Recursive Bayesian Approaches for Auto Calibration in Drift Aware Wireless Sensor Networks

Maen Takruri[1], Subhash Challa[2], Rajib Chakravorty[2]
[1]Centre for Real-Time Information Networks (CRIN)
University of Technology, Sydney, Australia
[2]NICTA Victoria Research Laboratory
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. Even when the sensors are properly calibrated at the time of their deployment, they develop drift in their readings leading to biased sensor measurements. Noting that a physical phenomenon in a certain area follows some spatio-temporal correlation, we assume that the sensors readings in that area are correlated. We also assume that the instantiations of drifts are uncorrelated. Based on these assumptions, and inspired by the resemblance of registration problem in radar target tracking with the bias error problem in wireless sensor networks, we follow a Bayesian framework to solve the Drift/Bias problem in wireless sensor networks. We present two methods for solving the drift problem in a densely deployed sensor network, one for smooth drifts and the other for unsmooth drifts. We also show that both methods successfully detect and correct sensor errors and extend the effective life time of the sensor network.**

***Index Terms*—Wireless Sensor Networks, sensor calibration, drift and bias, error detection and correction.**

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.

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 either due to running out of energy or the harsh environmental conditions surrounding them in the deployment area. Sensor nodes also tend to develop drift in their measurements as they age. The drift we consider in this context is unidirectional long-term change in the sensor measurement. In addition to drift, sensor nodes suffer from bias in their measurements [3]. This poses a major problem for the end application, as the data from the network becomes progressively useless.

Traditionally such errors are accounted for by calibrating the erroneous sensors against accurately calibrated standard 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, frequent manual calibration is impractical and cost prohibitive. Hence, there is a significant need for auto-calibration [4] in sensor networks.

In this paper, we address the sensor measurement drift/bias problem using the fact that neighbouring sensors in a network, observe correlated data, i.e., the measurements of one sensor are related to the measurements of its neighbours. 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 [5], [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.

The sensor bias and drift problems and their effects on sensor inferences have not been addressed thoroughly in the sensor networks literature. On the other extreme, the bias correction problem has been well studied in the context of multi-radar tracking problem. In target tacking literature the problem is usually referred to as the *Registration problem* [6], [7]. When the same target is observed by two sensors (radars) from two different angles, the data from those two sensors can be fused to estimate the bias in both sensors. In the context of image processing of moving objects, the problem is referred to as *Image Registration*, which is the process of overlaying two or more images of the same scene taken at different times, from different viewpoints, and/or by different cameras. It geometrically aligns two images, the reference and sensed images [8]. Image registration is a crucial step in all image analysis tasks in which the final information is gained from the combination of various data sources like in image fusion [9].

A straightforward approach to calibration is to apply a known stimulus to the sensor network and measure the response [10]. Then comparing the ground truth input to the response will result in finding the gain and offset for the linear drifts case [11]. The calibration problem of the sensor network was also tackled by [10] 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 a method for estimating these gains and offsets using subspace matching. The method only required routine measurements to be collected by the sensors and did not require ground truth measurements for comparison. The estimated gains and offsets (which they assumed to be constant for each sensor) were used for calibrating the future sensor readings to the true values. The method worked well in a controlled environment but not with noise and other disturbances.

An earlier work on blind calibration of sensor nodes in a sensor network was presented in [3]. They assumed that the sensors of the network under consideration were densely deployed that they observed the same phenomenon. They used the temporal correlation of signals received by neighbouring sensors when the signals were highly correlated to derive a function relating the bias in their amplitudes. Another method for calibration was considered by [12]. They used geometrical and physical constraints on the behaviour of a point light source to calibrate light sensors without the need of comparing the measurement with an accurate sensor (ground truth). They assumed that light sensors under consideration suffered form a constant bias with time. The author in [13] described a method for in-situ blind calibration of moisture sensors in a sensor network. She used the Ensemble Kalman Filter to correct the values measured by the sensors or in other words to estimate the true moisture at each sensor. The state equation was governed by a physical model of moisture used in environmental and civil engineering and the measurement was assumed to be related to the real state by a certain offset and gain. The state (moisture) vector was augmented with the calibration parameters (gain and offset) and then the gains and offsets were estimated to recover the correct state from the measurements.

The idea of drift aware wireless sensor networks was first introduced by Maen et al. [4]. We showed there that detecting drifting sensors and correcting their measurements would increase the effective life of the network. In [14], we introduced a formal statistical procedure for tracking and detecting smooth sensors drifts using Kalman filters. We also introduced in [15], an algorithm for tracking and detecting unsmooth sensors drifts using the Interacting Multiple Model algorithm (IMM). 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. No assumptions regarding the linearity of the drifts were made as in [10]. In this paper we elaborate on the work of [15] by providing more illustrations and evaluations for the IMM based unsmooth drift detection and correction algorithm. We also give a formal derivation of the smooth drift detection and correction algorithm which was introduced in [14] using Bayesian reasoning. In addition to that, we show that both algorithms not only detect and correct drifts, but also they detect and correct sensors biases. A comparison between the algorithms is made to show that the IMM based detection and correction algorithm performs better but at the cost of increased computational complexity.

The rest of the paper is organised as follows. We present our network structure and the problem statement in section II. Section III presents a Bayesian approach for solving the smooth drift/bias problem in wireless sensor networks. Section IV formulates our IMM framework as an upgrade of the last approach to solve the unsmooth drift problem. The evaluation of the proposed algorithm is given in section V. Section VI concludes with future work.

II. Network Structure and Problem Statement

Consider a WSN 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 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.

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 non trustworthy. 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 it's 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 (useful) 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

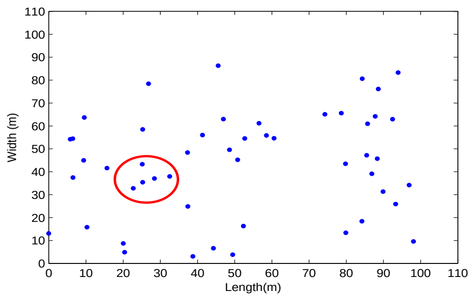

Figure 1. Wireless sensor area with encircled sub-network

beginning of the operation. This error is usually due to a possible manufacturing defect or a faulty calibration.

The sensor drift that we consider in this work is of two types. The first is slow smooth drift that we model as linear and/or exponential function of time. The second type is smooth drift with jumps. It is similar to the first; however, it suffers from sudden changes, surges or sharp peaks. Both of them are 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). Figures 2 and 3 show examples of the theoretical drift models for smooth drift and smooth drift with jumps, respectively.

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 ground truth 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 temperature in the cluster is space invariant, we denote it as T_k where $T_{i,k}$=$T_{j,k}$=T_k. At each time instant k, a 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 neighbours. 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 $\hat{x}_{i,k} = f(neighbourdata)$. Similar to our previous work in [14], $\hat{x}_{i,k}$ is taken to be equal to the average of the neighbour sensors' reported values $\hat{x}_{i,k} = \overline{x}_k = \sum_{i=1}^{n} x_{i,k}/n$. 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 the following section we introduce a Bayesian formulation for the drift problem in wireless sensor network. This will lead us to derivation of our smooth drift correction algorithm using Kalman Filter (KF) given in [14]. We then upgrade that algorithm to become capable of dealing with drift with sudden jumps by utilising the IMM algorithm.

III. Estimation and Correction of Smooth Drifts

In this section we introduce a Bayesian approach to solve the sensor measurement errors problem in WSN, assuming that the drifts are smooth (see figure 2) and that sensor nodes are densely deployed. Under the dense deployment assumption, all the sensors nodes in a cluster are assumed to measure the same value. Therefore, the average of corrected sensor measurements $\overline{x}_k$ is considered as a good estimate for the expectation of the ground truth value $E\{T_k\}$ in the cluster. It is also considered as a good basis for the sensors to self-assess their measurements.

Let us assume that at time instant k, a measurement or a reading $r_{i,k}$ is made by node i. Rather than sending that value to its neighbours, the node is aware of its drift, and has a predicted value $\widetilde{d}_{i,k}$ for it at this time instant. It is taken to be equal to the estimate of the drift made at the previous time instant, $\widetilde{d}_{i,k} = \hat{d}_{i,k-1|k-1}$, as the drift is assumed to be slow. Using this estimate of the drift, the node i computes its corrected measurement $x_{i,k}$ and sends it to its neighbouring nodes. This applies to all the nodes in the neighbourhood. Each node then collects all the neighbourhood sensors corrected measurements

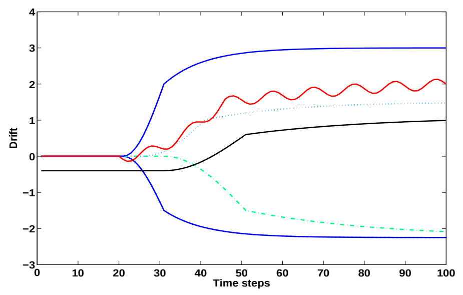

Figure 2. Examples of smooth drifts

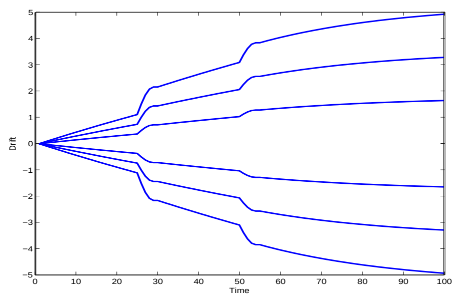

Figure 3. Examples of drifts with jumps and sudden changes

$\{x_{i,k}\}_{i=1}^{n}$, and computes the average $\overline{x}_k = \sum_{i=1}^{n} x_{i,k}/n$. At this point each sensor computes the drift measurement. We define the drift measurement as the difference between the sensor measurement and the average value computed by that sensor. We denote the drift measurement of node i at time instant k by $y_{i,k}$. The drift measurement is used by a KF to estimate the drift. The problem is formulated mathematically as follows:

Assuming that the drift is slow and smooth, it is modelled by:

$$d_{i,k} = d_{i,k-1} + v_{i,k} \quad v_{i,k} \sim N(0, Q_{i,k}) \tag{1}$$

where $d_{i,k}$ is the drift/bias on sensor node i at time instant k, $v_{i,k}$ is the process noise and is taken to be a Gaussian noise with zero mean and variance equal to $Q_{i,k}$.

Since the sensor measurement $r_{i,k}$ usually suffers from random error $w_{i,k}$ and systematic error (drift/bias) $d_{i,k}$, the reading or measurement of sensor i is given by:

$$r_{i,k} = T_k + d_{i,k} + w_{i,k} \quad w_{i,k} \sim N(\mu_{i,k}, R_{i,k})$$

where T_k is the actual (ground truth) value of the measured variable at sensor i and $w_{i,k}$ is the measurement noise and is taken here to be a Gaussian noise with zero mean ($\mu_{i,k} = 0$) and variance $R_{i,k}$.

We also define $x_{i,k}$, the corrected measurement of sensor i at time instant k. $x_{i,k}$ is never sensed but calculated. It is the difference between the sensor reading and the estimated drift and is calculated by $x_{i,k} = r_{i,k} - d_{i,k}$ to result in $x_{i,k} = T_k + w_{i,k}$.

Since the sensors are densely deployed and the instantiations of drifts in the sensors are random, we use the average of corrected sensors' measurements close to node i as an approximate estimate for the expectation of actual (ground truth) value $\overline{x}_k = E\{T_k\} + \frac{1}{n}\sum_{j=1}^{n} w_{j,k}$. We also define $y_{i,k}$ in equation (2) as the difference between the measurement $r_{i,k}$ and the average of corrected sensors measurements $\overline{x}_k$ and refer to $y_{i,k}$ as the drift measurement of node i at time instant k.

$$y_{i,k} = r_{i,k} - \overline{x}_k \tag{2}$$

At early stages of deployment of the sensor network when very few sensors have started to develop drift, and given that the instantiations of drifts in all the sensors are random, we assume that $E\{T_k\} = T_k$. Substituting $r_{i,k}$ into equation (2) results in:

$$\begin{aligned} y_{i,k} &= T_k + d_{i,k} + w_{i,k} - E\{T_k\} - \frac{1}{n}\sum_{j=1}^{n} w_{j,k} \\ &= d_{i,k} + w_{i,k} - \frac{1}{n}\sum_{j=1}^{n} w_{j,k} \\ &= d_{i,k} + \psi_{i,k} \quad \psi_{i,k} \sim N(0, \delta_{i,k}) \end{aligned} \tag{3}$$

where $\psi_{i,k} = w_{i,k} - \frac{1}{n}\sum_{j=1}^{n} w_{j,k}$ is the drift measurement noise and is actually a mixture of Gaussians. It is well known in literature [16], [17] that a Gaussian mixture can be approximated by a Gaussian $\psi_{i,k} \sim N(\pi_{i,k}, \delta_{i,k})$ with the mean found by the weighted sum of the means of the orignal Gaussians:

$$\pi_{i,k} = \mu_{i,k} - \frac{1}{n}\sum_{j=1}^{n} \mu_{j,k} = 0$$

and the variance found by:

$$\begin{aligned} \delta_{i,k} &= [R_{i,k} + (\mu_{i,k} - \pi_{i,k})(\mu_{i,k} - \pi_{i,k})^T] \\ &\quad - \frac{1}{n}\sum_{j=1}^{n}[R_{j,k} + (\mu_{j,k} - \pi_{i,k})(\mu_{j,k} - \pi_{i,k})^T] \\ &= R_{i,k} - \frac{1}{n}\sum_{j=1}^{n} R_{j,k} \end{aligned}$$

Assuming that all sensors are neighbours in the cluster and that they can report to each other, then equation (1) and equation (3) can be written in vector form for all the sensors of the cluster as follows:

$$D_k = FD_{k-1} + V_k \quad V_k \sim N(0, Q_k) \tag{4}$$

$$Y_k = HD_k + \Psi_k \quad \Psi_k \sim N(0, \Delta_k) \tag{5}$$

where Q_k and Δ_k are the process noise and measurement noise covariances, respectively. $D_k = \begin{bmatrix} d_{1,k} & \ldots & d_{i,k} \ldots & d_{n,k} \end{bmatrix}^T$ is the vector of drifts of all sensors in the cluster at time instant k. Similarly, Y_k, V_k, Ψ_k are the vectors of drift measurements, process noise and drift measurement noise of all sensors in the cluster at time instant k, respectively. F and H are the state transition model matrix and the observation model matrix, respectively. Both matrices, H and F, are taken in this scenario to be equal to the identity matrix.

We also define $Y^k = \{Y_1, Y_2 \cdots Y_k\}$ as the set of all drift measurements made up to time k. Accordingly, the problem can be stated as follows: given the set of drift measurements up until the current time k, what is the best estimate of the current drift D_k. Probabilistically, the conditional density relating the state and the measurement vectors is expressed as $p(D_k|Y^k)$ and the estimate would be the expected value $\int D_k p(D_k|Y^k) dD_k$. This estimate is denoted by $\hat{D}_{k|k}$. $\hat{D}_{k|k-1}$ denotes the estimate of D_k given the measurements up until time $k-1$. $p(D_k|Y^k)$ can be expanded by Bayes rule as follows:

$$\begin{aligned} p\,(D_k|Y^k) &= p\,(D_k|Y_k, Y^{k-1}) \\ &= \frac{p\,(Y_k|D_k, Y^{k-1}) \cdot p\,(D_k|Y^{k-1})}{p\,(Y_k|Y^{k-1})} \end{aligned} \tag{6}$$

Assuming the measurement noise is white Gaussian; i.e. not correlated in time, then the current measurements do not depend on the previous measurements and (6) reduces to:

$$p\,(D_k|Y^k) = \frac{\overbrace{p\,(Y_k|D_k)}^{likelihood} \cdot \overbrace{p\,(D_k|Y^{k-1})}^{predicted\ density}}{\underbrace{p\,(Y_k|Y^{k-1})}_{normalisation}} \tag{7}$$

The Likelihood $p(Y_k|D_k)$ for sensor i can be obtained from the measurement equation (5) where Ψ_k is a noise vector, assumed to be Gaussian with zero means and covariance Δ_k. Given D_k, the probability of obtaining a drift measurement vector Y_k should be equal to the probability of the noise with mean HD_k:

$$p\ (Y_k|D_k) = N(HD_k, \Delta_k) \tag{8}$$

The predicted density predicts the current state D_k of the sensors based on the old measurements. We expand it here by Chapman-Kolmogorov identity (an approach used by [18]) as follows:

$$p\ (D_k|Y^{k-1}) = \int p(D_k|D_{k-1}, Y^{k-1})p(D_{k-1}|Y^{k-1})\ dD_{k-1}$$

Assuming the system obeys markov evolution, which implies that its current state directly depends on the previous state, with any dependence on old measurements encapsulated in that previous state, then the transition density can be simplified by neglecting the measurement term as follows:

$$p\ (D_k|Y^{k-1}) = \int p(D_k|D_{k-1})p(D_{k-1}|Y^{k-1})\ dD_{k-1} \tag{9}$$

To evaluate the predicted density $p\ (D_k|Y^{k-1})$ we have to evaluate $p(D_k|D_{k-1})$ and $p(D_{k-1}|Y^{k-1})$ and then substitute them into (9). From (4) and similar to the likelihood:

$$p\ (D_k|D_{k-1}) = N(FD_{k-1}, Q_k) \tag{10}$$

$p(D_{k-1}|Y^{k-1})$ is the prior and is also assumed to be Gaussian with a known mean and covariance from the last iteration:

$$p\ (D_{k-1}|Y^{k-1}) = N(\hat{D}_{k-1|k-1}, P_{k-1|k-1}) \tag{11}$$

Substituting into (9) and evaluating the integral as given in [18] we get:

$$\begin{aligned} p\ (D_k|Y^{k-1}) &= \int N(\hat{D}_{k-1|k-1}, P_{k-1|k-1}) \times \\ &\quad N(FD_{k-1}, Q_k)\ dD_{k-1} \\ &= N(\hat{D}_{k|k-1}, P_{k|k-1}) \end{aligned} \tag{12}$$

where

$$\hat{D}_{k|k-1} = F\hat{D}_{k-1|k-1} \tag{13}$$
$$P_{k|k-1} = FP_{k-1|k-1}\ F^T + Q_k \tag{14}$$

Using total probability lemma and assuming that measurement noise is white Gaussian, the normalisation term can then be expanded as follows:

$$\begin{aligned} p\ (Y_k|Y^{k-1}) &= \int p(Y_k|D_k, Y^{k-1})p(D_k|Y^{k-1})dD_k \\ &= \int p(Y_k|D_k)p(D_k|Y^{k-1})dD_k \end{aligned} \tag{15}$$

Substituting the likelihood and the predicted density in the integral of (15) results in:

$$\begin{aligned} p\ (Y_k|Y^{k-1}) &= \int N(HD_k, \Delta_k)N(\hat{D}_{k|k-1}, P_{k|k-1})dD_k \\ &= N(H\hat{D}_{k|k-1}, S_k) \end{aligned} \tag{16}$$

where

$$S_k = HP_{k|k-1}H^T + \Delta_k \tag{17}$$

Putting all the terms together in (7) and evaluating using an identity given in the appendix of [18] results in:

$$\begin{aligned} p\ (D_k|Y_k) &= \frac{N(HD_k, \Delta_k)N(\hat{D}_{k|k-1}, P_{k|k-1})}{N(H\hat{D}_{k|k-1}, S_k)} \\ &= N(DD_k, P_{k|k}) \end{aligned} \tag{18}$$

where

$$K = P_{k|k-1}H^T(HP_{k|k-1}H^T + \Delta_k)^{-1} \tag{19}$$
$$\hat{D}_{k|k} = \hat{D}_{k|k-1} + K(Y_k - H\hat{D}_{k|k-1}) \tag{20}$$
$$P_{k|k} = (1 - KH)P_{k|k-1} \tag{21}$$

Equations sets (13-14) and (19-21) represent a KF framework [19], [20] for n sensor nodes in the cluster. Since F and H are identity matrices, the system above can be solved as an n-dimensional KF (by a central node and requires high computational capability) or as n 1-dimensional KFs solved by each sensor in the cluster. The first solution is centralised, whereas the latter is decentralised and requires no special processing power by the sensor nodes. We adopt the decentralised solution in this work. F and H are taken to be equal to one. This leads to the probabilistic solution for drift $d_{i,k} \sim N(\hat{d}_{i,k|k}, P_{i,k|k})$ with mean and variance:

$$\hat{d}_{i,k|k} = \hat{d}_{i,k-1|k-1} + K(y_{i,k} - \tilde{d}_{i,k}) \tag{22}$$
$$P_{i,k|k} = (P_{i,k-1|k-1} + Q_{i,k})(1 - K) \tag{23}$$
$$K = \frac{P_{i,k-1|k-1} + Q_{i,k}}{P_{i,k-1|k-1} + Q_{i,k} + \delta_{i,k}} \tag{24}$$

The above equations are obtained by substituting the prediction equations of the KF (13-14) into the update equations (19-21). $\tilde{d}_{i,k}$ is the predicted drift at the beginning of stage k, before the correction. In this case, $\tilde{d}_{i,k} = F\hat{d}_{i,k-1|k-1} = \hat{d}_{i,k-1|k-1}$, a straightforward prediction given by the KF solution. The variances $Q_{i,k}, \delta_{i,k}, P_{i,k-1|k-1}$ and $P_{i,k|k}$ are numbers, and therefore the solution is easy to compute. Once $\hat{d}_{i,k|k}$ is obtained, it is used as the predicted drift $\tilde{d}_{i,k+1}$ for the next stage. This allows for the correction of reading $r_{i,k+1}$. The solution is implemented in a decentralised iterative procedure i.e. it is run in each node and at each time step to estimate its drift $d_{i,k}$. Using this estimation; $r_{i,k+1}$ is corrected to $x_{i,k+1}$ and the drift $d_{i,k+1}$ is estimated again and so on. The block diagram shown in figure 4, with KF as the last sub-block, describes the smooth drift detection and correction algorithm. The algorithm is summarised as follows:

Decentralised error correction algorithm for smooth drifts

For each node i

- At step k, the predicted drift $\tilde{d}_{i,k} = \hat{d}_{i,k-1|k-1}$ and the previous time step process variance $P_{i,k-1|k-1}$ are available.

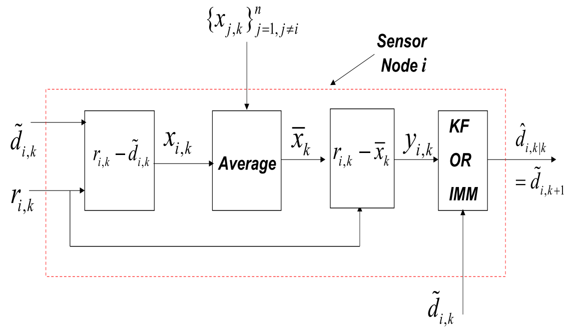

Figure 4. A block diagram for the drift detection and correction algorithms

- Each node i obtains its reading $r_{i,k}$
- The corrected reading is calculated, $x_{i,k} = r_{i,k} - \widetilde{d}_{i,k}$ and then transmitted to the neighbouring nodes.
- Each node computes the average $\overline{x}_k$.
- The Drift measurement $y_{i,k} = r_{i,k} - \overline{x}_k$ is computed.
- Substituting into (22-24) results in the current time step estimates $\hat{d}_{i,k|k}$ and $P_{i,k|k}$.
- The projected drift $\widetilde{d}_{i,k+1} = \hat{d}_{i,k|k}$ is obtained and the algorithm iterates.

IV. Estimation and correction of unsmooth drifts

In this section we present a probabilistic approach that accounts for errors in sensors measurements and instantly captures drifts that have surges and sudden escalations. Such drift behaviour is not followed well by the KF algorithm given in the previous section. The standard KF with single drift model is limited in performance since it does not efficiently respond to changes in the dynamics as the drift changes abruptly at some points. Therefore, we make use of the Interacting Multiple Model (IMM) in our solution since it is designed to deal with abrupt changes in the estimated states. The IMM approach is originally used in target tracking to track manoeuvring objects that show sudden changes in their dynamics [16], [17], [21], [22]. In accordance with the IMM algorithm, each sensor is assigned an M number of modes to account for the possible jumps in the drift. Our solution for the sudden step drift problem (also works for smooth drift) consists of the following iterative steps: As for the case of smooth drift, at time step k, a reading $r_{i,k}$ is made by node i. Rather than sending the reading as it is to it's neighbours, the node is aware of its drift $d_{i,k}$, and has a predicted value $\widetilde{d}_{i,k}$ for it at this stage. It is taken to be equal to the estimate of the drift made at the previous time instant $\widetilde{d}_{i,k} = \hat{d}_{i,k-1|k-1}$. Using this estimate of the drift, the node computes the corrected sensor reading $x_{i,k}$ and sends it to it's neighbours. Each sensor computes the average $\overline{x}_k = \sum_{i=1}^{n} x_{i,k}/n$ to self-assess it's measurements. To account for the possible jumps, the drift with abrupt changes is modelled as a jump markovian linear system. It is a system whose parameters evolve according to the realisation of a finite state markov chain [23]. Mathematically, we model the drift $d_{i,k}$ to belong to the set of models defined by (25):

$$\{d_{i,k} = d_{i,k-1} + u_i^{\theta} + v_{i,k}^{\theta}\}_{\theta=1}^{M} \quad v_{i,k}^{\theta} \sim N(0, Q_{i,k}^{\theta}) \tag{25}$$

where $\theta = 1, 2, \ldots M$, u_i^{θ} is the input or jump corresponding to θ_{th} model for sensor i. $v_{i,k}^{\theta}$ is the process noise for each model. It is taken to be Gaussian with zero mean and variance $Q_{i,k}^{\theta}$. We assume that all models have the same variance. Therefore, we denote the process variance for all the modes as $Q_{i,k}$.

Equation (25) 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^{θ}. The resultant estimated drift for node i at time instant k, $\hat{d}_{i,k|k}$, would be a weighted combination of the estimated drift of each model $\hat{d}_{i,k|k}^{\theta}$. The resultant estimated drift for each node $\hat{d}_{i,k|k}$ is found as will be shown later in this section by:

$$\hat{d}_{i,k|k} = \sum_{\theta=1}^{M} \mu_{i,k|k}^{\theta}\, \hat{d}_{i,k|k}^{\theta}$$

where $\mu_{i,k|k}^{\theta}$ is the model probability. It is the probability that the estimated drift $\hat{d}_{i,k|k}$ follows the drift model $\hat{d}_{i,k|k}^{\theta}$ 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 (25). Since the sensor measurement $r_{i,k}$ usually suffers from random error $w_{i,k}$ and systematic error (drift/bias) $d_{i,k}$, the reading or measurement of sensor i is given by:

$$r_{i,k} = T_k + d_{i,k} + w_{i,k} \quad w_{i,k} \sim N(\mu_{i,k}, R_{i,k})$$

where T_k is the actual (ground truth) value of the measured variable at sensor i and $w_{i,k}$ is the measurement noise and is taken here to be a Gaussian noise with zero mean ($\mu_{i,k} = 0$) and variance $R_{i,k}$.

Similar to the previous section, we denote the corrected measurement of sensor i at time instant k as $x_{i,k}$. $x_{i,k}$ is never sensed but calculated. It is the difference between the sensor reading and the estimated drift and is calculated by $x_{i,k} = r_{i,k} - d_{i,k}$ to result in $x_{i,k} = T_k + w_{i,k}$.

We also define $y_{i,k}$ in (2) as the difference between the measurement $r_{i,k}$ and the average of corrected sensors measurements $\overline{x}_k$ and refer to $y_{i,k}$ as the drift measurement of node i at time instant k.

$$y_{i,k} = r_{i,k} - \overline{x}_k \tag{26}$$

Since the sensors are densely deployed and the instantiations of drifts in the sensors are random, we use the average of corrected sensors' measurements close to node i as an approximate estimate for the expectation of actual (ground truth) value $\overline{x}_k = E\{T_k\} + \frac{1}{n}\sum_{j=1}^{n} w_{j,k}$.

Following the same reasoning given in the case of smooth drift in the previous chapter, $y_{i,k}$ is also expressed by:

$$y_{i,k} = d_{i,k} + \psi_{i,k} \quad \psi_{i,k} \sim N(0, \delta_{i,k}) \tag{27}$$

where $\psi_{i,k} = w_{i,k} - \frac{1}{n}\sum_{j=1}^{n} w_{j,k}$ is the drift measurement noise and is actually a mixture of Gaussians that can

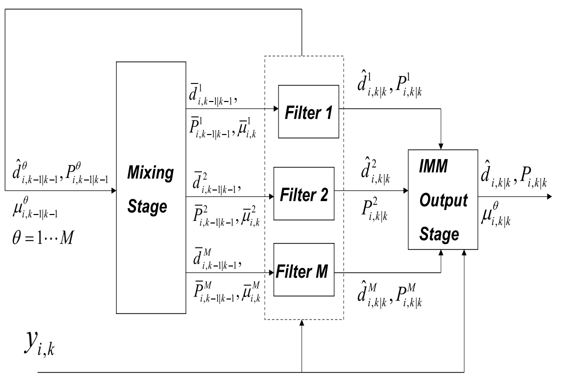
$\hat{d}^{\theta}_{i,k-1|k-1}, P^{\theta}_{i,k-1|k-1}$
$\mu^{\theta}_{i,k-1|k-1}$
$\theta = 1 \cdots M$
Mixing Stage
$\bar{d}^{1}_{i,k-1|k-1}$, $\bar{P}^{1}_{i,k-1|k-1}, \bar{\mu}^{1}_{i,k}$
$\bar{d}^{2}_{i,k-1|k-1}$, $\bar{P}^{2}_{i,k-1|k-1}, \bar{\mu}^{2}_{i,k}$
$\bar{d}^{M}_{i,k-1|k-1}$, $\bar{P}^{M}_{i,k-1|k-1}, \bar{\mu}^{M}_{i,k}$
Filter 1
Filter 2
Filter M
$\hat{d}^{1}_{i,k|k}, P^{1}_{i,k|k}$
$\hat{d}^{2}_{i,k|k}$ $P^{2}_{i,k|k}$
$\hat{d}^{M}_{i,k|k}, P^{M}_{i,k|k}$
IMM Output Stage
$\hat{d}_{i,k|k}, P_{i,k|k}$
$\mu^{\theta}_{i,k|k}$
$y_{i,k}$

Figure 5. The IMM step

be approximated by a Gaussian $\psi_{i,k} \sim N(\pi_{i,k}, \delta_{i,k})$ with mean $\pi_{i,k} = 0$ and variance $\delta_{i,k} = R_{i,k} - \frac{1}{n}\sum_{j=1}^{n} R_{j,k}$.

Referring to equations (25) and (27) 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 $\mu^{\theta}_{i,k|k} = p\,(model_{i,k} = \theta | y^{k}_{i})$ depending on the measured values until that time step. Switching between models is governed by a pre-defined Markov transition matrix Γ of dimension $M \times M$ for M models.

$$\Gamma = \begin{bmatrix} \gamma_{11} & \cdots & \gamma_{1M} \\ \vdots & \cdots & \vdots \\ \gamma_{M1} & \cdots & \gamma_{MM} \end{bmatrix} \tag{28}$$

where $\gamma_{\alpha\theta} = p\,(model_{i,k} = \theta | model_{i,k-1} = \alpha)$ which is the probability of switching from model α to model θ in single 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 models probabilities $\{\mu^{\theta}_{i,k-1|k-1}\}^{M}_{\theta=1}$, estimated drifts $\{\hat{d}^{\theta}_{i,k-1|k-1}\}^{M}_{\theta=1}$ and associated covariances $\{P^{\theta}_{i,k-1|k-1}\}^{M}_{\theta=1}$. Unlike our standard Kalman Filter drift tracking algorithm, the previous estimates are not used as priors for the M Kalman Filters. Instead, the predicted models probabilities $\{\overline{\mu}^{\,\theta}_{i,k} = \sum_{\alpha=1}^{M} \gamma_{\alpha\theta}\, \mu^{\alpha}_{i,k-1|k-1}\}^{M}_{\theta=1}$ are calculated. Then the previous estimates together with $\{\overline{\mu}^{\,\theta}_{i,k-1}\}^{M}_{\theta=1}$ are used in the mixing stage to calculate $\{\overline{d}^{\,\theta}_{i,k-1|k-1}\}^{M}_{\theta=1}$ and $\{\overline{P}^{\,\theta}_{i,k-1|k-1}\}^{M}_{\theta=1}$. The mixing stage drift estimates and the associated covariances are then fed as priors to the corresponding M filters (substituted in KF equations(22-24)) to result in the posterior models estimates $\{\hat{d}^{\theta}_{i,k|k}\}^{M}_{\theta=1}$,$\{P^{\theta}_{i,k|k}\}^{M}_{\theta=1}$.

The output of the IMM algorithm is then found by first updating the models probabilities $\{\mu^{\theta}_{i,k|k}\}^{M}_{\theta=1}$, which are used then together with the outputs of the M Kalman filters to find $\hat{d}_{i,k|k}$ and $P_{i,k|k}$. The algorithm then reiterates taking the predicted drift at time step $k+1$ to be equal to the estimated drift at the previous time step $\tilde{d}_{i,k+1} = \hat{d}_{i,k|k}$.

The block diagram shown in figure 4, with IMM as the last sub-block, describes the unsmooth drift detection and correction algorithm. The IMM block in figure 4 is further explained in figure 5. The full derivation of the IMM algorithm can be found in [16], [17].The steps of our unsmooth drift tracking algorithm are stated below:

Decentralised Unsmooth Drift Correction Algorithm

For each node i

- At step k, a predicted drift $\tilde{d}_{i,k} = \hat{d}_{i,k-1|k-1}$ is available.
- The prior model probabilities $\mu^{\theta}_{k-1|k-1}$ are 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 neighbouring nodes.
- Each node computes the average $\overline{x}_{i,k}$.
- The drift measurement $y_{i,k} = r_{i,k} - \overline{x}_k$ is obtained.
- The predicted model probabilities are calculated

$$\overline{\mu}^{\,\theta}_{i,k} = \sum_{\alpha=1}^{M} \gamma_{\alpha\theta}\, \mu^{\alpha}_{i,k-1|k-1}$$

- Mixing stage

$$\begin{aligned} \overline{d}^{\,\theta}_{i,k-1|k-1} &= \sum_{\alpha=1}^{M} \frac{\gamma_{\theta\alpha}\, \mu^{\alpha}_{i,k-1|k-1}}{\overline{\mu}^{\,\theta}_{i,k}}\, \hat{d}^{\alpha}_{i,k-1|k-1} \\ \overline{P}^{\,\theta}_{i,k-1|k-1} &= \sum_{\alpha=1}^{M} \frac{\gamma_{\theta\alpha}\, \mu^{\alpha}_{i,k-1|k-1}}{\overline{\mu}^{\,\theta}_{i,k}}\, (P^{\alpha}_{i,k-1|k-1} \\ &\quad + [\hat{d}^{\alpha}_{i,k-1|k-1} - \overline{d}^{\,\theta}_{i,k-1|k-1}]^2) \end{aligned}$$

- Kalman Filter update stage

$$\begin{aligned} \hat{d}^{\theta}_{i,k|k} &= \overline{d}^{\,\theta}_{i,k-1|k-1} + u^{\theta}_{i} + K(y_{i,k} - \overline{d}^{\,\theta}_{i,k-1|k-1}) \\ P^{\theta}_{i,k|k} &= (\overline{P}^{\,\theta}_{i,k-1|k-1} + Q_{i,k})(1-K) \\ K &= \frac{\overline{P}^{\,\theta}_{i,k-1|k-1} + Q_{i,k}}{\overline{P}^{\,\theta}_{i,k-1|k-1} + Q_{i,k} + \delta_{i,k}} \end{aligned}$$

- IMM output stage
  The Model probabilities are updated

$$\mu^{\theta}_{i,k|k} = \frac{\overline{\mu}^{\,\theta}_{i,k}\, e^{-\frac{(y_{i,k} - \overline{d}^{\,\theta}_{i,k-1|k-1} - u^{\theta}_{i})^2}{2A}}}{\sum_{\theta=1}^{M} \overline{\mu}^{\,\theta}_{i,k}\, e^{-\frac{(y_{i,k} - \overline{d}^{\,\theta}_{i,k-1|k-1} - u^{\theta}_{i})^2}{2A}}}$$

  where $A = \overline{P}^{\,\theta}_{i,k-1} + Q_{i,k} + \delta_{i,k}$. The resultant estimated drift and its associated covariance are updated as follows:

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

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

The systems described in this section and the previous section are completely observable at the deployment stage of the sensor network, when no sensors are drifting. However, as the sensors start to develop drift the system becomes partially observable. When employing our algorithm, the drifts are detected and consequently the readings of the sensors are corrected to become close to the ground truth. This together with that the probability of many sensors start drifting simultaneously is low, enhance our ability to extend the period of observability of the system. Hence, extending the useful time of the sensor network. Thus, giving us the opportunity for making the most use of the network.

V. 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 densely deployed sensor nodes measuring the temperature in a certain area. We assume that 2 sensors are developing smooth drifts with jumps of the forms shown in figure 3. We compare the IMM drift tracking algorithm with the plain KF drift tracking algorithm under the same drift and random error (noise) scenarios. The drift measurement variance $\delta_{i,k}$ for each node is chosen from [0.005-0.01] and the state variance $Q_{i,k}$ is taken to be 0.001. The number of models we consider in our evaluation of IMM algorithm is $M = 11$.

The results of the KF drift tracking algorithm are shown in figures 6 and 7, whereas, the results of the IMM drift tracking algorithm are shown in figures 8 and 9. Comparing figures 9 and 7, it is clear that both algorithms follow the drift in node 1. However, the IMM drift algorithm performs considerably better. It 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 drift tracking algorithm in terms of speed and accuracy of following the drift.

Looking at figures 8 and 6, It is clear that both the IMM and the KF drift tracking algorithms extend the effective operational life 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

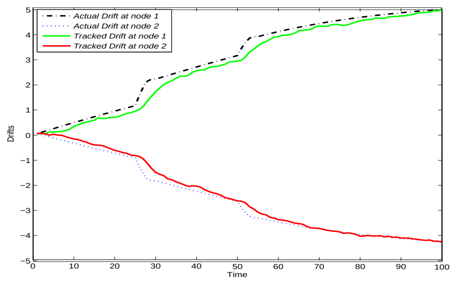

Figure 7. Actual and estimated drifts in nodes 1 and 2 for KF

units when the IMM or the KF algorithms 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 detection and correction algorithms. It is also worth noting from figures 8 and 6 that the difference between the actual and corrected reading curves tend to be on average 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.

In addition to the drift, we consider the bias problem. As we mentioned earlier, the bias is the starting reading error or in other words, the drift at time zero. We assume that the sensor nodes are factory calibrated before deployment and so it is unlikely that a high percentage of the sensors will suffer from bias. In a sensible situation, if most of the sensors are without bias at time 0, then such a bias can be captured by both of the proposed solutions as a constant drift of a certain amplitude. Figures 10 and 11 show the KF and IMM algorithms results when both sensors 1 and 2 suffer from bias and drift. It is obvious from both figures that the two algorithms efficiently capture and correct the bias. However, the IMM algorithm catches and corrects the bias faster. It is important to note here that if many sensors suffer from the bias, then both solutions will not be accurate. Having many sensors with initial bias, highly contradicts with one of our main assumptions that one sensor will start drifting at a time. Furthermore, it is unrealistic to have many sensors with initial bias as the sensors have to be factory calibrated before deployment.

Other comparisons between the two algorithms can be made by looking at figure 12 and figure 13 which show the RMS error for both algorithms when 2 sensors in the

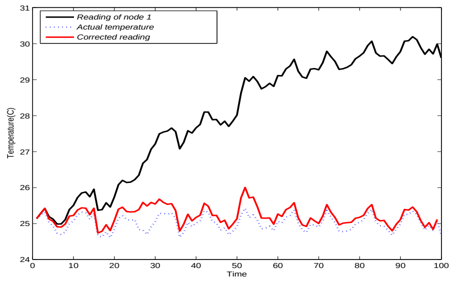

Figure 6. The reading of node 1, the actual temperature and the KF estimated (corrected) reading.

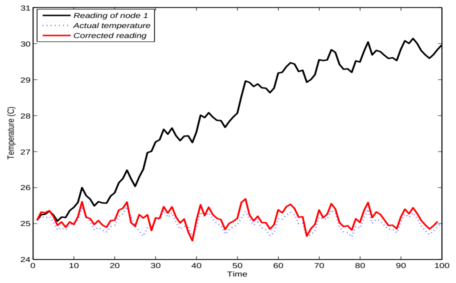

Figure 8. The reading of node 1, the actual temperature and the IMM estimated (corrected) reading.

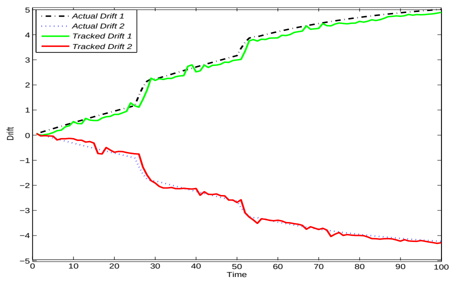

Figure 9. Actual and estimated drifts in nodes 1 and 2 for IMM

sub-network are subject to smooth drifts and unsmooth drifts, respectively. It is clear, in both scenarios, that the IMM drift tracking algorithm performs better than simple KF drift tracking algorithm in terms of speed of following the drift and in terms of the RMS error between the estimated and actual drifts. However, the improved performance of IMM is at the cost of the increased computational complexity. We use the processing time required by each algorithm as a measure of its computational complexity. Table I shows the average processing time required by the KF based algorithm and the IMM based algorithm (for different number of models) as reported by our MatLab simulations. The $Ratio$ column clearly shows that the IMM based algorithm requires approximately $2M$ the time required by the KF based algorithm. Obviously, the computational complexity can be reduced by reducing the number of models M used in the IMM algorithm. Figure 14 shows the RMS error in the estimated drift for the KF based algorithm and the IMM based algorithm for different number of models. We notice that the RMS error reduces with M. However, the rate of change in the RMS error also decreases with M; the difference between the RMS errors for $M = 7$ and $M = 11$ is very small. In fact, it is well known in target tracking literature that using more models does not necessarily lead to better estimation, whereas it definitely increases the computational complexity [24]. Therefore, M should be chosen carefully. Alternatively, a model such as the variable structure IMM (VSIMM) which adaptively determines the minimal number models for estimating the state may be used [25].

It is important here to note that the speed of following the drift for the KF based algorithm can be increased by increasing $Q_{i,k}$. However, this will lead to more oscillatory response and will result in increasing the RMS error. Adding another component to the state vector, namely, the speed of the drift while maintaining the same value of $Q_{i,k}$, will result in faster tracking of the drift with less RMS error. Unfortunately, this will increase the mathematical complexity, as the problem of estimating the drift will then involve matrix multiplications and inversions, which is undesirable in a wireless sensor with limited computational capability.

We conducted several simulation scenarios and observed that the method worked as long as not all sensor start drifting at the same instant of time. Generally speaking, we noticed that the performance of both algorithms is dependent on the number of drifting sensors, the amplitude of drifts or biases and the instantiations of drifts. If all the sensors suffer from considerable biases at time zero the method will not work accurately.

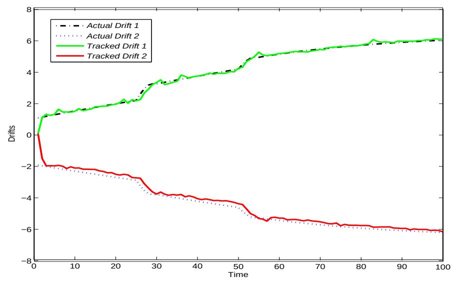

Figure 11. Actual and estimated bias/drift in nodes 1 and 2 for IMM

VI. Conclusion and Future Research

In this paper we have proposed a formal Bayesian framework for estimating sensor errors in a WSN based on the assumption that neighbouring sensors have correlated measurements and that the instantiation of drift in a sensor is uncorrelated with other sensors. The sensors in the neighbourhood are assumed to be densely deployed. Hence, the average of their corrected readings is taken as a basis for each sensor to self assess it measurements. We have introduced two probabilistic procedures that capture drifts; the first captures smooth drifts, whereas the other, captures both smooth drifts and unsmooth drifts (drifts with jumps). The solutions are computationally simple and scalable as they are decentralised dealing with one state KF in the case of smooth drifts and one state IMM in the case of unsmooth drifts. The IMM based algorithm performs better in both the smooth and unsmooth cases but at the cost of increased computational complexity.

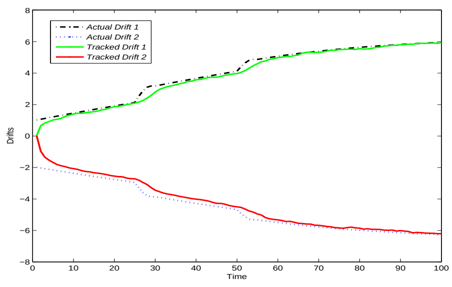

Figure 10. Actual and estimated bias/drift in nodes 1 and 2 for KF

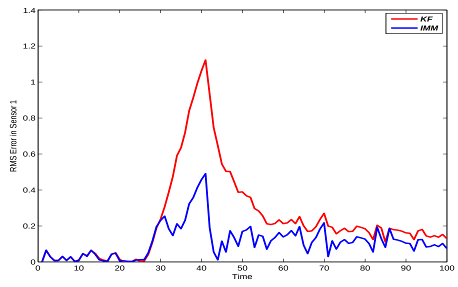

Figure 12. RMS error for both algorithms under smooth drift scenario

Table I
PROCESSING TIMES REQUIRED BY KF BASED AND IMM BASED DRIFT ESTIMATION AND CORRECTION ALGORITHMS.

Drift estimation and correction algorithm	Processing time/iteration (PT)	$Ratio = \frac{PT(Any)}{PT(KF)}$
KF	1.4 ms	1
IMM ($M = 3$)	8.51 ms	6.079
IMM ($M = 7$)	19.83 ms	14.16
IMM ($M = 11$)	32.49 ms	23.21

In future, we will address the drift/bias problem in sparsely deployed WSNs. In this case, the temperature (or any other measured phenomenon) is considered to vary with distance and time. Therefore, the average of the neighbours' readings cannot be used by any sensor to self asses its own measurements. Alternatively, we will use machine learning and regression techniques to predict the measurement of a sensor in terms of its neighbours readings as a first step for detecting the drift and correcting the readings. Moreover, we are planning to implement and test our solutions in WSNs deployed in outdoor environment.

VII. ACKNOWLEDGMENT

We acknowledge the support of THALES, Australia funding 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," *Int. Conference on Acoustics, Speech, and Signal Processing)*, May 2001.
[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Comp. Networks*, vol. 38, pp. 393–422, 2002.
[3] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak, "A collaborative approach to in-place sensor calibration," *Int. Workshop on Information Processing in Sensor Networks*, pp. 301–316, 2003.
[4] M. Takruri and S. Challa, "Drift aware wireless sensor networks," in *Proc. of the 10th intl. conference on information fusion*, (Quebec City, Canada), July 2007.
[5] B. Krishnamachari and S. Iyengar, "Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Tran. Computers*, vol. 53, no. 3, pp. 241–250, 2004.
[6] N. Okello and G. Pulford, "Simultaneous registration and tracking for multiple radars with cluttered measurements," *IEEE Signal Processing Workshop on Statistical Signal and Array Processing*, pp. 60–63, June 1996.
[7] N. Okello and S. Challa, "Simultaneous registration and track fusion for networked trackers," in *conference on information fusion*, (Montréal, Canada), August 2003.
[8] L. Brown, "A survey of image registration techniques," *ACM Computing Surveys*, vol. 24, pp. 326–376, December 1992.
[9] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, pp. 977–1000, June 2003.
[10] L. Balzano and R. Nowak, "Blind calibration of sensor networks," *Information Processing in Sensor Networks*, April 2007.
[11] B. Hoadley, "A baysian look at inverse linear regression," *J. of the American Stats. Association*, vol. 65, pp. 356–369, March 1970.
[12] J. Feng, S. Megerian, and M. Potkonjak, "Model-based calibration for sensor networks," *Sensors*, pp. 737 – 742, October 2003.
[13] L. Balzano, "Addressing fault and calibration in wireless sensor networks," Master's thesis, University of California, Los Angeles, California, 2007.
[14] M. Takruri, K. Aboura, and S. Challa, "Distributed recursive algorithm for auto calibration in drift aware wireless sensor networks," in *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering* (K. Elleithy, ed.), pp. 21–25, Springer, 2008.
[15] M. Takruri, S. Challa, and R. Chakravorty, "Auto calibration in drift aware wireless sensor networks using the interacting multiple model algorithm," in *Mosharaka International Conference on Communications, Computers and Applications (MIC-CCA 2008)*, (Amman, Jordan), August 2008.
[16] S. Challa, R. Evans, M. Morelande, and D. Musicki, *Fundamentals of Object Tracking*. Cambridge University Press, 2008.
[17] Y. Bar-Shalom, *Estimation and Tracking : Principles, Techniques and Software*. Boston Artech House, 1993.
[18] D. Koks and S. Challa, "An introduction to bayesian and dempster-shafer data fusion," tech. rep., 2005.
[19] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Tran. ASME J. Basic Eng.*, pp. 35–45, March 1960.
[20] G. Welch and G. Bishop, "An introduction to the kalman filter," 1995.
[21] Y. Bar-Shalom, S. Challa, and H. Blom, "Imm estimator versus optimal estimator for hybrid systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 3, pp. 986–991, 2005.
[22] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *IEEE Tran. on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.
[23] L. Johnston and V. Krishnamurthy, "An improvement to the interacting multiple model (imm) algorithm," *IEEE Tranctions on Signal Processing*, vol. 49, no. 12, pp. 2909–2923, 2001.
[24] X. Wang, S. Challa, R. Evans, and X. R. Li, "Minimal submodel-set algorithm for maneuvering target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1218–1231, 2003.
[25] X. R. Li, "Engineer's guide to variable-structure multiple-model estimation for tracking," in *Multitarget-Multisensor Tracking: Applications and Advances* (Y. Bar-Shalom and W. Blair, eds.), vol. 3, ch. 10, pp. 499–567, Boston, MA: Artech House, 2000.

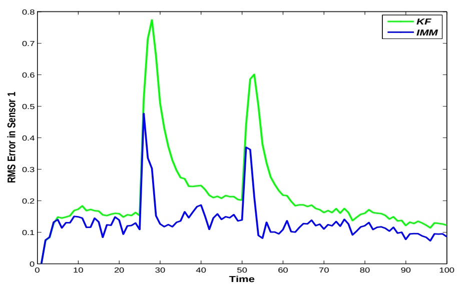

Figure 13. RMS error under unsmooth drift scenario

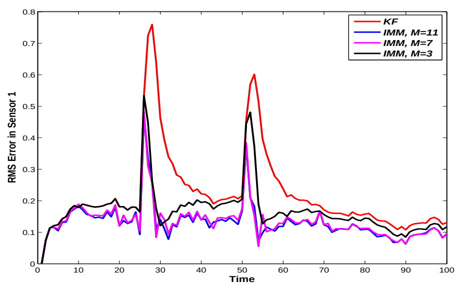

Figure 14. RMS error under unsmooth drift for different M.