, the calibration of the sensors relies on the outputs of the neighbour sensors and the method has the following steps:

1) Collect the observations of the neighbour sensors that measure the same phenomenon;
2) Calibrate the sensor:

a) Estimate the next measurement
b) Refine the prediction, denoising it.

The proposed solution is a decentralized and distributed system. In Figure 2 is shown the deployment architecture proposed.

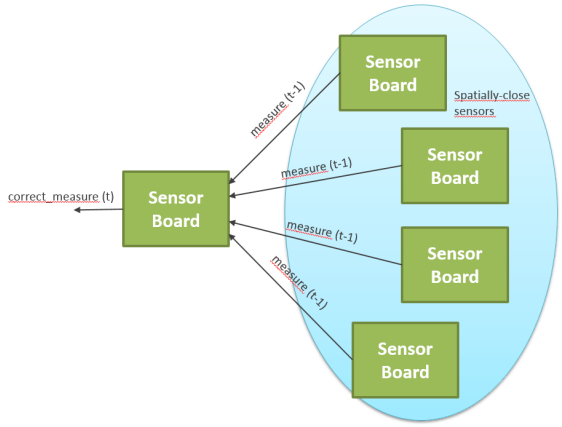

Fig. 2. Deployment architecture.

A number of spatially-close sensors send the measures, related to the previous sensing time, to the sensor to be calibrated. The sensor receives these data and processes them through the calibration framework, in order to provide a calibrated value. The logical building blocks of the calibration framework are shown in Figure 3.

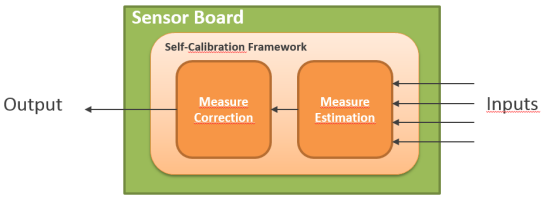

Fig. 3. Proposed calibration framework.

Leveraging on existing environmental sensor board (presented in the next section), the proposed solution consists in the implementation of the calibration framework on the sensor board itself. The calibration framework is composed by two different blocks:

- The *estimation block*, which performs the forecasting of the measures through the model of the physical phenomenon to be sensed.
- The *correction block*, which performs a correction to the estimated measure based on the error at previous step and the actual reading from the sensor.

As described in Section II, one possibility to solve the issue of calibrating sensors is to train neural networks to determine models for measures estimation. This training phase is based on a training set composed by historical data. Such approach is however characterized by high computational complexity and typically long processing time. A different approach leverages online machine learning techniques to deliver the same results in more scalable fashion, without leveraging large data stores. Instead, in order to correct values from the estimation block, the authors have chosen the Kalman filter because of its widespread application in several fields to optimize performances.

IV. PROOF-OF-CONCEPT

The proposed solution has been tested on a scenario consisting in an indoor environment extensively deployed with a number of environmental sensors. In Figure 4 is shown a representation of the deployment which has been done inside the Pervasive Technologies lab at Istituto Superiore Mario Boella (ISMB) research institute.

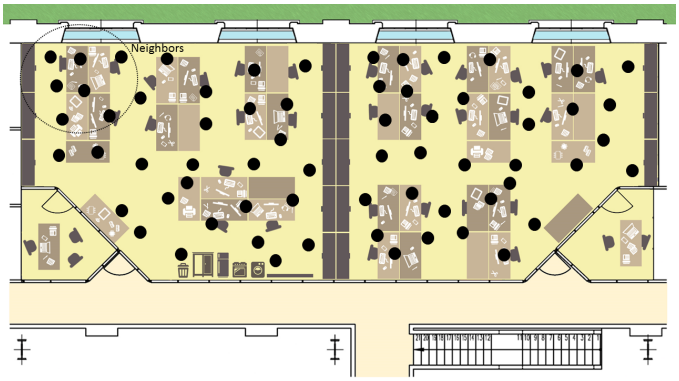

Fig. 4. System deployment.

This section describes the proof-of-concept built to test the solution described in this paper, indicating the hardware and software used and how it has been implemented. Finally, the results are presented.

A. Demo

The solution described in the previous section has been actually tested by the authors in a real environment in order to have concrete results to validate the solution proposed. For the proof of concept a Wireless Sensor Network (WSN) has been deployed using the sensor nodes shown in Figure 5.

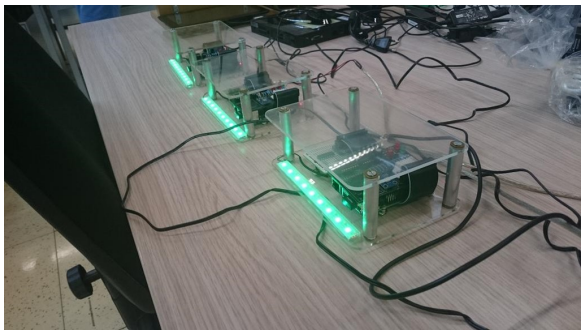

Fig. 5. Proof of concept Architecture

Every node is constituted by these three components: a raspberry PI, an AirPI shield kit and some leds (Figure 6).

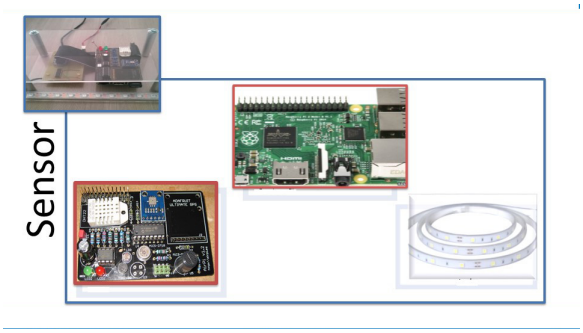

Fig. 6. Components of one sensor node

The Raspberry PI[1] is one of the most popular single-board computers available on the market, thanks to its modular architecture it can be used for different purposes, based on which sensors are connected to the board. In this case the AirPI shield kit is used to monitor the quality of the air. The AirPI shield kit[2] allows monitoring several environmental parameters, like light level, pressure, humidity, temperature and others relative to the pollution of the air. Here, we focus on the light-sensor, a photocell that varies its resistor value according to the light intensity.

Every sensor node of the WSN needs to send its values and to receive the ones measured by other sensors. To allow this communication, the MQTT protocol has been used. MQTT[3] is a standard messaging protocol, based on the publish-subscribe paradigm. Specifically, for the presented proof-of-concept, an MQTT broker has been deployed in a local web server, so that all the nodes can use it to send their data and to subscribe itself to the values sent from other nodes.

Every node use the data received by the others to calculate how much its value is calibrated, using the proposed calibration framework, presented in the previous section. The result of this operation is visually shown, though the leds connected to every sensor node: there are 10 leds, which can be red or green, and they represent the current level of calibration of the node; the scale of quality of the calibration goes from: 10 green leds, which represents the sensor perfectly calibrated, to 10 red leds for the sensor completely misaligned.

To allow the monitoring of the trend of the measured values, the proof of concept has been connected also to the Xively platform[4]. Xively is a cloud IoT platform, which allows the users to connect their products and to monitor the data they produce on a web interface. In this case, the calibrated values measured by the sensors are sent also on Xively, where their trend can be monitored, through the charts provided by the platform.

[1]https://www.raspberrypi.org
[2]http://airpi.es/
[3]http://mqtt.org/
[4]https://www.xively.com/

B. Results

The method has been tested using the light-sensor on the WSN node. In particular, some results obtained by monitoring the outputs are shown in both Figure 7 and 8.

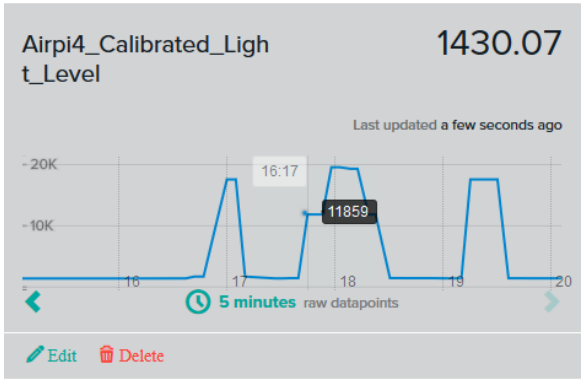

Fig. 7. Calibrated values

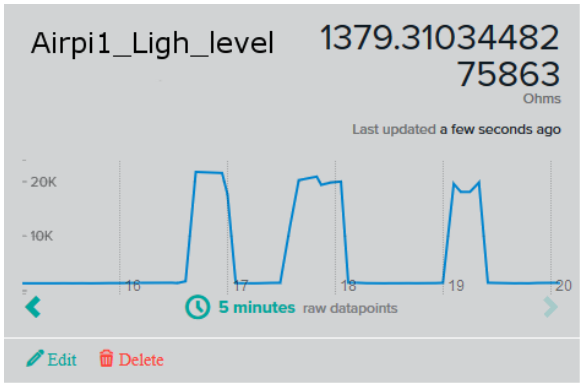

Fig. 8. Values of neighbor sensors

The first figure shows the trend of the light value measured by the sensor ”4” under calibration, instead the second one is shown the light value, sensed by one of its neighbor sensors called ”1”; it is worth to say that the other charts have not been reported here because they present almost the same shape of sensor ”1”.

To understand the behavior of our algorithm, in the time of the observation, some artificial drift has been introduced. As a result, the curve of sensor ”1” , Figure 8, presents some high peaks that simulate a time without light. Then, the sensor ”4” receives the observations from the neighbours and ”follows” the change by auto-calibrating its status, as the curve in Figure 7 shows. Indeed, it is possible to see how the calibrated value, after a fast period of learning, follows the trend of the other sensors, when the values change.

The fast calibration proves that the framework works, as explained in the Section III: the calibrating sensor, through the calibration framework, understand automatically that its values is diverging from the other ones and it is able to correct it.

This is the behavior if all the neighbor sensors change their values. Instead, if only one sensor changes (because it is

drifted), this issue doesn't affect the behavior of the calibration framework. Indeed, if a calibrating sensor receives n correct values and only one incorrect value from neighbour sensor, it is able to filter that one and to calculate a correct expected value.

There is a limit case in which the system doesn't work correctly: if all the neighbour sensors drift in the same way, sending all the same incorrect value. In this case, the calibration framework will consider that value as the correct one, with the consequence to calibrate the sensor in a wrong way. But this is really a rare case.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a solution for incremental learning and error correction in an auto-calibration model. Our solution aims to practically improve the maintenance of sensors in a distributed and large scenario, such as the smart city one. The proposed approach presents a classic scenario of auto-calibration that is able to overcome to several problems that corrupt the good work of a sensor. We have used an Incremental SVR to learn and model the observed phenomenon and the well-know Kalman filter to correct the model and stabilize the status of the sensor.

Perturbations and malfunctioning can in fact corrupt the reliability of the sensor's measurements and in a large deployment such as a smart city, identify and correct such errors is very time consuming. Then, our approach offers the possibility to deploy a sensor in a area without the need of training the algorithm, because it doesn't require the use of a long history of data. Indeed, in general, the sensor will learn the spatio-temporal model of the observed phenomenon by using the observations of the neighbors.

However, in this work we have assumed that sensors outside the neighborhood can not effect the model and the definition of the proximity between sensors has not been defined yet. In future works, the scenario will be extended in such a way that sets of neighbor sensors can in fact affect each other and evaluated on a large scale environment.

To avoid that malicious nodes/attackers provide bad values, the communication between the sensors are protected with encryption and authorization protocols, such as SSL and TLS [31]. Furthermore, in next versions of the solution, the authors will evaluate the possibility to increase the security level, leveraging the feature provided by the XMPP protocol [32].

Furthermore, in the current version, the structure of the WSN framework is established a-priori. Before deploying them, it is established the position of the sensors (setting the neighbour sensors). In the next versions, it will be possible to use a geo-tagging module mounted on the sensor board, to locate the sensors; as a result using different MQTT topics connected with the position, it will be possible to interconnect automatically the neighbour sensors and manage also automatically their movement; when a sensor is moved, it can unsubscribe itself from the previous topic and subscribe to the new one, changing the interconnected sensors.

ACKNOWLEDGMENT

This work has been made possible thanks to the support and funding of Smart City Strategic Programme[5] of Istituto Superiore Mario Boella (ISMB).

REFERENCES

[1] Estrin, D.; Girod, L.; Pottie, G.; Srivastava, M., "Instrumenting the world with wireless sensor networks," Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on , vol.4, no., pp.2033,2036 vol.4, 2001.

[2] Takruri, M.; Challa, S., "Drift aware wireless sensor networks," Information Fusion, 2007 10th International Conference on , vol., no., pp.1,7, 9-12 July 2007.

[3] Vladimir Bychkovskiy, Seapahn Megerian, Deborah Estrin, and Miodrag Potkonjak. 2003. A collaborative approach to in-place sensor calibration. In Proceedings of the 2nd international conference on Information processing in sensor networks (IPSN'03), Feng Zhao and Leonidas Guibas (Eds.). Springer-Verlag, Berlin, Heidelberg, 301-316.

[4] Balzano, L.; Nowak, R., "Blind Calibration of Sensor Networks," Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on , vol., no., pp.79,88, 25-27 April 2007.

[5] Hoadley, Bruce. "A Bayesian look at inverse linear regression." Journal of the American Statistical Association 65.329 (1970): 356-369.

[6] Bychkovskiy, Vladimir, et al. "A collaborative approach to in-place sensor calibration." Information Processing in Sensor Networks. Springer Berlin Heidelberg, 2003.

[7] R. Tan, G. Xing, Z. Yuan, X. Liu, and J. Yao, System-level calibration for data fusion in wireless sensor networks, ACM TOSN, vol. 9, no. 3, pp. 28:128:27, 2013.

[8] J. Feng, S. Megerian, and M. Potkonjak, Model-based calibration for sensor networks, in Proc. IEEE Sensors, vol. 2. Oct. 2003, pp. 737742.

[9] N. Ramanathan, L. Balzano, M. Burt, D. Estrin, T. Harmon, C. Harvey, et al., Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks, Ph.D. dissertation, Dept. Civil Environ. Eng., Univ. California, Los Angeles, Los Angeles, CA, USA,2006.

[10] E. Miluzzo, N. D. Lane, A. T. Campbell, and R. Olfati-Saber, CaliBee: A self-calibration system for mobile sensor networks, in Proc. IEEE DCOSS, Jun. 2008, pp. 314331.

[11] Ramanathan, Nithya, et al. "Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks." Center for Embedded Network Sensing (2006).

[12] Taylor, Christopher, et al. "Simultaneous localization, calibration, and tracking in an ad hoc sensor network." Proceedings of the 5th international conference on Information processing in sensor networks. ACM, 2006.

[13] Takruri, Maen, Khalid Aboura, and Subhash Challa. "Distributed recursive algorithm for auto calibration in drift aware wireless sensor networks." Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering. Springer Netherlands, 2008. 21-25.

[14] Byung-Tak Lee; Seung-Chul Son; Kyungran Kang, "A Blind Calibration Scheme Exploiting Mutual Calibration Relationships for a Dense Mobile Sensor Network," Sensors Journal, IEEE , vol.14, no.5, pp.1518,1526, May 2014.

[15] Feng, Jessica, Seapahn Megerian, and Miodrag Potkonjak. "Model-based calibration for sensor networks." Sensors, 2003. Proceedings of IEEE. Vol. 2. IEEE, 2003.

[16] Balzano, Laura. "Addressing fault and calibration in wireless sensor networks." University of California, Los Angeles (2007).

[17] Takruri, M.; Rajasegarar, S.; Challa, S.; Leckie, C.; Palaniswami, M., "Spatio-temporal modelling-based drift-aware wireless sensor networks," Wireless Sensor Systems, IET , vol.1, no.2, pp.110,122, June 2011.

[18] C. Wang, P. Ramanathan, and K. K. Saluja, Blindly calibrating mobile sensors using piecewise linear functions, in Proc. IEEE SECON, Jun. 2009, pp. 19.

[19] M. Takruri, S. Challa, and R. Yunis, Data fusion techniques for auto calibration in wireless sensor networks, in Proc. IEEE Inf. Fusion, Jul. 2009, pp. 132139.

[5] http://www.ismb.it/en/smart_city/

[20] C. Wang, P. Ramanathan, and K. K. Saluja, Blindly calibrating mobile sensors using piecewise linear functions, in Proc. IEEE SECON, Jun. 2009, pp. 19.
[21] R. E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME D, J. Basic Eng., Vol. 82, 35-45, 1960.
[22] G. E. D'Errico, N. Murru, An Algorithm for Concurrent Estimation of Time Varying Quantities, Meas. Sci. Technol., Vol. 23, Article ID 045008, 9 pages, 2012.
[23] Dong Hui, Fu Helin and Wumiug Leng, "Support Vector Machines For Time Series Regression and Prediction", Journal of System Simulation, vol. 18, no. 7, pp. 1785-1788, 2006
[24] V. Vapnik, The Nature of Statistical Learning Theory, 1999, Springer-Verlag.
[25] Smola, A.J., Scholkopf, B.: A tutorial on support vector regression, Stat. Comput., 2004, 14, (3), pp. 199222
[26] Clarke, S.M., Griebsch, J.H., Simpson, T.W.: Analysis of support vector regression for approximation of complex engineering analyses, J. Mech. Des., 2005, 127, (6), pp. 10771087
[27] M. Takruri, S. Rajasegarar, S. Challa, C. Leckie and M. Palaniswami, "Spatio-temporal modelling-based drift-aware wireless sensor networks," in IET Wireless Sensor Systems, vol. 1, no. 2, pp. 110-122, June 2011.
[28] Marc G. Genton, "Classes of Kernels for Machine Learning: A Statistics Perspective", Journal of Machine Learning Research 2 (2001) 299-312
[29] H. Xu, R. Wang and K. Wang, "A New SVR Incremental Algorithm Based on Boundary Vector," Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on, Wuhan, 2010, pp. 1-4.
[30] Nadir Murru, Rosaria Rossini. A Bayesian approach for initialization of weights in backpropagation neural net with application to character recognition. Neurocomputing, Volume 193, 12 June 2016, Pages 92-105.
[31] MQTT Specification (online) http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html
[32] Conzon, D., T. Bolognesi, P. Brizzi, A. Lotito, R. Tomasi, and M. A. Spirito, "The VIRTUS Middleware: An XMPP Based Architecture for Secure IoT Communications", 21st