

## Auto Calibration in Drift Aware Wireless Sensor Networks using The Interacting Multiple Model Algorithm

Maen Takruri<sup>1</sup>, Subhash Challa<sup>2</sup>, Rajib Chakravorty<sup>2</sup>

<sup>1</sup>CRIN, University of Technology, Sydney, Australia

<sup>2</sup>NICTA VRL, University of Melbourne, Victoria 3010, Australia

Email: mtakruri@eng.uts.edu.au, {subhash.challa, rajib.chakravorty}@nicta.com.au

**Abstract**—The purpose for wireless sensor networks is to deploy low cost sensors with sufficient computing and communication capabilities to support networked sensing applications. The emphasis on lower cost led to sensors that are less accurate and less reliable than their wired sensor counterparts. Sensors usually suffer from both random and systematic bias problems. Even when the sensors are properly calibrated at the time of their deployment, they develop drift in their readings leading to biased sensor measurements. The drift in this context is defined as a unidirectional long-term change in the sensor measurement. Assuming that neighboring sensors have correlated measurements and noting that the instantiation of drift in a sensor is uncorrelated with other sensors, we present the methodology for detecting and correcting sensors smooth and steep drifts. The methodology improves the reliability and the effective life of the network.

### I. INTRODUCTION

Recently, wireless sensor networks (WSN) have emerged as an important research area [1]. This development has been encouraged by the dramatic advances in sensor technology, wireless communications, digital electronics and computer networks, enabling the development of low cost, low power, multi-functional sensor nodes that are small in size and can communicate at short distances [2]. When they work as a group, they can accomplish far more complex tasks and inferences than individual super nodes. This led to a wide spectrum of possible military and civilian applications, such as battlefield surveillance, home automation, smart environments and forest fire detection.

On the down side, these wireless sensors are usually left unattended for long periods of time in the field, which makes them prone to failures. This is due to either sensors running out

of energy or harsh environmental conditions surrounding them. These cheap sensors also tend to develop drift as they age. This poses a major problem for the end application, as the data from the network becomes progressively useless. An early detection of such drift is essential for the successful operation of the sensor network.

The sensor error problems and their effects on sensor inferences have not been addressed thoroughly in the literature. We address this problem using the fact that neighboring sensors in a network observe correlated data, i.e., the measurements of one sensor are related to the measurements of its neighbors. Furthermore, the physical phenomenon that these sensors observe also follows some spatial correlation. Hence, in principle, it is possible to predict the data of one sensor using the data from other closely situated sensors [3], [4]. This predicted data provides a suitable basis to correct anomalies in a sensor's reported information. The early detection of anomalous data enables us not only to detect drift in sensor readings, but also to correct it.

Another common problem faced in large scale sensor networks is that sensors can suffer from bias or systematic errors. These errors have a direct impact on the effectiveness of the associated decision support systems. Calibrating these sensors to account for these errors is a costly and time consuming process. Traditionally such errors are corrected by site visits where an accurate, calibrated sensor is used to calibrate other sensors. This process is manually intensive and is only effective when the number of sensors deployed is small and the calibration is infrequent. In a large scale sensor network, constituted of cheap sensors, there is a need for frequent recalibration. Due to the size of such networks, it is impractic-

cal and cost prohibitive to manually calibrate them. Hence, there is a significant need for auto-calibration [4] in sensor networks.

A straightforward approach to calibration is to apply a known stimulus to the sensor network and measure the response [5]. Then comparing the ground truth input to the response will result in finding the gain and offset for the linear drifts case [6]. The calibration problem of the sensor network was also tackled by [5] in a different way. They stated that after sensors are calibrated to the factory settings when deployed, their measurements will differ linearly from the ground truth by certain gains and offsets for each sensor. They presented an intelligent method to estimate these gains and offsets without the need of ground truth measurements for comparison and referred to this problem as blind calibration of sensor networks. So by identifying these gains and offsets (which they assumed to be constant for each sensor), the future readings of the sensors can be calibrated to the true values. The method worked well in a controlled environment but not with noise and other disturbances.

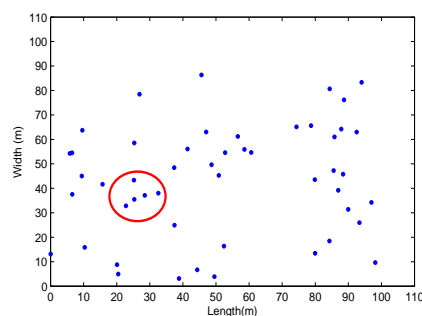
The idea of drift aware wireless sensor networks was first introduced by Maen et al. [4]. They showed that detecting drifting sensors and correcting their measurements would increase the effective life of the network. In [7], they introduced a formal statistical procedure for tracking and detecting sensors drifts using Kalman filters. The sensors of the network were close enough to have similar temperature readings and the average of their measurements was taken as a sensible estimation to be used by each sensor to self-assess. In this paper we extend the work in [7] to able to deal with drifts with sudden jumps. No assumptions regarding the linearity of the drifts are made as in [5].

The rest of the paper is organised as follows. We present our network structure and the problem statement in Section II. Sections III formulates our IMM framework for drift correction in sensor networks. The evaluation of the proposed algorithm is given in section IV. Section V concludes with future work.

## II. NETWORK STRUCTURE AND PROBLEM STATEMENT

Consider a wireless sensor network with a large number of sensors distributed randomly

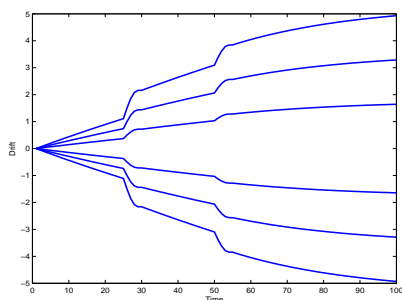
in a certain area of deployment such as the one shown in Figure 1. The sensors are grouped in clusters (sub-networks) according to their spatial proximity. Each sensor measures a phenomenon such as ambient temperature, chemical concentration, noise or atmospheric pressure. The measurement, say temperature, is considered to be a function of time. An example of a cluster is shown using a circle in Figure 1. The sensors within the cluster are considered to be capable of communicating their readings among themselves.



**Figure 1: A sensor area with encircled sub-network**

As time progresses, some nodes will start experiencing drift in their readings. If these readings are collected as such at these nodes, it would cause the network to accept erroneous conclusions. After some level of unreliability, the network inferences become un-trusted. At this point, the network becomes useless as it is impractical and infeasible to manually recalibrate the sensors. In order to mitigate the drift problem, each sensor node in the network has to detect and correct its own drift using the feedback obtained from its neighbour nodes. This is based on the fact that the data from all the nodes within a cluster are correlated and the faults or drifts instantiations are likely to be uncorrelated. The ability of the sensor nodes to auto-detect and correct their drifts helps to extend the effective lifetime of the network. In addition to the drift problem, we also consider the inherent bias that may exist within some sensor nodes. There exists a distinct difference between these two errors. The former changes with time and often becomes accentuated, while the latter, is considered to be a constant error from the beginning of the operation. This error is usually due to a possible manufacturing defect or a faulty calibration.

The sensor drift we consider in this work has sudden jumps or changes. We model the drift as a linear or exponential with sudden changes, surges or sharp peaks. Besides, it is dependent on the environmental conditions, and strongly related to the manufacturing process of the sensor. This is what makes the instantiation of drift different from one sensor to another. It is highly unlikely that two electronic components fail in a correlated manner unless they are from the same integrated circuit (IC). Figure 2 shows examples of the theoretical drift models that can be captured by the framework we propose.



**Figure 2: Examples of drifts with jumps and sudden changes**

Consider a sensor sub-network that consists of  $n$  sensors deployed randomly in a certain area of interest. Without loss of generality, we choose a sensor network for measuring temperature, even though this is generally applicable to all other types of sensors that suffer from drift and bias problems. Let  $T$  be the groundtruth temperature. In this work  $T$  is considered to vary only with time inside the sub-network or the cluster. Therefore we denote the temperature at a certain time instance and sensor location as  $T_{i,k}$  where  $i$  is the sensor number and  $k$  is the time index. Since the measured temperature inside the cluster is space invariant, we denote the measured temperature as  $T_k$  where  $T_{i,k}=T_{j,k}=T_k$ . At each time instant  $k$ , node  $i$  in the sub-network measures a reading  $r_{i,k}$  of  $T_k$ . It then reports a *drift corrected* value  $x_{i,k}$  to its neighbors. The corrected value  $x_{i,k}$  should ideally be equal to the ground truth temperature  $T_k$ . If all nodes are perfect,  $r_{i,k}$  will be equal to the  $T_k$ , and the reported values will ideally be equal to the readings, i.e.,  $x_{i,k} = r_{i,k}$ .

During this process, each node  $i$  finds a

predicted value  $\hat{x}_{i,k}$  as a function of corrected measurements collected from its neighbour sensors using  $\hat{x}_{i,k} = f(\text{neighbourdata})$ . Similar to our previous work in [7],  $\hat{x}_{i,k}$  is taken to be equal to the average of the neighbour sensors' reported values  $\hat{x}_{i,k} = \bar{x}_k = \sum_{i=1}^n x_{i,k}/n$ . In an ideal situation,  $\hat{x}_{i,k} = T_k$ . In practice, each sensor reading comes with an associated reading error, and drift  $d_{i,k}$ . This drift may be null or insignificant during the initial period of deployment, depending on the nature of the sensor and the deployment environment. The problem we address here is how to account for the drift in each sensor node  $i$ , using the predicted value  $\hat{x}_{i,k}$ , which is obtained using information gathered from neighbouring nodes, so that the reading  $r_{i,k}$  is corrected and reported as  $x_{i,k}$ .

In this paper, we extend the work in [7] to make the algorithm proposed there capable of dealing with smooth and non-smooth (has jumps) drifts by using the interacting multiple model algorithm (IMM). In the following section we describe our drift detection and correction formulation in detail.

### III. ITERATIVE DRIFT ESTIMATION AND CORRECTION USING IMM

In this section we introduce a probabilistic model that captures drifts that have surges and sudden escalations. This Model is called IMM and it is originally used in target tracking to track manoeuvring objects that show sudden changes in their dynamics [8], [9] and [10]. We apply the IMM algorithm in a decentralized manner as we did for smooth drift tracking model in [7]. For each sensor an  $M$  number of modes will simulate the possible drift jumps. Our solution to the sudden step drift (and it also works for smooth drift) problem consists of the following iterative steps. As for the case of smooth drift, at stage  $k$ , a reading  $r_{i,k}$  is made by node  $i$ . Rather than sending that value automatically to the its neighbours, the node is aware of its drift, and has an estimate for it at this stage. It is a projected value from an estimate of the drift made at the previous time step. Using this estimate of the drift, the node sends the corrected sensor reading  $x_{i,k}$  to its neighbours. Each sensor computes the average  $\bar{x}_k = \sum_{i=1}^n x_{i,k}/n$  to self-assess its measurements. To estimate the drift of a node, the mathematical model of equation (1)

is used. Assuming sudden jumps in the way the drift changes, our drift transition equation becomes:

$$d_{i,k}^\theta = d_{i,k-1}^\theta + u_{i,k}^\theta + v_{i,k} v_{i,k} \sim N(0, Q_k) \quad (1)$$

where  $\theta = 1, 2, \dots, M$ ,  $u_{i,k}^\theta$  is the input or jump corresponding to  $\theta_{th}$  model at time  $k$  for the  $i_{th}$  sensor and  $v_{i,k}$  is the process noise.  $v_{i,k}$  is taken to be Gaussian with zero mean and  $Q_k$  as covariance.

Equation (1) represents an  $M$  number of possible drift models for each node. Each model differs from the others in the size of the jumps  $u_{i,k}^\theta$ . The resultant estimated drift for node  $i$  at time instant  $k$  ( $\hat{d}_{i,k}$ ) would be a weighted combination of the estimated drift of each model  $\hat{d}_{i,k}^\theta$ . The resultant estimated drift for the node ( $\hat{d}_{i,k}$ ) is found as will be shown later in this section by  $\hat{d}_{i,k} = \sum_{\theta=1}^M \mu_{i,k}^\theta \hat{d}_{i,k}^\theta$  where  $\mu_{i,k}^\theta$  is the probability that the estimated drift  $\hat{d}_{i,k}$  follows the drift model  $\hat{d}_{i,k}^\theta$  given the measured values until the time step  $k$ .

A source of information is needed to provide input to a statistical model such as equation (1). In target tracking, a measurement equation is established to model observations made over time. These measurements provide the statistical information needed for the estimation procedure. In this case, the preferred source of information would be the real value of  $T$ , the quantity being sensed. For example, if  $T_k$  was available after the reading  $r_{i,k}$  is made by node  $i$ , then the drift would be assessed exactly, it being  $r_{i,k} - T_k$ . However, the whole purpose of the sub-network is to assess  $T$ , and so  $T_k$  will never be known. Only an estimate of it is available, and in this case, it is the average of the collected sensor reports,  $\bar{x}_k$ . Using this average as a sensible estimation, an approach used by [3], measurements  $\{y_{i,k}\}$  are obtained, where

$$y_{i,k} = r_{i,k} - \bar{x}_k \quad (2)$$

This is based on the assumption that not all sensors will start drifting together, and that it is more likely that one sensor at a time will start drifting. This, with the addition that the nodes are self-correcting, makes  $\bar{x}_k$  a good statistic for  $T_k$ . Equation (2) is derived from the relationship

$$r_{i,k} = T_k + d_{i,k}$$

Often, there is an error associated with the reading of a node,  $w_{i,k} \sim N(0, R_k)$ , leading to  $r_{i,k} = T_k + d_{i,k} + w_{i,k}$ . so the measurement equation becomes:

$$y_{i,k} = d_{i,k}^\theta + w_{i,k} \quad w_{i,k} \sim N(0, R_k) \quad (3)$$

where  $w_{i,k}$  is assumed to be a Gaussian noise with zero mean and  $R_k$  as the covariance. Referring to equations (1) and (3) we notice that they represent an  $M$  number of kalman filter equations corresponding to  $M$  number of drift models (jumps). This leads according to the IMM algorithm to  $M$  number of kalman filters working in parallel to result in  $M$  number of estimations for drift and covariance. Each model has a probability ( $\mu_{i,k}^\theta = p(\text{model}_{i,k} = \theta | y_i^k)$ ) depending on the measured values until that time step. Switching between models is governed by a pre-defined Markov transition matrix  $\Gamma$  of dimension  $M \times M$  for  $M$  models.

$$\Gamma = \begin{bmatrix} \gamma_{11} & \dots & \gamma_{1M} \\ \vdots & \dots & \vdots \\ \gamma_{M1} & \dots & \gamma_{MM} \end{bmatrix} \quad (4)$$

where  $\gamma_{\theta\alpha} = p(\text{model}_{i,k} = \theta | \text{model}_{i,k-1} = \alpha)$  which is the probability of switching from model  $\alpha$  to model  $\theta$  in one time step.

The IMM step of our drift tracking algorithm is explained as follows: At time step ( $k$ ) each node is supposed to know the previous time step ( $k-1$ ) models probabilities ( $\mu_{i,k-1}^\theta$ ), estimated drifts ( $d_{i,k-1}^\theta$ ) and associated covariances ( $P_{i,k-1}^\theta$ ). Unlike the standard Kalman Filter tracking algorithm, the previous estimates are not used as priors for the  $M$  Kalman Filters. Instead, they are used in the mixing step to calculate the IMM prior models probabilities ( $\bar{\mu}_{i,k-1}^\theta$ ), prior drift estimates ( $\bar{d}_{i,k-1}^\theta$ ) and associated prior covariances ( $\bar{P}_{i,k-1}^\theta$ ) that are then fed to the corresponding  $M$  filters to result in the posterior models estimates ( $\hat{d}_{i,k}^\theta, \hat{P}_{i,k}^\theta$ ).

The output the IMM algorithm is then found by first updating the models probabilities ( $\mu_{i,k}^\theta$ ), which are used then together with the outputs of the  $M$  kalman filters to find  $\hat{d}_{i,k}$  and  $P_{i,k}$ . The algorithm then reiterates taking the predicted drift at time step ( $k+1$ ) to be equal to the estimated drift at time step ( $k$ )  $\hat{d}_{i,k+1} = \hat{d}_{i,k}$ . The steps of our sudden drift tracking algorithm are stated below:

### Distributed Drift Correction Algorithm

For each node  $i$

- At step  $k$ , a predicted  $\tilde{d}_{i,k}$  drift is available
- Each node  $i$  obtains its reading  $r_{i,k}$
- The corrected reading is calculated,  $x_{i,k} = r_{i,k} - \tilde{d}_{i,k}$  and then transmitted to the neighboring nodes.
- Each node computes the average  $\hat{x}_{i,k}$ .
- The measurement  $y_{i,k} = r_{i,k} - \hat{x}_{i,k}$  is obtained.
- Mixing stage

$$\begin{aligned}\bar{\mu}_{i,k}^{\theta} &= \sum_{\alpha=1}^M \gamma_{\theta\alpha} \mu_{i,k-1}^{\alpha} \\ \bar{d}_{i,k-1}^{\theta} &= \sum_{\alpha=1}^M \frac{\gamma_{\theta\alpha} \mu_{i,k-1}^{\alpha}}{\bar{\mu}_{i,k}^{\theta}} \hat{d}_{i,k-1}^{\alpha} \\ \bar{P}_{i,k-1}^{\theta} &= \sum_{\alpha=1}^M \frac{\gamma_{\theta\alpha} \mu_{i,k-1}^{\alpha}}{\bar{\mu}_{i,k}^{\theta}} \times \\ &\quad (P_{i,k-1}^{\alpha} + \{\hat{d}_{i,k-1}^{\alpha} - \bar{d}_{i,k-1}^{\theta}\}^2)\end{aligned}$$

- Kalman Filter update stage

$$\begin{aligned}\hat{d}_{i,k}^{\theta} &= \bar{d}_{i,k-1}^{\theta} + u_{i,k}^{\theta} + \\ &\quad \frac{\bar{P}_{i,k-1}^{\theta} + Q_k}{\bar{P}_{i,k-1}^{\theta} + Q_k + R_k} (y_{i,k} - \bar{d}_{i,k-1}^{\theta}) \\ P_{i,k}^{\theta} &= (\bar{P}_{i,k-1}^{\theta} + Q_k) \times \\ &\quad \left(1 - \frac{\bar{P}_{i,k-1}^{\theta} + Q_k}{\bar{P}_{i,k-1}^{\theta} + Q_k + R_k}\right)\end{aligned}$$

- IMM output stage

Model probability is updated as follows:

$$\mu_{i,k}^{\theta} = \frac{1}{C} \cdot \bar{\mu}_{i,k}^{\theta} \cdot \frac{1}{\sqrt{A}} e^{-\frac{(y_{i,k} - \bar{d}_{i,k-1}^{\theta} - u_{i,k}^{\theta})^2}{2A}}$$

where  $A = \bar{P}_{i,k-1}^{\theta} + Q_k + R_k$  and  $C = \sum_{\theta=1}^M \mu_{i,k}^{\theta}$  and then the drift and its associated covariance are updated as follows:

$$\begin{aligned}\hat{d}_{i,k} &= \sum_{\alpha=1}^M \mu_{i,k}^{\alpha} \hat{d}_{i,k}^{\alpha} \\ P_{i,k} &= \sum_{\alpha=1}^M \mu_{i,k}^{\alpha} \hat{d}_{i,k}^{\alpha} (P_{i,k}^{\alpha} + \{\hat{d}_{i,k}^{\alpha} - \hat{d}_{i,k}\}^2)\end{aligned}$$

- The projected drift  $\tilde{d}_{i,k+1} = \hat{d}_{i,k}$  is obtained and the algorithm reiterates.

### IV. EVALUATION

Our aim is to evaluate the ability of our proposed framework to correct the drift experienced in a sensor node using the information gathered from the nearest neighbouring nodes. We simulate a small sub-network of 10 nodes measuring the temperature in a certain area. We assume that 2 sensors are developing smooth drifts of the forms mentioned above. We compare our IMM drift tracking algorithm with the plain KF drift tracking algorithm given in [7]. Figures 3 and 4 show the KF drift tracking algorithm results obtained in [7]. The results of the IMM drift tracking algorithm are shown in figures 5 and 6. It is clear from figures 6 and 4 that the IMM drift algorithm follows the drift with jumps instantly with minimal errors and more efficiently than the plain KF drift tracking algorithm. Hence, the IMM drift tracking algorithm outperforms the KF tracking algorithm in terms of speed and accuracy of tracking the drift.

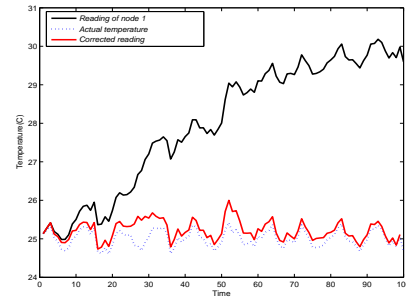


Figure 3: Tracking drift and correcting reading of node 1 for KF

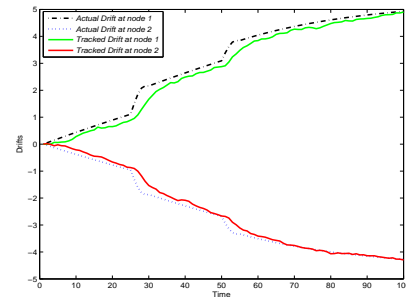
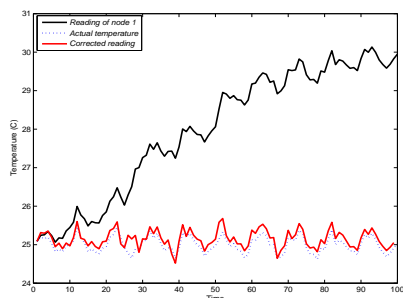
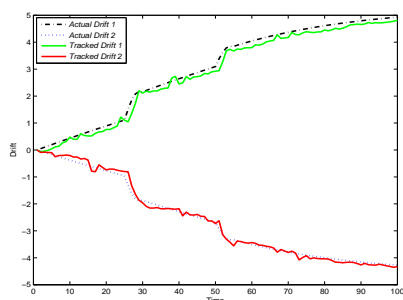


Figure 4: Actual and estimated drifts in nodes 1 and 2 for KF

Looking at figures 5 and 3, It is clear that both the IMM and the KF drift tracking algorithms extend the effective operational life



**Figure 5: Tracking drift and correcting reading of node 1 for IMM**



**Figure 6: Actual and estimated drifts in nodes 1 and 2 for IMM**

time for node 1. If we assume that for our application that the maximum tolerable temperature error in node's 1 reading is  $1 C^o$ , then the life of node 1 is extended from 20 time units when there is no drift correction (Reading of node 1 curve) to at least 100 time units when the IMM or the KF algorithm is applied (Corrected reading curve). This applies to all of the network's sensors that develop drift. Hence, the life of the network will be extended by applying the drift tracking and correction algorithms. It is also worth noting from figures 5 and 3 that the difference between the actual and corrected reading curves tends to be smaller for the IMM algorithm results. This indicates that the error accumulation in the case of IMM is less and so it is expected to give longer life for the network.

We conducted several simulation scenarios and observed that the method worked as long as not all sensors start drifting at the same instant of time. We went as far as 70% of the sensors drifting, and the corrections still made the network system data validity hold. It can be higher than this, but it depends on the error amplitudes.

## V. CONCLUSION AND FUTURE RESEARCH

In this paper we have proposed a formal statistical procedure for estimating sensor errors in a WSN based on the assumption that neighboring sensors have correlated measurements and that the instantiation of drift in a sensor is uncorrelated with other sensors. The solution is computationally simple allowing its implementation in a WSN. We introduced a probabilistic procedure that captures sudden jumps, changes and escalations in the drift.

In future, we will also consider the solution to problems with drifts that are correlated to sensors value. We will also address the problem when temperature in the sub-network varies with distance and time. We aim by introducing this solution to initiate a methodical research in the drift of WSN nodes.

## VI. ACKNOWLEDGMENT

We acknowledge the support of THALES, Australia for this work through an ARC Linkage grant (LP0561200)/APAI Scholarship.

## REFERENCES

- [1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," *Intl. Conference on Acoustics, Speech, and Signal Processing (ICASSP2001)*, May 2001.
- [2] I. F. Akyildiz, W. Su, Y. Sankarasubramanian, and E. Cayirci, "Wireless sensor networks: a survey," *Comp. Networks*, vol. 38, pp. 393–422, 2002.
- [3] B. Krishnamachari and S. Iyengar, "Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Trans. Computers*, vol. 53, no. 3, pp. 241–250, 2004.
- [4] M. Takruri and S. Challa, "Drift aware wireless sensor networks," *Proc. of the 10th intl. conference on information fusion (Fusion 2007)*, July 2007.
- [5] L. Balzano and R. Nowak, "Blind calibration of sensor networks," *Proceedings of Information Processing in Sensor Networks*, April 2007.
- [6] B. Hoadley, "A bayesian look at inverse linear regression," *Journal of the American Statistical Association*, vol. 65, pp. 356–369, March 1970.
- [7] M. Takruri, K. Aboura, and S. Challa, "Distributed recursive algorithm for auto calibration in drift aware wireless sensor networks," *Proc. of the Intl. Joint Conferences on Computer, Information, and Systems Sciences, and Engineering*, 2007.
- [8] Y. Bar-Shalom, *Estimation and Tracking : Principle and Software*. Boston Artech House, 1993.
- [9] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.
- [10] S. Challa, R. Evans, M. Morelande, and D. Musicki, *Fundamentals of Object Tracking*. Cambridge University Press, in print 2008.