Maximum Entropy-Based Auto Drift Correction Using High- and Low-Precision Sensors

PUNIT RATHORE and DHEERAJ KUMAR, The University of Melbourne, Australia
SUTHARSHAN RAJASEGARAR, Deakin University, Australia
MARIMUTHU PALANISWAMI, The University of Melbourne, Australia

With the advancement in the Internet of Things (IoT) technologies, variety of sensors including inexpensive, low-precision sensors with sufficient computing and communication capabilities are increasingly deployed for monitoring large geographical areas. One of the problems with the use of inexpensive sensors is that they often suffer from random or systematic errors such as drift. The sensor drift is the result of slow changes that occur in the measurement driven by aging, loss of calibration, and changes in the phenomena being monitored over a time period. These drifting sensors need to be calibrated automatically for continuous and reliable monitoring. Existing methods for drift detection and correction do not consider the measurement errors or uncertainties present in those inexpensive low-precision sensors, hence, resulting in unreliable drift estimates. In this article, we propose a novel framework to automatically detect and correct the drifts by employing Bayesian Maximum Entropy (BME) and Kalman filtering (KF) techniques. The BME method is a spatiotemporal estimation method that incorporates the measurement errors of low-precision sensors as interval quantities along with the high-precision sensor measurements in their computations. Our scheme can be implemented in a centralized as well as in a distributed manner to detect and correct the drift generated in the sensors. For the centralized scheme, we compare several Kriging-based estimation techniques in combination with KF, and show the superiority of our proposed BME-based method in detecting and correcting the drift. We also propose a multivariate BME framework for drift detection, in which multiple features can be used to improve the drift estimates. To demonstrate the applicability of our distributed approach on a real-world application scenario, we implemented our algorithm on each wireless sensor node in order to perform in-network drift detection. The evaluation on real IoT datasets gathered from an indoor and an outdoor deployments reveal the superiority of our method in correctly identifying and correcting the drifts that develop in the sensors, in real time, compared to the existing approaches in the literature.

CCS Concepts: • **Information systems** → **Sensor networks**; • **Computing methodologies** → **Distributed algorithms**; • **Mathematics of computing** → *Probabilistic inference problems*;

Additional Key Words and Phrases: Sensor data reliability, internet of things, distributed computing, anomaly detection, spatial estimation, kalman filtering

This work was supported by the Australian Research Council (ARC) Linkage Project grant (LP120100529), the ARC Linkage Infrastructure, Equipment and Facilities scheme (LIEF) grant (LF120100129), HP2020 OrganiCity grant and EUFP7 SocioTal grant.

Authors' addresses: P. Rathore, D. Kumar, and M. Palaniswami, Department of Electrical and Electronics Engineering, The University of Melbourne, Parkville, Melbourne, Australia, 3010; emails: {prathore, d.kumar}@student.unimelb.edu.au, palani@unimelb.edu.au; S. Rajasegarar, School of Information Technology, Deakin University, Geelong, Australia, 3125; email: sutharshan.rajasegarar@deakin.edu.au.

https://doi.org/10.1145/3085579

ACM Reference format:
Punit Rathore, Dheeraj Kumar, Sutharshan Rajasegarar, and Marimuthu Palaniswami. 2017. Maximum Entropy-Based Auto Drift Correction Using High-and-Low Precision Sensors. *ACM Trans. Sen. Netw.* 13, 3, Article 24 (August 2017), 41 pages.
https://doi.org/10.1145/3085579

1 INTRODUCTION

With the advances in the computation capabilities of inexpensive, low-precision sensor nodes, wireless sensor networks (WSNs) via emerging Internet of Things (IoT) technologies are increasingly facilitating complex monitoring applications. For applications involving monitoring in dynamic environments, the quality of the spatiotemporal (ST) data is of utmost importance for the real-time decision making. Consider a smart city application where particulate matter (PM) sensors are deployed at different locations in the city. While top-end pollution monitoring equipment that are capable of reliably measuring very low levels of pollutants are available, in the current practice, monitoring is done at a low spatial resolutions due to the high cost of the equipments. For example, a region of over 2,200km^2 surrounding the city of Melbourne in Victoria, Australia, is monitored using only about 10 sensor stations [42]. The sparsity between monitoring stations leads to unreliable pollution levels estimates at the locations that are far from the monitoring stations, making them unreliable. Obtaining reliable measurements with low deployment cost can be achieved by deploying a mix of fewer number of high-precision sensors at low spatial resolutions, and a larger number of inexpensive, low-precision sensors at high spatial resolutions over an area of interest.

As the WSN is operational, some of the inexpensive, low-precision sensor nodes tend to develop drift in their measurements [22] and hence begin to transmit erroneous values to base stations. These drifted sensors need to be recalibrated to maintain accurate measurements. Generally, high-precision sensor stations do not drift much or are periodically calibrated to maintain their measurement accuracy. Manual calibration is possible for these monitoring stations, as they are considerably less in numbers than the low-precision sensors. A typical WSN application uses hundreds of nodes distributed over a large area, sometimes deployed in harsh environment or in nearly inaccessible locations. Therefore, it is impractical to calibrate them manually, albeit it is extremely time-consuming and expensive. Therefore, there is a need to identify the drifting sensors and calibrate them automatically.

Existing drift detection techniques in the literature often require neighborhood sensor measurements to estimate the value at any sensor node location. This requires data communication among sensor nodes in the network. However, the inexpensive sensor nodes in WSNs have limited memory and energy resources (often battery powered) [8, 43, 44], and the majority of the energy is consumed in radio communication rather than in computation [40]. Therefore, communicating all sensor measurement to a base station for performing drift detection and calibration is an energy-intensive task. Hence, there are advantages in performing in-network drift detection and correction to reduce the need for radio communication in the network and thus prolonging the lifetime of energy-limited WSNs. Further, this in-network processing facilitates early detection and mitigation of drifting errors in the sensors at the source.

There has been some attempts made in the past to detect drift automatically [15, 48–50, 54]. These either use learning-based predictor,s which require prior offline training, or use only neighborhood sensor measurements by ignoring the location information. Most recent work in this area [32, 33] uses geospatial-based estimation method called kriging [39] and Kalman filtering (KF) [31] for automatic drift detection in WSN. Existing methods, including kriging, will not provide reliable estimates in an IoT environment where inexpensive low-precision sensors are used for deployments, which generally have uncertainty in their measurements (i.e., measurement errors). Hence, there is a need to incorporate those uncertainties within the estimation framework for obtaining accurate drift estimation and correction.

In this article, we address the problem of automatic drift detection and correction in sensor networks using the combined data from neighboring high- and low-precision sensors, which measure correlated data. In particular, the property of WSN environmental data that have high-ST correlation [2] among themselves, is exploited to predict the measurement of each sensor utilizing neighboring sensors measurements. The drift value is then iteratively estimated and eventually corrected dynamically. The Bayesian maximum entropy (BME)-based algorithm [46] is employed for estimation, which incorporates high- and low-precision sensor measurements to provide reliable estimates and subsequently use it to iteratively detect and correct the drift via a KF approach.

Our main contribution in this article are as follows:

- We propose a novel BME-based automatic drift detection and correction framework for an IoT environment for both centralized and distributed scenarios.
  - In the centralized approach, all the sensor nodes communicate their data to a central station where all the computation are performed.
  - In a distributed approach, each node communicates with each other to detect and correct their drift locally using on-board processing.
  - We demonstrate the applicability and superiority of our proposed distributed BME scheme by evaluating on real WSN datasets, such as the Intel Research Berkeley Laboratory (IBRL) dataset and an environmental data obtained from deployments in the City of Melbourne, Australia, for real-time urban forest microclimate monitoring.
- We incorporate the BME-based ST estimation technique to obtain estimates from high- and low-precision sensor measurements. To the best of our knowledge, this is the first time such a ST estimation method, which utilizes the high- and low-precision sensors, is used for drift correction. We also compare the existing kriging-based drift detection techniques with our scheme to demonstrate its capability and superiority in detecting drift in an IoT environment.
- We implemented our algorithm in a distributed manner on a real WSN consisting of Libelium sensor nodes to perform in-network drift detection. This require a novel algorithm to be developed for performing the *nonlinear optimization* task at each node. We implemented such an algorithm in this work. To the best of our knowledge, this is the first time computation of multivariate Gaussian probabilities and the optimization task to find minima of nonlinear function in BME is performed on the sensor node itself.
- We analyze the effect of the uncertainty levels, expressed in terms of interval quantities, of low-precision sensor measurements in order to investigate how the measurement errors affect the estimation accuracy. This analysis provides a practical mechanism to select the low-precision sensors with an appropriate measurement error level (or uncertainty level or sensitivity) in order to achieve a given (desired) estimation accuracy within a geographical area.
- We analyze the performance of our algorithm by varying the number of high- and low-precision sensors deployed in a given area. This study helps to choose optimal number of high- and low-precision sensors such that the communication overhead in the network is minimized.
- We propose a novel *multivariate* BME-based scheme for drift detection where additional features can be used in conjunction with the principal feature in order to improve the

estimates of principal feature at each estimation location. This in turn improves the accuracy of drift detection and correction. We demonstrate this using humidity as an additional feature in conjunction with temperature (principal feature) in our experiment.

The remainder of this article is organized as follows. Section 2 presents the related work on drift detection and correction methods. In Section 3, we formulate our problem and present the proposed scheme in detail. Section 4 describes the BME-based ST estimation technique used in our scheme for drift detection. In Section 5, we describe kriging-based spatial estimation methods that are used in our experiments for comparison. Section 6 describes KF for drift tracking and correction. In Section 7, we evaluate our proposed scheme on real IoT data. It also discusses the performance of the proposed algorithm for different values of parameters such as the number of neighboring high- and low-precision sensors, the number of drifting sensors, and for different measurement error intervals of low-precision sensors. Section 8 describes the framework of multivariate BME for drift detection. In Section 9, we discuss the implementation of our distributed BME algorithm in a network consisting of real sensor nodes, followed by the conclusion in Section 11.

2 RELATED WORK

Sensor drifting is one of the major problems that have been observed in real life deployments. Sensor drift problem have been addressed for gas sensors such as the one in electronic nose [27] using system identification theory and recursive least square methods [26]. These models have limitations to handle continuously varying variables (e.g., gas concentration gradient). Moreover, these methods are sensor and process dependent and can not be generalized. In other works, multivariate methods such as principle component analysis with partial least square–based methods [4], self-organizing maps [55], and state space estimation models [51] have been used.

A general and simple approach for calibration is to compare the sensor measurement and the ground-truth value for a known stimulus. Drift can be detected by comparing the response to ground truth [10, 24]. This method is referred as nonblind calibration in Ref. [10], since the ground truth is used to calibrate the sensors. In nonblind calibration schemes [45], some reference information, such as prechecked sensor measurements, are needed for calibration. These methods are time-consuming and require human intervention at each stage. Furthermore, these methods do not work effectively for a WSN deployed over a large area.

Given these shortcomings, blind calibration methods attracted more attention as they utilize the redundant information instead of reference information. Many previous works use correlation of sensors to recover or estimate true measurements from the measured data, but most of the existing blind calibration approaches are based on an assumption that sensors are densely deployed, which makes application of these approaches limited. Balzano and Nowak [6] tackled the calibration problem for sensor networks by assuming that the drift produced in sensors are linear with a certain gain and offset, and they proposed a method for estimating these gains and offsets using a subspace matching approach. The calibration process was converted into a problem of solving the linear equations through signal subspace projection. Though this method does not require the ground-truth measurements and human intervention, it can only be used in a controlled environment that has linear drift scenarios.

In Refs. [32, 33, 48–50, 53], the authors proposed new approaches that use prediction techniques to estimate the ground-truth value of the measured variable. What makes these methods different to each other is the techniques used for prediction. All of these approaches assume high ST correlation among sensors. First, the measurement value for any sensor node is predicted using the nearest neighbor nodes, and then this predicted value is subtracted from measured value to obtain the predicted drift. This value is then processed using KF to estimate the correct drift value.

In Takruri et al. [48], a drift aware WSN framework was introduced, where KF and the Interacting Multiple Model (IMM) were used to predict and correct the drift in a dense network. The average of the neighbor nodes' measurements is used to estimate the ground-truth value for any particular node in the network. This value and the measured value from the node is used to calculate the drift using the KF. This algorithm is also able to detect and correct the sensor bias that appear in the node. This approach is limited for use in dense networks, as the average of the neighbor nodes' measurement cannot be reliably estimated in the case of a sparsely deployed WSN. To predict the drift in a sparsely deployed sensor network, Takruri et al. [50] implemented a support vector regression (SVR) and an unscented Kalman filtering (UKF)-based drift detection and correction algorithm. UKF is better than the simple KF because it additionally takes into the account mean and variance information. They initially developed a model based on ST correlation between each sensor and the corrected readings from its neighboring nodes, using an SVR. In the training phase of SVR, these correct sensor readings are expressed as a function of its own reading and the neighboring sensors' previous corrected readings. In the testing phase, this function is used by each sensor to predict its correct value. Then, this correct value is fed to the KF to detect the error (drift) in the measured values and to correct them. In this approach, SVR is trained only once during the lifetime of the network. They used only the trained support vectors to upload to the sensor node, which results in less communication and memory overhead. This method needs prior training for efficient working, which is a computationally expensive task. Hence, it is not suitable for use in a dynamic environment, for example, if the sensor starts drifting during the training phase or if there is a significant change happens in the observed phenomena during the monitoring period.

Most of existing algorithms make use of the available sensor data and ignore the location information. To address this problem, in a recent work Kumar et al. [32, 33], a geospatial-based estimation method, called kriging [39] and a KF were used for automatic drift detection. Kriging is a spatial interpolation method that estimates the value of a variable at an arbitrary location by utilizing correlation as well as location information of sensor nodes in that area. In this scheme, the ground-truth value at any sensor node is predicted from its neighboring sensor nodes using the kriging approach. This predicted value is then used to calculate sensor drift by subtracting it from the measured sensor value. This estimated drift is processed using a KF to obtain the predicted drift, and then eventually corrected. This algorithm has shown considerable improvement in the performance (root mean square error (RMSE)) against previously discussed methods.

To ensure accurate drift detection, a reliable estimate of the ground-truth value is necessary. Existing methods, including kriging, may not provide reliable estimates in an IoT environment where inexpensive low-precision sensors are used, which generally have uncertainty associated with their measurements (i.e., measurement errors). Very limited work has been done so far on the ST modeling of low-precision measurements [29] obtained from inexpensive sensors. In Sutharshan et al. [42], a BME estimation approach was used that incorporates high-precision pollution monitoring sensors along with the sparsely deployed low-precision sensors, which are less expensive. However, they have not addressed the drifting problem in the sensors.

In this article, we use the BME-based [46] estimation algorithm, which incorporates the low-precision sensor measurements with high-precision sensor measurements to produce reliable estimates. We then use this to iteratively detect and correct the drift using a KF. BME is an improved method of ST estimation that combines information theory with Bayesian statistics. Some of the advantages of BME [46] over traditional kriging estimation methods are as follows:

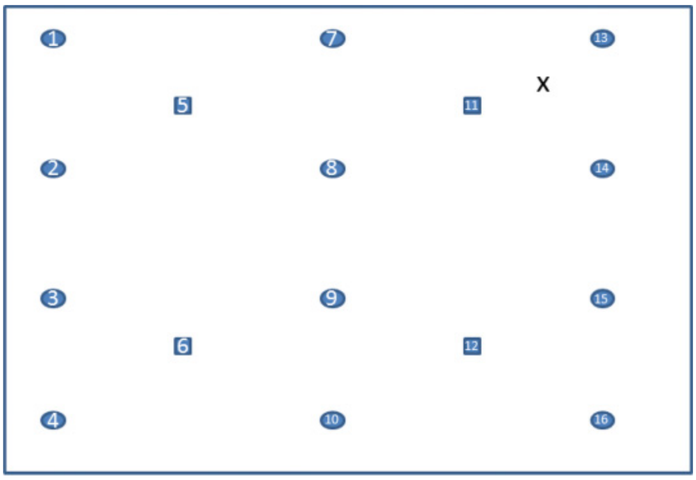


Fig. 1. High-precision and low-precision sensor nodes. Squares represent high-precision sensor nodes, circles represent low-precision sensor nodes, and X represents the estimation point.

(1) Unlike kriging, it incorporates general knowledge of process (such as background information, statistical moments, etc.) as well as specificatory knowledge (e.g., sensors measurements in its computation).
(2) In kriging, one can only use hard data (i.e., data without any errors), while in the BME, measurement errors that are present in the low-precision sensor measurements can also be incorporated for estimations in the form of interval or probabilistic distribution. This is called *soft data*.
(3) The BME method provides better confidence intervals than the kriging method.
(4) BME is not restricted to linear estimators, and further non-Gaussian probability distributions can be incorporated into the analysis.

In the following, we formulate our drift detection problem and present our approach in detail.

3 PROBLEM FORMULATION AND APPROACH

Almost all natural processes provide realizations (values) in space-time continuum. Environmental data obtained from various wireless sensor nodes are also ST in nature; thus, each point in this continuum can be represented using space and time information such that $p = (s, t)$, where s is the spatial location represented by longitude and latitude, and t is time. Consider a wWSN where few high-precision sensors N_h in low spatial resolutions and many low-precision sensors N_l in high spatial resolutions are deployed at known geographical locations in the region of interest, as shown in Figure 1, which monitors a phenomenon, say temperature, over $\mathbb{T}$ discrete time points. Hence, a collection of $n_h = N_h \times \mathbb{T}$ high-precision sensor measurements and $n_l = N_l \times \mathbb{T}$ low-precision sensor measurements are collected at $n_d = n_h + n_l$ ST points.

We assume that the high-precision sensors do not drift in our experiments. This is a reasonable assumption as in practice high-precision sensors or top end monitoring stations either do not produce drift or their manual calibration is done periodically, as they are few in numbers. We consider that the drift in the inexpensive sensors is smooth and bidirectional, that is, it changes slowly and can vary both in positive or negative directions [25], and in a linear or exponential manner as shown in Figure 2. The sensor drift generally depends on environmental conditions and is strongly

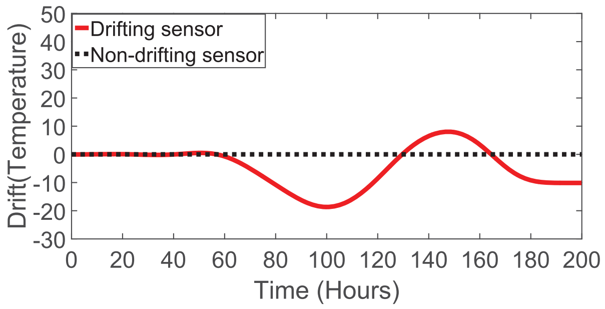


Fig. 2. Example of a smooth bidirectional drift.

related to the sensor's manufacturing process. Hence, the instantiation of drift in different sensors will be different as it is less likely that the two electronic components fail in a correlated manner. This allows us to assume that all the sensors do not start drifting simultaneously, and their drift starting times are random. The other reasonable assumption is that the sensors do not start drifting at the beginning of operation after deployment, as the sensors are usually calibrated before deployment to ensure that they are in working order.

Let T be the ST profile of ground-truth temperature values. The ground-truth value of the temperature at s^{th} sensor location at time t is denoted as $T_{s,t}$. At each time instant t, the sensor node measures its temperature value $r_{s,t}$ for the ground-truth temperature value $T_{s,t}$. Each sensor node keeps track of its drift $\hat{d}_{s,t}$ at time instant t and then reports a drift corrected value $x_{s,t}$ to its neighbors. Ideally, the corrected value $x_{s,t}$ and the measured value $r_{s,t}$ should be equal to the ground-truth temperature $T_{s,t}$, for nondrifting sensors. For drifting sensors, the corrected value is given by

$$x_{s,t} = r_{s,t} - \hat{d}_{s,t} \tag{1}$$

During the process, the corrected value $x_{s,t}$ for each node can be estimated by predicting its measurement from neighborhood sensors' measurements as follows:

$$x_{s,t} = f(\text{Neighborhood nodes' measurement}) \tag{2}$$

These neighborhood sensors are chosen from a group of both hard and soft sensors. For example, consider a sensor network comprising of high- and low-precision sensors, as shown in Figure 1. The node X, called the *central node*, estimates its temperature value from the measurements obtained from N_{nh} number of nearest (neighbor) high-precision sensors and N_{nl} number of nearest low-precision sensor nodes using Equation (2). For example, if we consider $N_{nh} = 1$ and $N_{nl} = 4$, then the node X estimates its value by using the measurements from the high-precision sensor 11 and the low-precision sensors 7,8,13, and 14. In our drift detection framework, each low-precision sensor node behaves as the central node after each measurement instant and collects a temperature reading from its nearest high- and low-precision sensor nodes. Based on these readings, each sensor node predicts the ground-truth temperature value at its location.

Our main objective is to find the most suitable model function $f(\cdot)$ in Equation (2), that is, the ST estimation method which accurately estimates the value $\hat{x}_{s,t}$ by utilizing neighboring sensor measurements that include both high- and low-precision sensors. Existing estimation algorithms discussed in Section 2 make use of measured sensor value and consider the fact that a large number of low-precision sensors are used in the deployment, resulting in inaccurate estimates.

In this work, we use an ST estimation framework, called BME, for improving the estimation accuracy. Compared with the classical geostatistics methods, such as kriging and its variants, BME

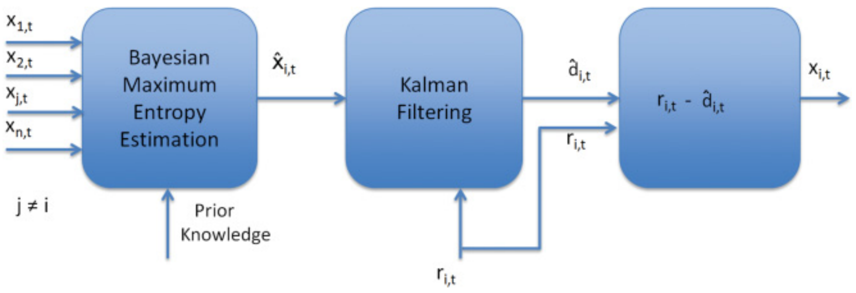


Fig. 3. Block diagram for drift detection and correction at node i (at location s) and time t.

considers two prime knowledge bases, namely the general (prior) knowledge and the geographic region specific knowledge, which incorporates the uncertainty in the data called as soft data (interval or probabilistic data that are uncertain due to, for example, inexact knowledge, experience, intuition, low-precision sensor outputs, etc.), in addition to the hard data (measured values that we are willing to concede as correct with near certainty) to improve the accuracy of the ST analysis.

Once the estimation is obtained, the next task is to model and track the drift present in the sensors and correct them. We use a KF to compute the drift $\hat{d}_{s,t}$ recursively and then use it to obtain a corrected measurement $x_{s,t}$. Each node estimates its current drift $\hat{d}_{s,t}$ using the measured temperature $r_{s,t}$, the estimated temperature $\hat{x}_{s,t}$, and its previous drift $\hat{d}_{s,t-1}$ by means of a KF-based tracking, which is discussed in following sections. A block diagram of the drift tracking process is shown in Figure 3. The steps involved in our proposed drift correction algorithm are shown in Algorithm 1.

ALGORITHM 1: Bayesian Maximum Entropy–Based Drift Correction Algorithm

At every time instance t, the following operations are performed.
Step 1. Each node at location s_i measures its temperature.
Step 2. The low-precision (soft) sensor node at s_i requests its N_{nh} number of neighboring high-precision sensor nodes $\mathbf{S_{nh}} = \{s_{j_h} | j = 1, 2 \ldots N_{nh}; j \neq i\}$, and N_{nl} number of low-precision sensor nodes $\mathbf{S_{nl}} = \{s_{j_l} | j = 1, 2 \ldots N_{nl}; j \neq i\}$ to send their temperature measurements $x_{j,t-1}$. These $x_{j,t-1}$ are drift-corrected measurements from the neighboring *high- and low*-precision sensors computed during the previous time step $t - 1$.
Step 3. Based on the measurements of neighboring *high- and low*-precision sensor nodes, and prior knowledge (sensor measurement errors (uncertainties), mean and covariance function of the field), the node at s_i predicts its temperature $\hat{x}_{i,t}$ using the Bayesian Maximum Entropy–based estimation method as explained in Section 4.
Step 4. The node at s_i estimates its current drift $d_{i,t}$ using the measured temperature $r_{i,t}$, the predicted temperature $\hat{x}_{i,t}$, and its previous drift $\hat{d}_{i,t-1}$ by means of a Kalman filter–based tracking as explained in Section 6.
Step 5. The estimated drift $\hat{d}_{i,t}$ is subtracted from $r_{i,t}$ to obtain the corrected temperature $x_{i,t}$.

4 BAYESIAN MAXIMUM ENTROPY–BASED ESTIMATION

In this work, we employ a BME-based estimation method as the model function $f(\cdot)$ to estimate the correct temperature value for each node and then use that to detect and correct its drift by means of KF-based tracking mechanism. In the following text, we present this technique and show how it integrates low-precision sensor measurements (soft data) with high-precision sensor measurements (hard data) for estimation.

Given a set of N sensor nodes at location $\mathbf{S} = [s_1, s_2 \ldots s_N]$, realization of random variable X (e.g., temperature) at these locations can be represented as $\mathbf{X} = [\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3, , \ldots \mathrm{X}_N]$. The total physical knowledge K regarding a natural process, used by the BME to estimate the values, consists of two primary knowledge bases: (1) general knowledge K_G, such as law of sciences, structured patterns, summary statistics; and (2) specificatory knowledge K_S obtained through experience with specific situations, and associated with physical data points. Physical data points may consist of (a) hard data points, which are exact measurements of natural process with probability 1 (e.g., high-precision measurements $\mathbf{X}_{hard} = [\mathrm{X}_{1_h}, \mathrm{X}_{2_h} \ldots \mathrm{X}_{N_h}]$ at locations $\mathbf{S_h} = [s_{1_h}, s_{2_h} \ldots s_{N_h}]$); and (b) soft data points, that may be interval or probabilistic data due to uncertain knowledge, intuition, or uncertainty in measurements (e.g., low-precision sensor outputs $\mathbf{X}_{soft} = [\mathrm{X}_{1_l}, \mathrm{X}_{2_l} \ldots \mathrm{X}_{N_l}]$ at locations $\mathbf{S_l} = [s_{1_l}, s_{2_l} \ldots s_{N_l}]$, such that $\mathbf{X} = \{\mathbf{X}_{hard} \cup \mathbf{X}_{soft}\}$).

We estimate the realization X_E of the random variable X at location s_E, where s_E represents the location of a node at which we estimate its measurement utilizing measurements from N_h high-precision sensors and the remaining $N_l - 1$ low-precision sensor nodes, and subsequently use this estimated value to detect its drift. Henceforth, an ST map can be generated using the realizations $\mathbf{X}_{map} = [\mathrm{X}_{1_h}, \ldots .\mathrm{X}_{N_h}, \mathrm{X}_{1_l}, \ldots .\mathrm{X}_{N_{l-1}}, \mathrm{X}_E]$ at locations $\mathbf{S}_{map} = [s_{1_h}, \ldots .s_{N_h}, s_{1_l}, \ldots .s_{N_{l-1}}, s_E]$. In our work, estimation is performed at each low-precision sensor node location (i.e., at estimation location $s_E \in \mathbf{S_l}$). BME uses three stages for knowledge acquisition and processing as follows:

(i) Prior stage, which starts with basic set of assumptions and general knowledge K_G with the goal of prior information maximization.
(ii) Preposterior stage, which uses specificatory knowledge K_S, including high-precision sensor measurements (hard data) and low-precision sensor measurements (soft data).
(iii) Posterior stage, which uses the knowledge from (i) and (ii) to integrate them with the goal of posterior probability maximization.

In the prior stage, the expected information contained in the prior probability distribution function (PDF) can be expressed using a Shannon's information measure as follows:

$$\mathscr{E}(Info_\mathrm{G}[\mathbf{X}_{map}]) = -\int \Phi_\mathrm{G}(\mathbf{X}_{map}) \log \Phi_\mathrm{G}(\mathbf{X}_{map}) d\mathbf{X}_{map}, \tag{3}$$

where $\Phi_\mathrm{G}(\mathbf{X}_{map})$ is the prior PDF model, which refers to the general knowledge K_G before any specific knowledge base K_S has been taken into consideration, and $\mathscr{E}(\cdot)$ denotes the expectation operator. The shape of the prior PDF is derived by maximizing the expected information, which takes the following constraints into consideration:

$$\mathscr{E}(g_\alpha[\mathbf{X}_{map}]) = \int g_\alpha(\mathbf{X}_{map}) \Phi_\mathrm{G}(\mathbf{X}_{map}) d\mathbf{X}_{map}, \tag{4}$$

where $\alpha = 0, 1, 2, \ldots, N_c$ and g_α are known functions of $\mathbf{X}_{map}$. N_c is chosen in such a way so that the stochastic moments included with N_c involve all $p = (s, t)$ points. For example, by choosing g_α as *mean* and *covariance function* with $g_0 = 1$, the total number of required constraints N_c becomes $1 + (N + 1)(N + 4)/2$. The detail descriptions for these functions including other higher moments can be found in Serre and Christakos [46].

In our work, we use the mean $\bar{\mathrm{X}}_{map}(p)$ and the covariance function as the general knowledge. The space-time variability of $\bar{\mathrm{X}}$ can be described in terms of a centered covariance function as:

$$C_{map}(h, \tau) = \mathscr{E}[(\mathrm{X}_{map}(p) - \bar{\mathrm{X}}_{map}(p))(\mathrm{X}_{map}(p') - \bar{\mathrm{X}}_{map}(p'))] \forall p, p', \tag{5}$$

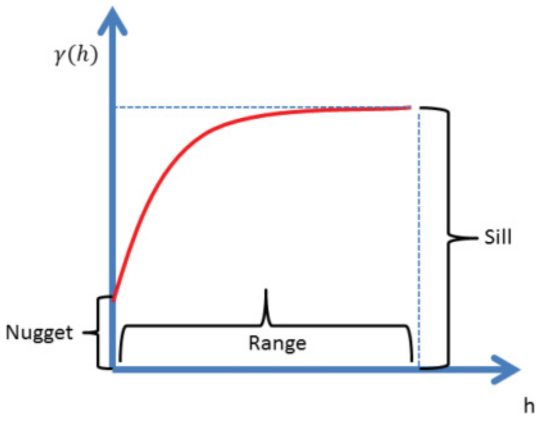


Fig. 4. Example of a semivariogram model.

where $\bar{\mathbf{X}}_{map}(p) = \mathscr{E}[\mathbf{X}_{map}(p)], \forall p$. For any two ST points p, p', $p - p' = ((h - h'), (\tau - \tau')); h = ||s - s'||; \tau = ||t - t'||$. h and τ are space and time lags that show covariance is spatially isotropic and stationary in time. Hence, in our case, g_α is adapted such that expectation $\mathscr{E}(g_0(\mathbf{X}_{map}))$ defines the ST mean and covariance function (Equation (6)) as the ST domain of interest.

Since we perform estimation and drift detection iteratively in the time domain, only the spatial covariance function is required to model the ST covariance structure, which is also known as the variogram. Variogram (or semivariogram) [14, 52] is an experimental function, which includes self-variograms and cross-variograms, and is used to determine spatial correlations in observations measured at sample locations, as shown in Equation (6).

$$\hat{\gamma}(h) = \frac{1}{2N_{sh}} \sum_{j=1}^{N_{sh}} [\mathbf{X}_{map}(\mathbf{s}_i) - \mathbf{X}_{map}(\mathbf{s}_i + h)]^2, \tag{6}$$

where $\hat{\gamma}(h)$ is the experimental semivariance at a space lag h, $\mathbf{X}_{map}(\mathbf{s}_i)$ is the value of the random variable at $\mathbf{s}_i$, and N_{sh} is the number of pairs of points separated by h. Once an experimental variogram is obtained, a model function can be fitted to obtain a semivariogram model. Examples of some semivariogram models are linear, spherical, exponential, and Gaussian models [37]. In our experiments, we use the nugget-exponential variogram, as shown in Equation (7), based on fitting the best model to the empirical variogram obtained for the datasets that we used for evaluation (see Section 7 for more details).

$$\hat{\gamma}(h) = C_0 + C_1[1 - exp(-3h/a)], \tag{7}$$

where C_0 is a constant (nugget) due to the nugget effect that raises the whole theoretical semivariogram by C_0 units, a is the distance at which samples become independent of one another, called as the range, and C_1 is the sill value of $\hat{\gamma}(h)$ at which the semivariogram graph levels off. A typical semivariogram model is shown in Figure 4 with sill, nugget, and range parameters [12].

Once the constraints in Equation (4) are formed as described earlier, then the optimization for maximizing the Shannon information in Equation (3) is done by using the Lagrange multipliers λ_α. Hence, the prior PDF can be expressed as $\Phi_G(\mathbf{X}_{map}) = H^{-1}e^{\Theta_G[\mathbf{X}_{map}]}$, where $H = e^{-\lambda_0}$ is a normalization constant, Θ_G represents the operator processing general knowledge K_G, and is given by $\sum_{\alpha=1}^{N_c} \lambda_\alpha g_\alpha(\mathbf{X}_{map})$.

Similarly, specificatory knowledge K_S is considered in the preposterior stage, and the prior PDF Φ_G is updated by means of Bayesian conditionalization. Note that the hard data may be used in both the stages. In the prior stage, they are included indirectly in the form of mean and covariance function calculations, while in the preposterior stage, they are used directly.

In this article, we use interval $I = [l, u]$ for expressing the soft data, that is, the measured values lie within lower l and upper u limits with probability 1. In our work, interval ranges are defined using measurement error/uncertainty of low-precision sensors. At the posterior stage, the updated PDF obtained is a conditional PDF and can be expressed in terms of the prior PDF and the specific knowledge considered at the preposterior stage as follows:

$$\Phi_K(X_E|\mathbf{X}_{data}) = J^{-1}\Theta_S[\Phi_G(\mathbf{X}_{map}), \mathbf{X}_{soft}], \tag{8}$$

where $\mathcal{J}$ is the normalization parameter given by

$$J = \Theta_S[\mathbf{X}_{soft}, \Phi_G(\mathbf{X}_{data})] = \int_I \Phi_G(\mathbf{X}_{data}) d\mathbf{X}_{soft},$$

and $\Theta_S[\cdot]$ represents the posterior operator that incorporates the soft data. For the interval kind of soft data

$$\Theta_S[\mathbf{X}_{soft}, \Phi_G(\mathbf{X}_{map})] = \int_I \Phi_G(\mathbf{X}_{map}) d\mathbf{X}_{soft} \tag{9}$$

$$= H^{-1} \int_I e^{\Theta_G[X_{map}]} d\mathbf{X}_{soft}. \tag{10}$$

Note that the multiple integral in Equation (9) has the form of a *multivariate Gaussian probability*, which is very useful in numerical implementations [19].

Estimate at s_E can be obtained by maximizing the posterior PDF with respect to X_E such that the BME estimate minimizes the RMSE. Maximization of the posterior PDF can be done by minimizing $-log(pdf)$. This minimization becomes an optimization problem where the objective is to find the mean estimate X_E on some fixed interval (derived from soft data) so that $-log(\Phi_K(X_E|\mathbf{X}_{data})$ is minimum. In this article, the mean estimate is used to find the realization of random variable at an arbitrary location, which is given by $X_{E,mean} = \int X_E \Phi_K(X_E) dX_E$. Next, we discuss some other generally used geospatial estimation techniques for ST analysis, which we use in our experiments for comparison purposes.

5 CLASSICAL GEOSTATISTICS-BASED ESTIMATION TECHNIQUES

In this section, we discuss the different kinds of estimation function $f(\cdot)$ that can be used to estimate the value of a random field at any location incorporating neighboring sensor measurements.

Kriging [12] is a geospatial interpolation method used to estimate the value at an unobserved location using a linear combination of observed values at neighbor locations. It belongs to a group of linear least mean square error estimation algorithms and is also known as Gaussian process regression. It uses semivariogram to minimize the spatial distribution of estimated variance.

All kriging estimators are but variants of the basic linear regression estimator $\hat{\mathbf{X}}(\mathbf{s}_E)$, defined as

$$\hat{\mathbf{X}}(\mathbf{s}_E) - m(\mathbf{s}_E) = \sum_{j=1}^{N(s_E)} w_j \times [\mathbf{X}(\mathbf{s}_j) - m(\mathbf{s}_j)], \tag{11}$$

where $N(\mathbf{s}_E)$ is the number of local neighborhood points used for estimation $\hat{\mathbf{X}}(\mathbf{s}_E)$, $\mathbf{s}_E$ and $\mathbf{s}_j$ are the location vectors of estimation point and one of the neighboring data points, indexed by j, $m(\mathbf{s}_E)$ and $m(\mathbf{s}_j)$ are the expected values (mean) of random variable at $\mathbf{s}_E$ and $\mathbf{s}_j$, and w_j is the kriging weight assigned to the sensor node j. These weights are chosen such that they minimize the variance of the estimator, as given in Equation (12).

$$\hat{\sigma}^2(\mathbf{s}_E) = Var\{\hat{\mathbf{X}}(\mathbf{s}_E) - \mathbf{X}(\mathbf{s}_E)\} \tag{12}$$

$$= \sum_{i=1}^{N(s_E)} \sum_{j=1}^{N(s_E)} w_i w_j Cov(\mathbf{s}_i, \mathbf{s}_j) + Cov(0) - 2 \sum_{i=1}^{N(s_E)} w_i Cov(\mathbf{s}_i, \mathbf{s}_E), \tag{13}$$

under the unbiasedness constraint,

$$E\{\mathbf{X}(\mathbf{s}_E) - \mathbf{X}(\mathbf{s}_j)\} = 0, \tag{14}$$

where $\mathbf{X}(\mathbf{s}_E)$ is the actual value of the variable at location $\mathbf{s}_E$, and $Cov(\mathbf{s}_i, \mathbf{s}_j)$ is the covariance function of $\mathbf{s}_i$ and $\mathbf{s}_j$, derived from the input semivariogram model.

$\mathbf{X}(\mathbf{s_E})$ is considered as a random field, which is decomposed into two components, named *trend* $m(\mathbf{s_E})$ and *residual* $R(\mathbf{s_E})$ such that $\mathbf{X}(\mathbf{s_E}) = m(\mathbf{s_E}) + R(\mathbf{s_E})$. $R(\mathbf{s_E})$ is considered a random field with zero mean and stationary covariance function (function of lag, h, but not of the locations, $\mathbf{s}_j$). Kriging estimates residual at $\mathbf{s_E}$ as a weighted sum of residuals at surrounding data points. The trend component $m(\mathbf{s_E})$ describes the stochastic properties of underlying random field. The main kriging variants differ in their treatment of the trend component $m(\mathbf{s_E})$. A few of them, namely simple kriging, ordinary kriging, and blind kriging, which are used in our experiments for comparison, are described in the following text.

Simple Kriging

In simple kriging [37], it is assumed that the trend component is constant and have unknown mean, $m(\mathbf{s}_1) = \ldots. = m(\mathbf{s}_E) = m$

$$\hat{\mathbf{X}}(\mathbf{s}_E) = m - \sum_{j=1}^{N} w_j * [\mathbf{X}(\mathbf{s}_j) - m] \tag{15}$$

This estimate is automatically unbiased, since $E\{\mathbf{X}(\mathbf{s}_E) - m\} = 0$.

Ordinary Kriging

For ordinary kriging [18], it is assumed that the mean is constant in the local neighborhood of each estimation point rather than assuming it to be constant over the entire region. The mean for ordinary kriging is given by $m(\mathbf{s}_j) = m(\mathbf{s})$ for each local data value, and the kriging estimation can be written as

$$\hat{\mathbf{X}}(\mathbf{s}_E) = m(\mathbf{s}) - \sum_{j=1}^{N} w_j \times [\mathbf{X}(\mathbf{s}_j) - m(\mathbf{s})] \tag{16}$$

Blind Kriging

Blind kriging [30] is the form of *universal kriging* that has an unknown trend model. Therefore, it is known as blind kriging. The trend model is assumed as multivariate polynomial given by

$$m(\mathbf{s}) = \sum_{j=1}^{N} v_j(s_j), \tag{17}$$

where $v_j(s_j)$ are functions of sensor locations, and identified from experimental data using Bayesian variable selection technique to minimize the RMS error of the predicted value at sample points.

6 DRIFT DETECTION AND CORRECTION USING KALMAN FILTERING

Suppose that the underlying process $y \in R^v$ can be represented by the following linear discrete time model

$$y_t = Ay_{t-1} + Bu_t + v_t, \quad v_t \sim N(0, Q), \tag{18}$$

with a measurement $z \in R^{\gamma}$ given by

$$z_t = Cy_t + w_t, \quad w_t \sim N(0, R), \tag{19}$$

where y_t is the process state at time t, u_t represents the control input, and z_t is the output at time t. A, B, and C represent the state transition matrix, control input model, and measurement model, respectively. The random variables v_t and w_t represent the process and measurement noise, respectively, which are assumed to be zero mean white Gaussian noise. Q and R are the process and the measurement noise covariance matrix, respectively. Given the knowledge of the process y prior to step at t and measurement z_t, *a priori* and *a posteriori* state estimate at step t is defined as $\hat{y}_{\bar{t}} \in R^v$ and $\hat{y},_t$ respectively. The *a priori* estimate error $e_{\bar{t}}$, *a posteriori* estimate error e_t, the *a priori* estimate error covariance $P_{\bar{t}}$, and the *a posteriori* estimate error covariance P_t are defined as

$$e_{\bar{t}} \equiv y_t - \hat{y}_{\bar{t}}, \tag{20}$$

$$e_t \equiv y_t - \hat{y}_t, \tag{21}$$

$$P_{\bar{t}} = E[e_{\bar{t}} e_{\bar{t}}^{\mathsf{T}}], \tag{22}$$

$$P_t = E[e_t e_t^{\mathsf{T}}] \tag{23}$$

For the KF, the *a posteriori* state estimate $\hat{y}_t$ can be presented as a linear combination of the *a priori* estimate $\hat{y}_{\bar{t}}$ and a weighted difference between the actual measurement z_t and the predicted measurement $C\hat{y}_{\bar{t}}$

$$\hat{y}_t = \hat{y}_{\bar{t}} + K(z_t - C\hat{y}_{\bar{t}}), \tag{24}$$

where the difference $z_t - C\hat{y}_{\bar{t}}$ is called the measurement residual. K is the KF gain, which is chosen in such a way that it minimizes the *a posteriori* error covariance, P_t, and given by

$$K_t = P_{\bar{t}} C^{\mathsf{T}} (C P_{\bar{t}} C^{\mathsf{T}} + R)^{-1} \tag{25}$$

The KF is a recursive estimator, which works in two-step process: (1) the predict and (2) the update step. The predict step incorporates the state estimate from the previous timestep and produces an estimate (*a posteriori* estimate) of the state at the current timestep. This predicted estimate is also known as the *a priori* state estimate because it does not include observation information from the current timestep; however, it is an estimate of the state at current timestep. The update or correct step is responsible for the feedback, for incorporating the current *a priori* prediction into the current observation information to obtain improved *a posteriori* estimate. The final estimation algorithm resembles that of a predictor-update algorithm for solving numerical problems. It can run in real time, using only the present input measurements and the previously calculated state and its uncertainty matrix without any need of any additional past information.

The specific predict equations are given by:

$$\hat{y}_t = A\hat{y}_{t-1} + Bu_t, \tag{26}$$

$$P_{\bar{t}} = AP_{t\bar{-}1}A^{\mathsf{T}} + Q, \tag{27}$$

and the update equations are given by:

$$K_t = P_{\bar{t}} C^{\mathsf{T}} (C P_{\bar{t}} C^{\mathsf{T}} + R)^{-1}, \tag{28}$$

$$\hat{y}_t = \hat{y}_t + K_t(z_t - C\hat{y}_t), \tag{29}$$

$$P_t = (I - K_t C) P_{\bar{t}} \tag{30}$$

Drift detection using Kalman filtering is an iterative process. At timestep t, a reading $r_{s,t}$ is made by a node at location s. Each node is aware of its drift $\hat{d}_{s,t}$, estimated at timestep $t-1$. Using this drift, the node corrects its measured reading to obtain a corrected reading $x_{s,t}$ and sends it to its neighbor nodes. Each node collects the corrected reading sent by its neighbor nodes and estimates the value $\hat{x}_{s,t}$ at its location using the BME estimation technique. Based on this value, it modifies the estimate of drift at time instant t.

Assuming that the drift is slow and smooth, it is modeled by the following mathematical model (with $A = 1$ and $B = 0$, as in [49]),

$$d_{s,t} = d_{s,t-1} + v_{s,t}, \quad v_{s,t} \sim N(0, Q_{s,t}). \tag{31}$$

Here, $v_{s,t}$ is the Gaussian noise having $Q_{s,t}$ as the state covariance matrix (process noise variance). The measurements $z_{s,t}$ are taken to be

$$z_{s,t} = r_{s,t} - \hat{x}_{s,t}, \tag{32}$$

where $\hat{x}_{s,t}$ is the estimated temperature value computed by the node at location s. As there is a noise (random error) associated with the measurement at each node, we can write

$$r_{s,t} = T_{s,t} + d_{s,t} + w_{s,t}, \quad w_{s,t} \sim N(0, R_{s,t}). \tag{33}$$

Here $T_{s,t}$ is ground-truth value of temperature and $w_{s,t}$ is the Gaussian noise having $R_{s,t}$ as the state covariance matrix (measurement noise). Equations (32) and (33) form a KF predict and update equations (with $C = 1$ in Equations (18) and (19)) and its solution as per Takruri et al. [49] is given by

$$\hat{d}_{s,t} = \hat{d}_{s,t-1} + \frac{P_{s,t-1} + Q_{s,t}}{P_{s,t-1} + Q_{s,t} + R_{s,t}} (z_{s,t} - \hat{d}_{s,t-1}), \tag{34}$$

$$P_{s,t} = (P_{s,t-1} + Q_{s,t}) \left(1 - \frac{P_{s,t-1} + Q_{s,t}}{P_{s,t-1} + Q_{s,t} + R_{s,t}}\right). \tag{35}$$

7 CENTRALIZED BME APPROACH FOR DRIFT DETECTION AND CORRECTION

In this section, we evaluate the performance of our algorithm, which uses the BME and Kalman filtering mechanism for sensor drift detection and correction in WSNs. Furthermore, we discuss the performance of our approach for different values of parameters, such as the number of drifting sensors, the number of high- and low-precision sensors, and the measurement error (interval) value of low-precision sensors.

We used two real datasets, consisting of sensor measurements collected from an indoor and an outdoor deployments of WSNs, for our evaluations. We also implemented our algorithm on real sensor nodes (hardware) and deployed them in the lab to demonstrate its applicability in real-life situations (see Section 9). The first dataset used in our evaluation is a set of real sensor measurements collected from a deployment of a WSN in the Intel Research Berkeley Laboratory (IBRL) [16]. This is a publicly available WSN dataset consisting of about 2.5 million readings collected between February 28th and April 5th, 2004. The sensor network consists of 54 sensor nodes, as shown in Figure 5, and has collected humidity, temperature, luminosity, and battery level measurements at a sampling interval of 31 seconds. In the following, we first discuss the data preprocessing performed before we used this data in the BME framework.

7.1 Data Preprocessing

As both the temperature and humidity are slowly varying phenomenon, we took samples at 10-minute intervals by averaging the data available within the 10-minute window. This averaging also eliminates the discrepancy in the time information of samples due to synchronization error.

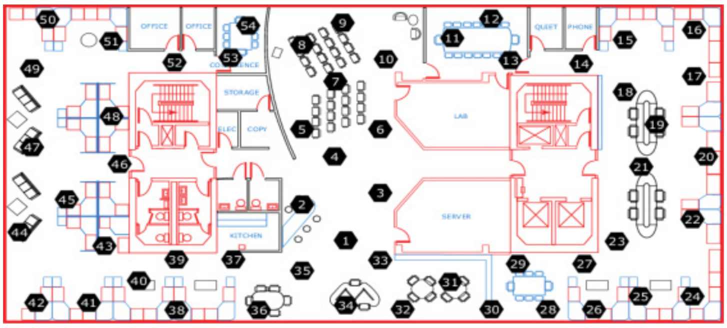


Fig. 5. Sensors nodes in the IBRL deployment. Nodes are shown in the black with their corresponding Node IDs.

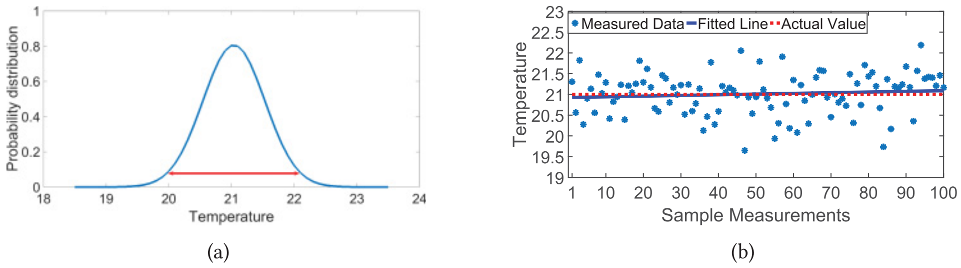


Fig. 6. Methods of computing interval range of sensor measurements. (a) Gaussian PDF-based method. (b) Regression-based method.

We discarded the data from sensor node IDs 5, 15, and 18, as their measurements are too noisy. We considered 1, 200 data points for each node in our experiment.

7.1.1 Preparation of Hard and Soft Data for BME. In our experiments, we use temperature measurements from all the sensors; however, our approach is equally applicable to any other gradually changing phenomenon. Since we do not have the classification as hard- and soft-precision sensors for this dataset, we considered 10 sensors nodes distributed evenly over the spatial locations as hard sensors, and the remaining as soft sensors. The spatial arrangements of the hard (shown as blue *s) and soft sensors (shown as red +s) for IBRL WSN deployment is shown in Figure 7. For the 10 sensors considered as high-precision sensors (hard), we assumed that there is no drift or they are calibrated periodically to prevent any drift occurrences. The other 41 sensors (shown as red +s in Figure 7) are considered soft sensors (low-precision sensors). We preprocessed the measurements of those 41 soft sensors in such a way that their measurements lie within some interval $I = [l, u]$, that is, the measured values lie within the lower l and upper u limits with probability 1. A larger interval value corresponds to a higher measurement error (i.e., less precise measurement). One way to obtain the measurement interval in practice is to take multiple measurements at any location and then fit a Gaussian distribution to them. The spread (standard deviation) of the distribution can be used as an estimate for the interval, as shown in Figure 6(a). Another indirect way is to fit the regression model between the actual field value measured by top-end high-precision equipment and the low-precision sensor value at that site, as shown in Figure 6(b). First, we can calculate the variance of residuals σ_ϵ^2 and then a linearly estimated value (*LEV*) from the best-fitting regression model can be found. The lower and upper bound can be calculated by following

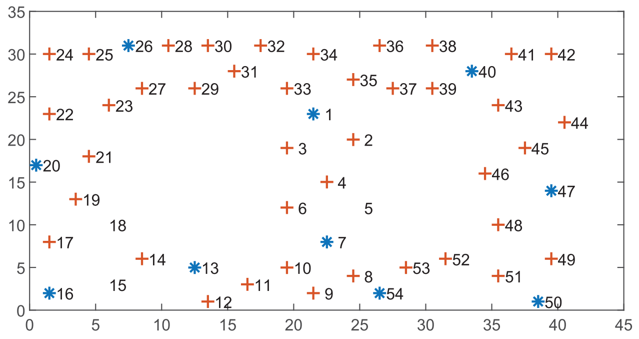


Fig. 7. High-precision (hard) and low-precision (soft) sensors nodes. Nodes IDs with * represent hard sensor nodes and Node IDs with + represent soft sensor nodes.

equations.

$$value_{lower} = LEV - \sigma_\epsilon, \quad value_{upper} = LEV + \sigma_\epsilon \tag{36}$$

As we are using the repository data (IBRL), for which we do not have the information about measurement errors, we used the following approach to create the soft data for our experiment. We added some uncertainty (uniformly distributed (u.d.)) in the original measurements to make them as low-precision measurements. These levels are determined by the widths of the interval $I_i = [\mathrm{x} - u\delta(s_i), \mathrm{x} + u\delta(s_i)]$, where $u \in (0,1)$ is a random number, i is the soft sensor ID, and the $\delta(s_i)$ are selected as follows:

$$\delta(s_i) = \begin{cases} 0.4 & \text{for 8 random sensor nodes} \\ 0.6 & \text{for next 8 random sensor nodes} \\ 0.7 & \text{for next 8 random sensor nodes} \\ 0.9 & \text{for next 8 random sensor nodes} \\ 1.0 & \text{for the remaining 9 random sensor nodes.} \end{cases}$$

Initially, we performed our experiments for the same interval level of 0.5 for each of the sensors assuming that they are all manufactured by the same process. Later on, we also performed our experiments by changing the interval values in two ways: (a) same interval value for each sensor and (b) different interval values for different sensors.

7.1.2 Mean and Covariance Function for General Knowledge. In the BME-based drift detection approach, we used the mean and covariance functions to define the general or prior knowledge. The mean is computed assuming a uniform distribution. Linear regression is used to compute the mean for hard data, and an average of interval values is used to compute the mean for soft data. The next step is to compute the experimental variogram in order to obtain the covariance estimates at sample locations. It includes fitting a wide sense stationary, spatially isotropic, ST covariance model to the real data in order to compute the variogram of the sensor measurements. Since we are tracking the drift iteratively in time domain, we used the spatial covariance model to fit the data. The basic steps involved in this process are as follows:

(1) In the first step, variogram, viz., covariance estimates at different sample locations are calculated from the data using Equation (6).
(2) The next step (optional) is to interpolate the variogram onto a uniform grid.
(3) The final step is to choose a suitable variogram model (exponential, spherical, Gaussian, etc.) to obtain a best fit for our empirical variogram.

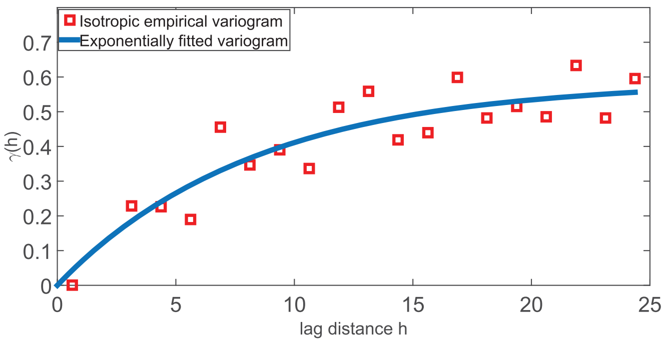


Fig. 8. Empirical (in red square) and fitted exponential spatial variogram (blue line) for IBRL data.

Based on best fitting, for the IBRL dataset, we obtained the nugget value as −0.0795, sill as 0.6656, and the spatial range value as 22.35 for our nugget-exponential covariance model, given in Equation (7). Figure 8 shows the actual and the fitted spatial variogram for the IBRL data.

7.2 Evaluation of Our Algorithm for Drift Detection

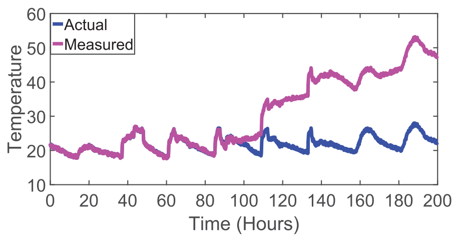
We first discuss the performance of our approach for multiple drifting sensors and then compare our results with the kriging-based estimation techniques. Subsequently, we discuss how the different number of hard and soft sensors, the different number of drifting sensors, and the different interval values affect the performance of the BME algorithm for drift detection.

The BME and Kalman filtering-based centralized method for drift detection was implemented in MATLAB, utilizing the BME spatial estimation library [11]. The parameters, $Q_{s,t}$ and $R_{s,t}$, of the KF are tuned using a grid search method over the parameter space, and the values used in our evaluation are $Q_{s,t} = 1$ and $R_{s,t} = 0.0001$. The parameterd $Q_{s,t}$ and $R_{s,t}$ were same for all sensors and all t. For a lower value of $R_{s,t}$, estimated drift starts to follow the reading as it starts to trust the reading more than the estimation value. A high value of $Q_{s,t}$ makes estimation oscillatory and unstable. Hence, a compromise has to be met in Q and R value selection to obtain the best drift estimates.

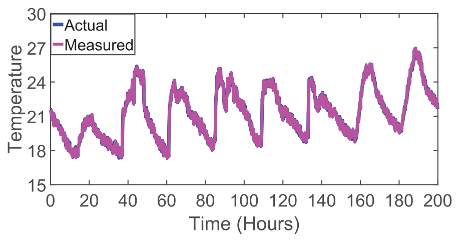
We introduced synthetic drifts in the sensor measurements in order to evaluate the capability of our algorithm to detect and correct them. Among the 51 working sensor nodes, we considered 10 sensors as the hard sensors; thus, no drift was incorporated into their measurements. A drift having a process noise variance of 1 ($Q_{s,t} = 1$, refer to Equation (31)) was added to 30 randomly chosen sensors out of 41 remaining (low-precision) sensors (soft sensors) that were obtained after preprocessing, as discussed in Section 7.1.1. The drift model in Equation (31) justifies our initial smooth drift hypothesis. The instantiation time of drift was chosen randomly for each sensors. To account for measurement noise, a Gaussian noise of zero mean and variance 0.1 was added to the measured values of each hard and soft sensors. Figure 9 shows an example of a drifting and a nondrifting sensor measurements.

In this experiment, each sensor uses a maximum of 3 nearest hard sensors out of the 10 high-precision sensors, and 3 nearest soft sensors from the 41 low-precision sensors in order to estimate the temperature value at its location. Figure 10 shows the comparison of the actual drift and the estimated drift for temperature value using the BME method. It can be observed that the BME-based drift detection algorithm is able to trace the actual sensor drift with high accuracy.

Figure 11 shows the performance of the BME method in estimating and correcting drift for the temperature sensor measurements. It can be inferred from the figure that the sensor measured

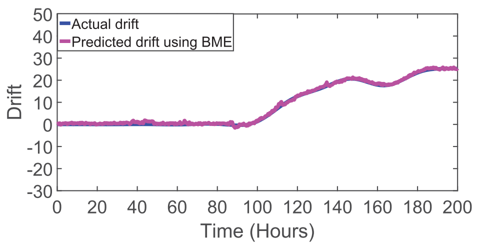


(a)

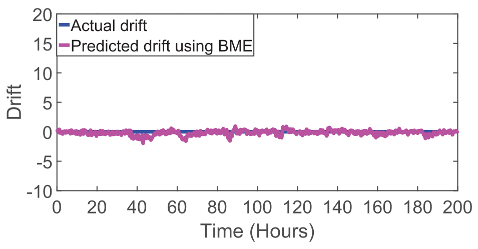


(b)

Fig. 9. Actual and measured temperature values for (a) drifting sensor, and (b) nondrifting sensor.



(a)



(b)

Fig. 10. Actual and estimated drift for (a) drifting sensor and (b) sensors having no drift.

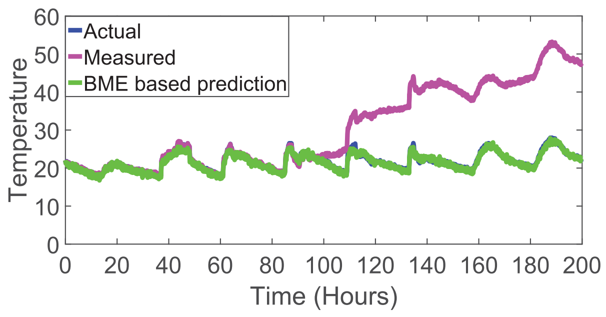
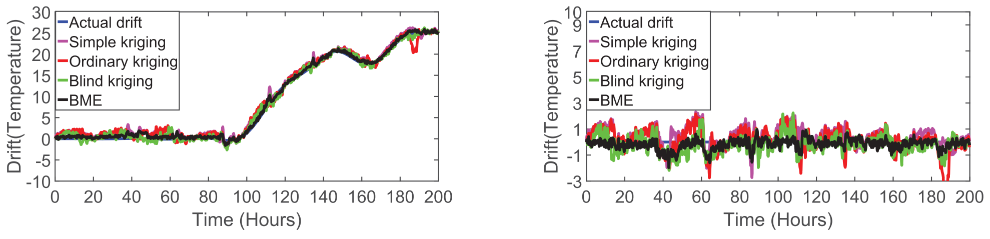


Fig. 11. Actual, measured, and corrected temperature values.

value is quite different from the actual value (ground-truth value) due to drift, but the corrected drift values are much closer to the ground-truth values.

7.2.1 Comparison of Kriging Variants and the BME-Based Algorithm. We compared several kriging-based estimation methods such as simple kriging, ordinary kriging, and blind kriging, with the BME for drift detection and correction. In this experiment, we use the five nearest hard sensors and the three nearest soft sensors for estimation. As kriging-based variants do not allow us to incorporate interval kind of data, we considered those eight sensor measurements as hard data. Note that the kriging becomes a special case of the BME when only the hard data is considered along with the mean and covariance functions as prior knowledge.

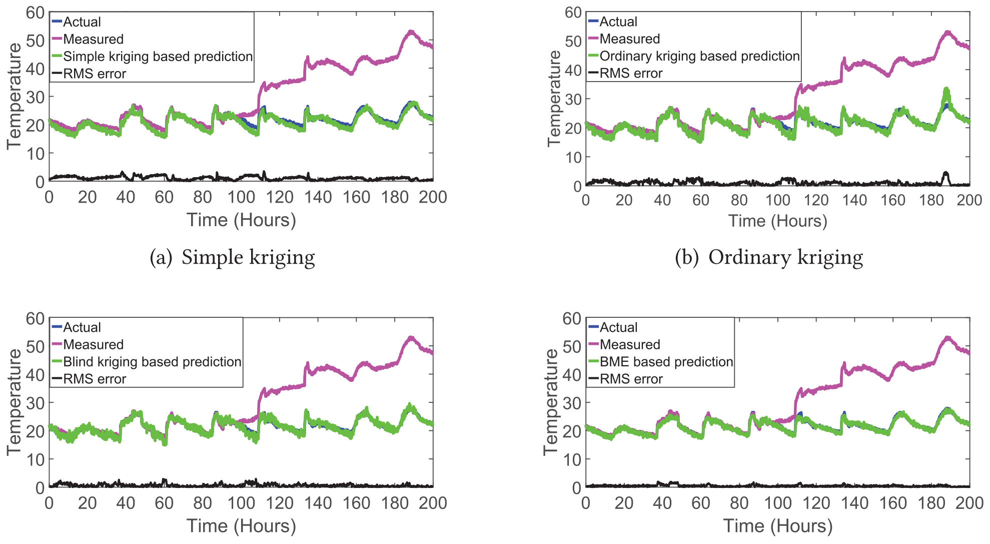
Figure 12 shows a comparison of the actual drift and the estimated drift for temperature sensors. All the estimation algorithms are able to trace the drift with low RMSE. Moreover, the BME



(a) Drifting sensor

(b) Sensor with no drift

Fig. 12. Actual and estimated drift for temperature sensor.



(a) Simple kriging

(b) Ordinary kriging

(c) Blind kriging

(d) BME

Fig. 13. Actual, measured, and corrected temperature values with RMS error for each estimation method.

shows negligible drift in the nondrifting sensors, while all the other estimation schemes resulted in leaving a small drift (residue) in the nondrifting sensors, which are undesirable.

Figure 13 shows the performance of various spatial estimation methods in estimating and correcting drift for temperature sensors. It can be noted that the measured temperature is different than the actual ground-truth value due to sensor drift, but the corrected temperature values are very close to the actual values.

Figure 14 shows the comparison of all the estimation algorithms for all the sensor nodes based on the average RMSE over the entire duration of the experiment. It is clear that the BME-based drift detection algorithm provides the best estimate of the sensor drift as it has the lowest RMSE for almost all of the sensor nodes. Remarkably, this is true even when the kriging-based method is allowed to use all the hard data available, while BME is restricted to using only a few hard data points [47]. The main reason for the better result using the BME is that, unlike all the other kriging-based (and its variants) interpolation methods, it incorporates low-precision data (soft data) and prior information in terms of uncertainty along with the high-precision sensors output (hard data) and solves a nonlinear function of data to maximize the posterior probability distributions.

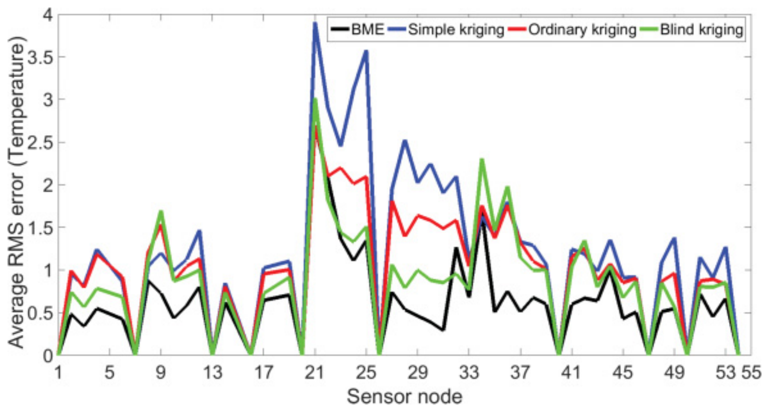


Fig. 14. Average RMSRe error of temperature for different spatial interpolation schemes.

Table 1. Average RMSE for BME-Based and Kriging- Based Schemes on the IBRL Dataset

Method	Average RMS error (Temperature)
Simple kriging	1.24
Ordinary kriging	1.08
Blind kriging	0.98
BME	0.81

Traditional kriging estimators are restricted to linear combinations of the hard data and do not provide a systematic mechanism to incorporate low-precision data and prior information.

The average RMSEs incurred for the BME- and kriging-based methods, for the entire duration of experiment, is shown in Table 1. Among the kriging-based estimation methods, blind kriging works better than simple and ordinary kriging. One point to note is that though the BME-based method outperforms the other kriging-based schemes, the difference between the average RMSE is small. The main reason was that this experiment was conducted in a controlled environment (office); hence, the readings of all the sensors are pretty close to each other, which do not cause much differences in the performances of both the methods. For the outdoor deployment, where a higher variance between the measurements of each sensor is expected, the BME-based algorithm is expected to perform much better than the kriging-based method. In Section 9, we discuss the results from the outdoor deployment data. Next, we analyze the complexity of our algorithm in detail.

7.2.2 Computational Complexity. In this experiment, we compare the BME-based and the kriging-based estimation algorithms for drift detection on the basis of their computational complexity. In the centralized mode, the central node performs the spatial estimation and the Kalman filtering after collecting the data from all the nodes in the network of size N. In this way, a communication overhead of $O(N^2)$ and a computational overhead of $O(N^3)$ will be incurred for the kriging operation, and the Kalman filtering operation can be performed in linear time. Kriging has high computational complexity, which limits its use for small training datasets. A trade-off between the computational complexity and the optimality of solutions needs to be met when using kriging-based spatial interpolation methods in practice [23, 39]. In ordinary kriging, the mean function, which is an unknown function of linear combinations of its neighbor's sample,

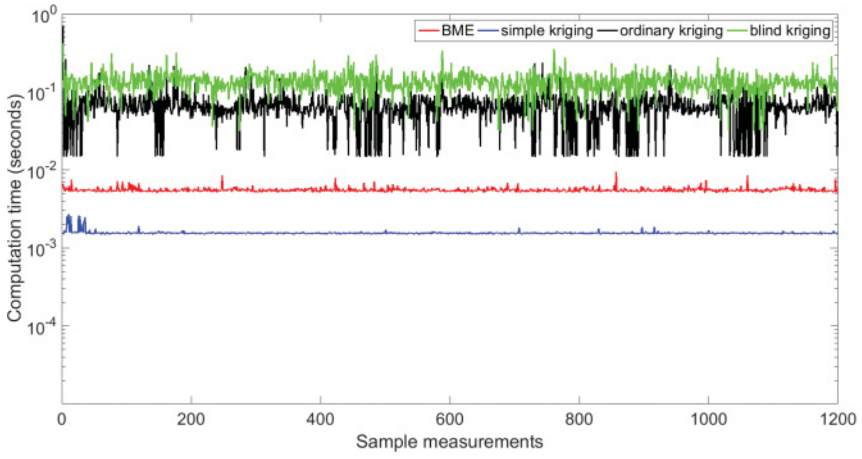


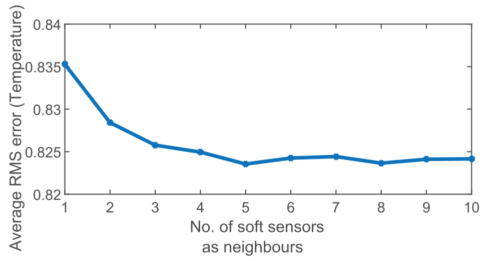
Fig. 15. Time complexity of different spatial estimation schemes.

is computed using a regression method, thus incurring additional computational cost. In blind kriging, the mean function is formed as a combination of known functions, but their weights (v_j in Equation (17)) are computed iteratively, resulting in incurring approximately double the computational cost of the ordinary kriging method.

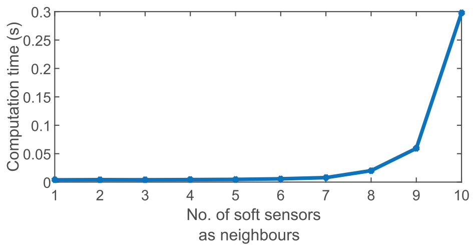
The BME approach improves the estimation accuracy at the cost of computing multiple integrals for calculating multivariate Gaussian PDF. In BME, first, the regression is performed in $O(N^3)$ to obtain a primary estimate from hard data and soft data (low-precision sensor measurements). Second, a multivariate cumulative distribution function (cdf) calculation is performed with a time complexity of $O(2^k(k-1)!)$ [21] for k random variables (the number of soft sensors considered in estimation). The third time-consuming part is the process of finding the minima of a single-valued function within a range, defined by the lower and upper limits corresponding to the lower and upper bounds of low-precision sensors' measurements, which uses a golden section search method with successive parabolic interpolation [9] algorithm. The computational complexity to compute ϵ-accurate solution at a linear rate is given by $O(log(1/\epsilon))$ [7].

In all the methods, each (central) node collects the readings from its neighboring nodes (of size $n << N$, for a network of size N) and performs the estimation and Kalman filtering process for drift detection. In the decentralized development, where each node keeps track of its drift and corrects it on its own, there would be an additional communication overhead of $O(n^2)$ incurred as each node has to transmit its measurement to its neighboring nodes. As only $n << N$ number of node measurements will be used for estimation, the computational complexity is reduced significantly. Furthermore, the environmental physical variables do not change rapidly, thus reducing the need for very high sampling rate, and hence will not affect the performance of our scheme in real-time environmental applications.

Figure 15 shows the time taken (log scale) for different spatial estimation schemes. Spatial interpolation for BME was performed using the BME library from Refs. [11] and [35], and the kriging and its variants were performed using the DACE toolkit [36] and ooDACE toolkit [13] implemented in MATLAB. The computational platform used is a PC with the following configurations; OS: Windows 7 (64 bit); processor: Intel(R) Core(TM) i7-4770 @3.40GHz; RAM: 16GB. It is clear from Figure 15 that the simple kriging algorithm takes less time than the BME and the other kriging-based methods, while BME being the second least time-consuming method among the

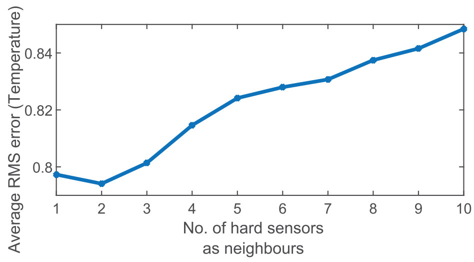


(a) Average RMS error

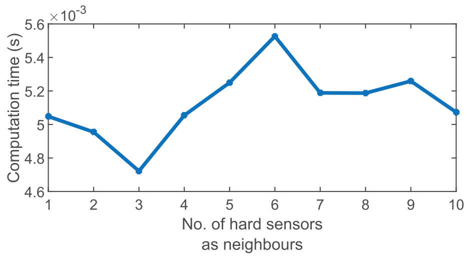


(b) Computation time

Fig. 16. Average RMSE and computation time for different numbers of neighboring soft sensors.



(a) Average RMS error



(b) Computation time

Fig. 17. Average RMSE and computation time for different number of neighboring hard sensors.

other four estimation techniques. Although the BME is a little computational-intensive algorithm compared to the simple kriging, the computational complexity can be reduced by converting the multivariate integral computation to bivariate or trivariate ones. This will make the BME a bit more computationally efficient approach comparable to the simple kriging.

7.2.3 Performance of the BME-Based Method for Different Numbers of Hard and Soft Sensors. In this experiment, we evaluate the performance of the BME-based drift detection algorithm using different numbers of hard N_{nh} and soft N_{nl} sensors as nearest neighbors. We keep the number of drifting sensors at 30 and the interval value at 0.5 for low-precision sensors, similar to the last experiment. We compute the RMSE and the computation time of the BME-based method for different number of neighboring hard and soft sensors. Figure 16 shows the RMSE and the computation time of the BME algorithm for different numberS of soft sensors when the number of hard sensors is fixed to three (i.e., $N_{nh} = 3$). It can be inferred from the figure that the RMSE decreases as the number of soft sensor increases from 1 to 10, while the computation time increases. This is due to the fact that the BME framework maximizes the posterior PDF by utilizing a larger amount of interval-type data inorder to provide a robust estimate; however, the computation time also increases due to the calculation of multiple integrals involved in computing the multivariate Gaussian probabilities.

In our next experiment, we vary the number of nearest hard sensors in the BME estimation, from 1 to 10 for $N_{nl} = 5$. Figure 17(a) shows the overall RMSE, which first decreases from 1 to 2 (nearest hard sensors) and then increases linearly. This result is interesting because the hard sensors have very low spatial resolution (spread far apart) as compared to the soft sensors. Due to this fact, only a few (in this case two) nearest high-precision sensor measurements are highly correlated for each sensor node. As the nearest number of hard sensors increases, the less correlated values (from the distant hard sensors) also become accounted for estimation, which increases the estimation

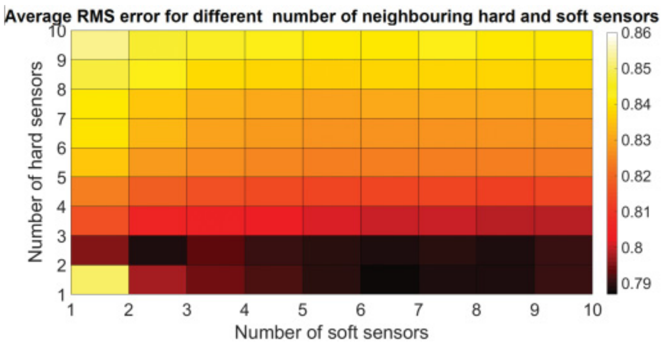


Fig. 18. The colormap of the average RMSE for different numbers of neighboring hard and soft sensor nodes. The average RMSE is mapped from dark/black to light/yellow color based on its minimum and the maximum value, respectively, in the overall map.

error. The computation time is almost the same for different number of hard sensors, as shown in Figure 17(b).

Figure 18 shows the colormap matrix of the average RMSE for different numbers of neighboring hard and soft sensor nodes, both varying from 1 to 10. It can be inferred from the figure that for each value of the number of hard sensors, the average RMSE decreases as the number of soft sensors increases, while for each value of the number of soft sensors, the average RMSE increases as the number of hard sensors increases. $N_{nh} = 2$ provides the lowest RMS values for each value of the number of soft sensors. This is because of the same reason as we discussed in the last experiment. As we have seen in the previous experiments, the computation time increases as the number of soft sensors increases. This reveals that the value of N_{nh} and N_{nl} can be chosen in such a way that a desired estimation accuracy can be achieved with less computation time. It is interesting to see that even by using two hard and two soft sensors, the BME algorithm is able to detect and trace the drift accurately.

7.2.4 Performance of BME for Different Numbers of Drifting Sensor. Generally, in practical WSNs, it is very rare that majority of the sensor nodes drift during the same time period. However, we are interested in evaluating the suitability of our BME algorithm when more than 50% of the total sensor nodes are drifting during the same time period. In this experiment, we evaluate the performance of our BME-based drift detection algorithm for different numbers of drifting sensors varying from 2 to 40 (almost all low-precision sensors nodes). Based on our last experiments, we use the optimal value of N_{nh} and N_{nl} as 2.

Figure 19 shows the increasing pattern of the average RMSE as the number of drifting sensors in the WSN increases. However, it does not significantly increase and still gives a lower RMSE (maximum 0.81) than the other estimation schemes, as shown in Table 1. Hence, BME performs well even if around half of the sensor nodes are drifting in a WSN; however, note that the performance may become worse for any node if all of its neighboring soft sensors are drifting and it does not have any hard sensors in its neighborhood.

7.2.5 Performance of BME for Different Interval Values of Low-Precision Sensors. Low-cost sensors may have different measurement errors/uncertainties, which generally depend on their manufacturing process. Generally, measurement errors would be same for all sensors which are drawn from the same wafer or manufacturing process. Therefore, in the case of a small-scale WSN deployment, most of the low-precision sensors might have the same precision levels; however, for

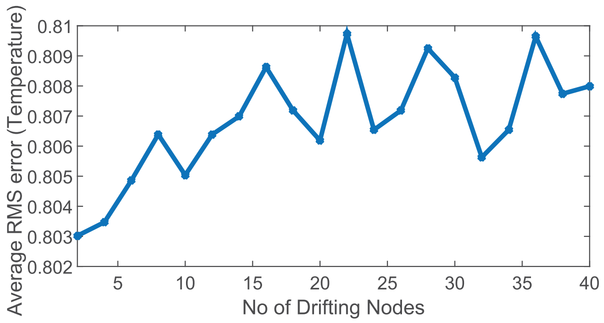
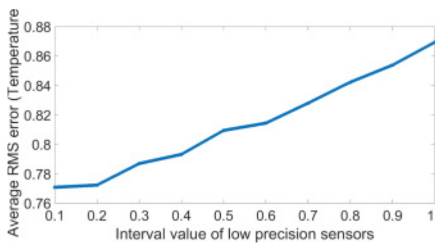
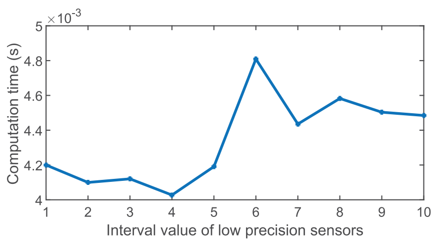


Fig. 19. Average RMSE for different numbers of neighboring hard and soft sensor nodes.



(a)



(b)

Fig. 20. Average RMSE and computational time for different intervals of low-precision sensors.

a large-scale WSN, they may have different precision levels. These measurement errors/precision levels are represented using an interval of measurements for the low-precision sensors. We performed two experiments for different intervals: (1) all sensors having the same measurement errors and (2) sensors having different measurement errors within the WSN.

In our first experiment, interval values for all the low-precision sensors are changed in the range from 0.1 to 1 in steps of 0.1. For each of these intervals, the overall RMSE is computed across all the estimation locations, by using measurement at all the time instances. Figure 20(a) shows the average RMSE, which increases linearly for increasing interval lengths of low-precision sensors. This is due to the fact that these intervals correspond to measurement errors of sensors; hence, the accuracy of the estimation is affected by the large measurement errors of all the low precision sensors. As expected, in Figure 20(b), the computation time increases as we increase the interval lengths. This is because of the multiple integration operations involved in computing the multivariate Gaussian probabilities, which need to be calculated on a large upper and lower limits corresponding to the large intervals.

In our second experiment, we used different interval values for different low-precision sensor nodes ranging from 0.4 to 1, as given in Section 7.1.1. This experimental study may help in sensor selection for WSN deployment, for example, how many low-cost sensors and with what measurement errors can be used so that a desired estimation accuracy can be achieved for the overall ST map estimation. Figure 21 shows the RMSE for five sensor nodes having different interval values. As expected, the RMSE is high for low-precision sensor having large interval range and vice versa. However, the average RMSE over all the sensor nodes remains almost similar to the previous experimental results.

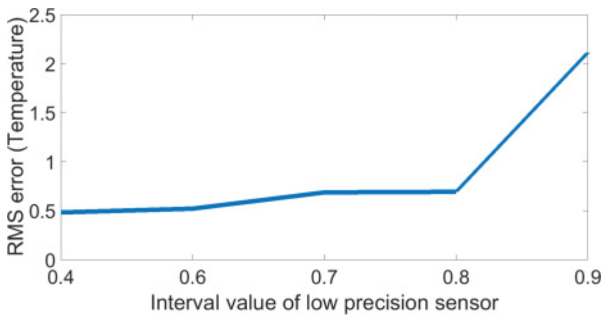


Fig. 21. RMSE for low-precision sensors with different intervals.

8 MULTIVARIATE BME FOR DRIFT DETECTION

The BME framework allows us to include additional features (i.e., realization of more than one parameter at any locations) in order to provide an improved quality of estimates of the principal feature. We take advantage of this property of the BME algorithm to incorporate the additional feature (e.g., humidity) with the principal feature (e.g., temperature) as an auxiliary information at each location in order to improve the estimates of the principal feature. Suppose that we have deployed a WSN comprising few hundreds of low-precision nodes in different locations in a geographical area of interest. Each node is measuring humidity and temperature at its location. Each node is predicting its temperature value by using temperature measurements of neighboring nodes. The accuracy of estimated value can be improved by using the humidity values of the neighboring nodes in the BME framework. Note that it is not compulsory to have high-precision humidity measurements at hard sensor locations for this scheme.

In this experiment, we use $1,200$ data vectors of humidity and temperature values of the IBRL datasets that we obtained following the preprocessing steps as discussed in Section 7.1.1. We considered temperature and humidity values of 10 hard sensors as high-precision sensor measurements, and both the feature values (temperature and humidity) at the other 41 soft sensor locations as low-precision measurements. The intervals of the humidity and temperature values were obtained using the procedure discussed in Section 7.1.1. As the prior knowledge in BME, we use the mean and covariance functions. In this experiment, we compute self- and cross-variograms by using the humidity and temperature information at each sensor location. Cross-variogram shows the relationship between two variables and is defined as follows

$$\hat{\gamma}_{xy}(h) = \frac{1}{2N_{sh}} \sum_{j=1}^{N_{sh}} [\mathbf{X}_{map}(\mathbf{s}_i) - \mathbf{X}_{map}(\mathbf{s}_i + h][\mathbf{Y}_{map}(\mathbf{s}_i) - \mathbf{Y}_{map}(\mathbf{s}_i + h)], \tag{37}$$

where $\hat{\gamma}_{xy}(h)$ is the empirical cross-variogram at a sample location h, $\mathbf{X}_{map}(\mathbf{s}_i)$, and $\mathbf{Y}_{map}(\mathbf{s}_i)$ is the principal feature (temperature) and the auxiliary feature (humidity) values, respectively, at the ith location, and N_{sh} is the number of pairs of points separated by a distance h. In our experiment, cross-variograms are computed using a nested exponential covariance model, which has a nugget effect. Figure 22 shows the self- and cross-variogram models for the temperature and humidity feature. The range of influence in cross-variogram is obtained as 22.35, while the nugget and sill parameters are defined using a 2×2 square matrix whose values obtained in our experiments are as follows:

$$Nugget_{2\times 2} = \begin{bmatrix} 0.0009 & -0.0096 \\ -0.0096 & 0.7265 \end{bmatrix} \quad Sill_{2\times 2} = \begin{bmatrix} 0.5668 & -1.0739 \\ -1.0739 & 2.0467 \end{bmatrix}$$

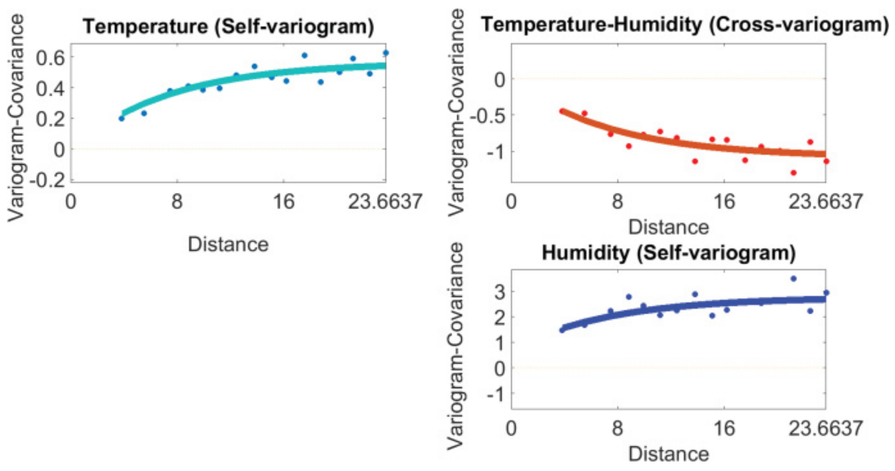


Fig. 22. Empirical and fitted exponential spatial self-variogram and cross-variogram for IBRL data.

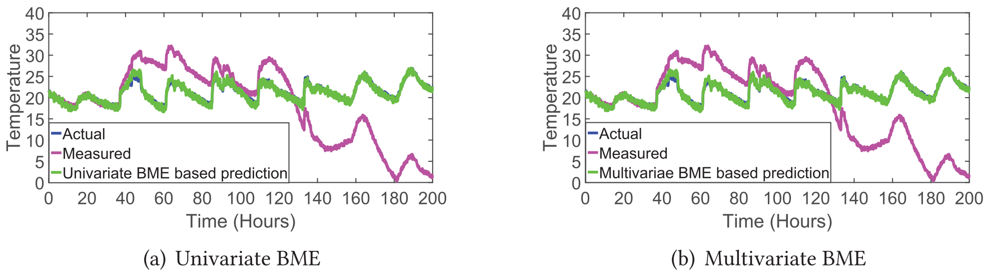


Fig. 23. Actual, measured, and corrected temperature values for the univariate and multivariate BMEs.

In order to investigate the improvement in estimated temperature values, we compare the multivariate (using temperature and humidity) and the univariate (using temperature only) BME estimation algorithm for drift detection based on the RMSE. A drift having zero mean and a process noise variance of 1 is added to the temperature values of 30 randomly chosen sensors out of the 41 low-precision sensors. A random noise having zero mean and variance 0.1 is added to the temperature and humidity readings to simulate the measurement noise. The other settings, such as parameters for the Kalman filter and the number of neighboring hard and soft sensors, are kept to the same values as the previous experiments.

Figure 23 shows that both the univariate and multivariate BME-based drift detection algorithms are able to track and correct the drift with good accuracy. It is clear from the figure that while temperature measurements are far away from the actual temperature values, both the univariate and multivariate BME-based methods are able to estimate the drift correctly, and hence the corrected temperature values resulted are much closer to the actual temperature values. Figure 24 shows the average RMSE of each sensor node for the univariate and multivariate BME-based methods. It can be noticed that the average RMSE for almost each of the sensor nodes is different for the univariate and multivariate BMEs. Although the difference is small, the multivariate BME has a lower RMSE than the univariate BME method. The overall RMSEs over all the sensor nodes and the time instances for univariate and multivariate BME are 0.81 and 0.70, respectively. It shows

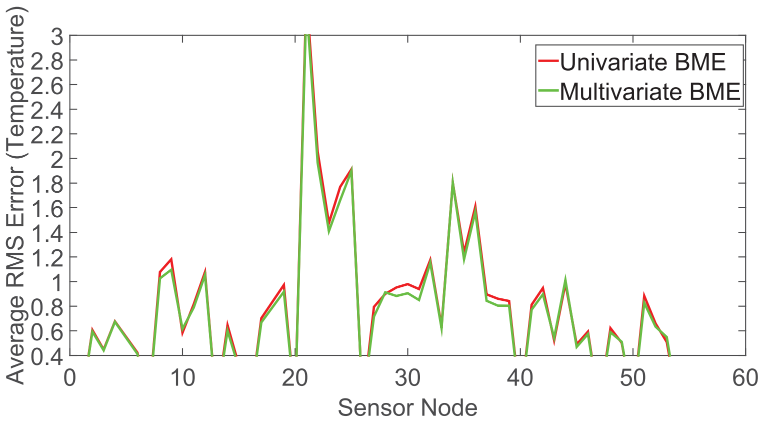


Fig. 24. Average RMSE (temperature) for univariate and multivariate BME methods.

that the multivariate BME method improves the temperature estimates by incorporating humidity measurements of neighborhood sensors.

9 DISTRIBUTED APPROACH OF THE BME ALGORITHM FOR REAL WSNs

In order to use the BME algorithm for in-network drift detection, implementation of the BME algorithm in a distributed manner is required, which can be deployed at sensor nodes itself. However, there are many challenges that need to be addressed in order to develop the BME for use in a distributed manner. As we have discussed in our previous experiments, the BME operation is a computationally intensive task. Computation time increases significantly when a large number of low-precision sensors are used for estimation. The most time-consuming part in the BME algorithm is the multiple integration (refer to Equation (9)) involved in computing the multivariate Gaussian probabilities. Multivariate normal cumulative probabilities are computed using adaptive quadrature (for small soft data) and Monte Carlo integration methods (for large numbers of soft data) [19] over a hyper-rectangle defined by the lower and upper limits corresponding to the lower and upper bounds of the interval kind of soft data. It uses the complementary error function $erfc$ [3] to compute the normal cumulative distribution function. The other time-consuming part is the maximization of the posterior PDF (refer to Equation (9)) over some fixed interval, which is derived from the soft data. This maximization of posterior information (Inf) can be done by minimizing $-log(Inf)$. Thus, it becomes as an optimization problem in which we have to find the estimated value in the interval $I = [a, b]$ so that $-log(Inf)$ is minimum. In the following text, we discuss the associated challenges in detail and propose our approaches for developing a distributed BME algorithm.

Challenges in Developing an In-Network Distributed BME Algorithm

With limited computational capabilities and memory constraints of low-cost sensor nodes, it is challenging to develop the BME algorithm at a sensor node level for in-network drift detection. In the following, we discuss the main challenges in developing the BME algorithm in a distributed manner at each sensor node and our approach to deal with them.

- **Computation of multivariate Gaussian probabilities:** It is difficult to implement the algorithm for computing the multiple integrals on a sensor node due to limited processing power and memory constraints. As we discussed in Section 7.2.3, reliable estimates can be obtained using BME even if we use only a few neighboring hard ($N_{nh} = 2$) and soft sensors ($N_{nl} = 2$). In this way, we only have to compute the bivariate or trivariate normal PDFs over the hyperrectangles, instead of computing a full multivariate Gaussian PDF. Hence, in this case, multiple integrals have been reduced to the bivariate or trivariate ones. We implemented the algorithms for bivariate and trivariate normal PDF on the sensor node itself based on the scheme described in Refs. [17] and [20], which are computationally efficient than computing the multivariate Gaussian PDF.

- **Computation of complementary error function:** Bivariate and trivariate normal PDF of any random variable X computes complementary error function in order to obtain the cumulative distribution function of X. For any realization z of X, calculation of error function $erfc(z)$ is done by integrating an exponential function over the limits from z to ∞. It is very challenging to implement this on the resource-constrained low-cost wireless sensor nodes due to the high computational effort required in integrating over very large limits. To deal with this challenge, we implemented an approximation-based [1] method, as shown in Algorithm 6 (Appendix) to compute the error function for cumulative distribution function at a sensor node.
- **Optimization to find the minimum of $-log(pdf)$:** In this subroutine, the estimated value is obtained for which the single-valued function $-log(pdf)$ is minimum. Bivariate or trivariate Gaussian PDFs are computed iteratively in this subroutine until the optimum estimation value is obtained. In order to deal with this time-consuming part, we implemented a Brent's numerical method [9] to find the minimum of a single-valued nonlinear function within a range, which finds the minima without using any derivatives. Note that the computational complexity to compute ϵ-accurate solution at a linear rate is given by $O(log(1/\epsilon))$, as discussed in Section 7.2.2, a desirable value for ϵ can be chosen to reduce the computational time.
- **Variogram calculation:** As a prior knowledge, mean and covariance function are used in BME. Covariance estimate matrices are obtained using inverse of the variogram. The Gauss Jordan method [41] is implemented to compute the inverse of variogram at each node.

By using the aforementioned approaches, we developed the BME-based estimation algorithm in a distributed manner for drift detection, where each node tracks and corrects its own drift by using measurements from neighboring sensor nodes. We evaluate the performance of the distributed scheme, which uses the BME and KF for sensor drift detection and correction in WSNs by performing three experiments on different real-life sensor network datasets. In the first experiment, we implemented our algorithm on each sensor node for in-network drift detection using the on-board processing capabilities, and we discuss how it improves the battery life of the sensor nodes. A small, real WSN is set up at the ISSNIP Lab, University of Melbourne, Australia, for this experiment. In the second experiment, we evaluate the applicability of our scheme in detecting the drift in a WSN when multiple sensors are drifting. A set of real sensor measurements gathered from a deployment of wireless sensors in the IBRL were used for this purpose. In the last experiment, we demonstrate the applicability of our scheme for outdoor IoT applications. We used real IoT data collected from two deployments (for an urban microclimate monitoring at two locations), namely Fitzroy Gardens and Docklands Library, in the City of Melbourne, Australia. In the following text, we discuss each of these experiments in detail.

9.1 In-Network BME Algorithm on Sensor Node in a Real WSN

In order to demonstrate that the proposed algorithm can be implemented on real WSNs, we built a small WSN consisting of six sensor nodes at the ISSNIP Lab [28], University of Melbourne,

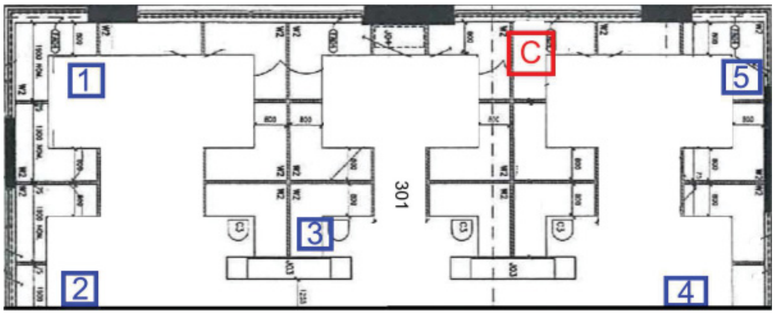


Fig. 25. ISSNIP Lab schematic showing the positions of the central node (red square) and the neighboring nodes (blue squares).

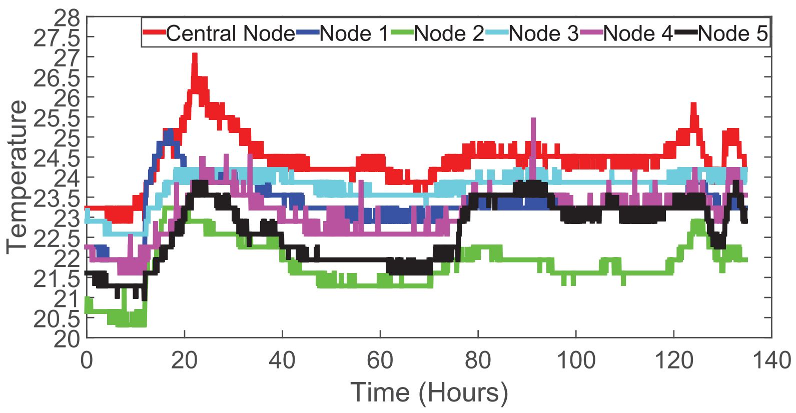
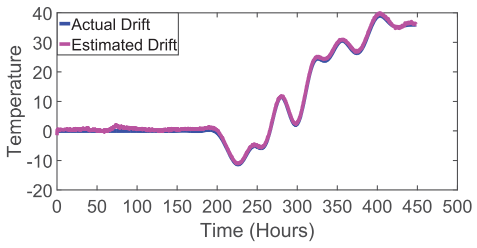


Fig. 26. Actual temperature of all sensor nodes.

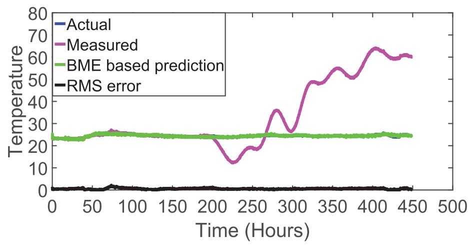
Australia. The WSN setup is shown in Figure 25 with the sensor node locations. From these six sensors nodes, we use one sensor as the central node and the remaining five as its neighboring nodes. In Figure 25, the location of the central node and its neighboring nodes are shown using red and blue squares, respectively. All of the neighboring sensor nodes are considered as soft sensor nodes. The measurement errors of sensors can be used to derive the interval value for each sensor. Based on the method for calculating intervals, as discussed in Section 7.1.1, the interval value for each soft sensor was obtained as 0.5. Figure 26 shows the central and neighboring sensor nodes' temperature values, collected over the period between May 3 and May 9, 2016, at a sampling rate of 3 minute.

We used Waspmote [34] as a wireless sensor node, which is an open-source wireless sensor platform. The MCP9700A sensor was used to measure the temperature at each of the six different locations. These sensor nodes have limited memory and processing power. Waspmote has a 14.75MHz processor, 8kB SRAM, 128kB Flash, 4kB EEPROM, and 32kHz real-time clock, and communicates with other devices at 2.4GHz frequency using an XBee module plugged into them. Zigbee protocols was used to transmit data from neighboring sensor nodes to the central node.

In this experiment, we considered that only the central node is drifting. A synthetic drift is injected into the central node measurements, where the drift of the sensor is simulated using an autoregressive moving average (ARMA) process having a mean $\mu_{process} = 0$ and variance $\sigma^2_{process} = 0.5$. In addition to the drift, a random noise having mean $\mu_{random} = 0$ and variance $\sigma^2_{random} = 0.1$ is also added to the sensor readings to simulate the measurement noise. The



(a) Actual and estimated drift



(b) Central node temperature

Fig. 27. (a) Actual and estimated drift. (b) Actual, measured, and corrected temperature values with RMSE at the central node.

central node measures its temperature and requests the neighboring nodes to send their temperature readings to it. Upon receiving the request, each neighboring node measures its local temperature and sends it to the central node. The central node then performs the BME estimation using the neighbouring hard and soft sensor measurements to predict the temperature at the central node. The estimated drift is then obtained using the Kalman filter.

We implemented the distributed BME algorithm at each sensor node for in-network drift detection. As we used approximations to calculate the complementary error function, and used bivariate and trivariate cdfs to avoid multiple integrals, only approximately half of the SRAM (4,134 bytes out of 8,192kB) and flash memory (56,514 bytes out of 122,880kB) is utilized. The pseudocode of our main distributed BME-based drift detection algorithm implemented in the central node is illustrated in Algorithm 2. The subroutine algorithms used within this main code are given in the Appendix. These subroutines are for Kalman filtering (Algorithm 3), BME estimation (Algorithm 4), computation of posterior PDF (Algorithm 5), and normal cdf (Algorithm 6). The exponential variogram model with parameters, range $a = 8.36$, sill $C_1 = 0.54$, and zero nuggets are used as prior knowledge. The parameter values of the Kalman filter used for the sensor drift are $Q = 1$ and $R = 0.0001$. Since our distributed algorithm can compute bivariate and trivariate Gaussian PDFs, we keep $N_{nl} = 2$. Figure 27(a) shows the actual and estimated drift at the central node. This figure shows that that in-network BME-based drift detection algorithm traces the drift perfectly and then corrects them using a Kalman filtering accurately. Figure 27(b) shows the actual, measured, corrected temperature values, and the RMSEs at the central node. It can be inferred from the figure that our distributed BME algorithm is capable of detecting and correcting the drift at the sensor node itself.

When a network is deployed over a large area and/or the number of nodes in the network increases, the communication overhead also increases, and it reduces the lifetime of the resource-constrained network. To address the issue of power constraint in a large WSN, only a limited number of neighboring sensors can be used for estimation. A study was performed in Ref. [33] to investigate the effect of the faraway sensors on the estimation accuracy at the central node. It was perceived that the contribution by the faraway sensors for the estimation are much smaller than the close by sensors. The aforementioned in-network implementation of the BME enables drift detection in a distributed manner where each sensor node has to use measurements only from a few neighboring sensors to detect and correct its drift; thus, it reduces the communication overhead in the network. As the majority of the energy in a WSN is consumed in communication [40], our in-network drift detection improves the battery life of the sensor node and keeps the WSN operational for a longer time while correcting the drift in a distributed manner. Furthermore, the scalability issue in a large WSN can be addressed by partitioning the network into

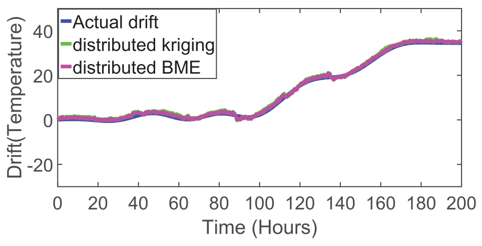
ALGORITHM 2: Central Node Code

```
Input: $N_{nh}$ - number of neighboring high-precision sensors
       $N_{nl}$ - number of neighboring low-precision sensors
       $S_h$ - locations of high-precision sensors
       $S_l$- locations of low-precision sensors
       $S_E$ - location of the central node
       $X_{hard}$ - measurement of high-precision sensors
       $aX_{soft}$ - values of lower bound of interval of low-precision sensors
       $bX_{soft}$ - values of upper bound of interval of low-precision sensors
       $C_0, C_1, a$ - parameters of the covariance model (e.g., value of nugget, sill, and range)
       $\mu_{process}$ - process noise mean
       $\sigma^2_{process}$- process noise variance
       $\mu_{random}$ - random noise mean
       $\sigma^2_{random}$- random noise variance
       $Q$ - assumed process variance
       $R$ - assumed measured variance
Output: $D_{present}$- actual sensor drift at present time
       $d_{present}$ - Eestimated sensor drift at present time
Setup $D_{previous}$ - 0 (actual previous drift initialization)
       $d_{previous}$ - 0 (estimated previous drift initialization)
      $P_{previous}$ - 1 (Kalman filter parameter initialization)
//Measure temperature of central node
$T_{central}$ = Central node measured temperature
//Add ARMA process and measurement noise to $T_{central}$ to get drifted measurement
$D_{present} = D_{previous} + \mu_{process} + \sigma^2_{process} \times rand(-1, 1)$
$noise_{measurement} = \mu_{random} + \sigma^2_{random} \times rand(-1, 1)$
$T_{drifted} = T_{central} + D_{present} + noise_{measurement}$
//Get temperature readings from neighboring nodes
Send request to all the neighboring nodes to measure and send their temperature values
for $s \leftarrow 1$ to $n$ do
    $T_s$ = Received temperature values from node $s$;
end
//Apply Bayesian Maximum Entropy technique to neighboring nodes' temperatures to
get predicted temperature at central node
$T_{predicted} = BME(S_E, S_{nh}, S_{nl}, X_{hard}, aX_{soft}, bX_{soft}, C_0, C_1, a, N_{nh}, N_{nl})$
Apply Kalman filter to get the estimated present drift
$(d_{present}, P_{present}) = Kalman(T_{drifted}, T_{predicted}, d_{previous}, P_{previous}, Q, R)$
//Actual and estimated drift for present step becomes $D_{previous}$ and $d_{previous}$ for next
step $D_{previous} = D_{present}$,    $d_{previous} = d_{present}$,    $P_{previous} = P_{present}$
```

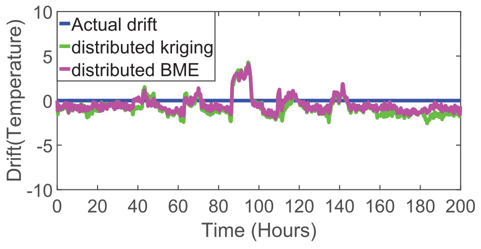
small groups based on distances, such as using well-known node clustering methods, and using the sensor within that group for performing estimations and drift detection.

9.2 Distributed BME Algorithm for Multiple Drifting Sensors

In this experiment, we implemented our distributed BME algorithm for the IBRL temperature dataset described in Section 7 and compared it with the distributed approach of kriging [33] for



(a) Drifting sensor



(b) Sensor with no drift

Fig. 28. Actual and estimated drift using distributed kriging and BME for (a) drifting sensors and (b) sensors having no drift.

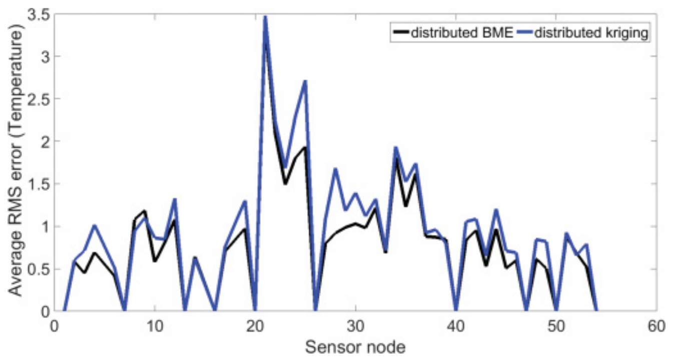
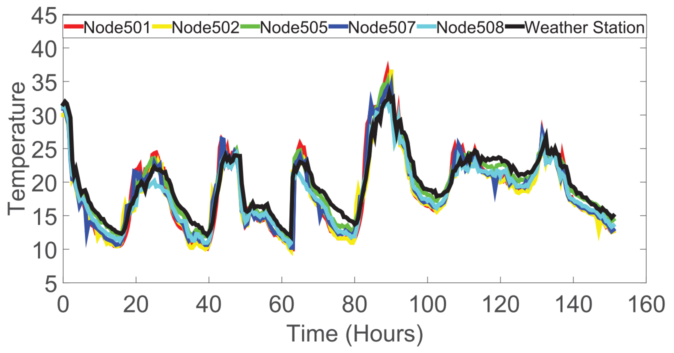


Fig. 29. Average RMSE of temperature for distributed approach of the kriging and BME methods.

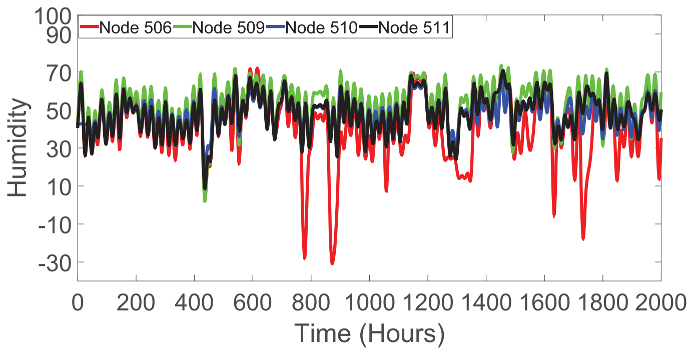
drift detection. In this experiment, instead of performing drift detection and correction at a centralized location, the distributed BME has been implemented at each node. All the settings are kept similar as per the previous experiments using the same dataset, except the number of nearest soft sensor nodes N_{nl}, which is now set to 2. We kept the number of drifting sensors at 30 and the interval value to 0.5 for low-precision sensors. All the nodes keep track of their estimated drift values and shared their drift-corrected temperature values with all the other nodes in the network. Figure 28 shows the actual and estimated drift values for drifting and nondrifting sensors for both the distributed estimation algorithms. Our distributed BME algorithm is not only able to track the drift perfectly for the drifting sensors but also produces negligible drift (desirable) for nondrifting sensors unlike the distributed kriging-based method. Figure 29 shows the average RMSE of all the sensor nodes. Our distributed BME algorithm shows a lower error than the distributed kriging-based algorithm.

9.3 Urban Microclimate Data from IoT Deployment

In this experiment, we evaluate the performance of the distributed BME-based and the kriging-based drift detection schemes on the real-life data obtained from an outdoor IoT deployment. These data were collected from our WSN deployment at two locations in the City of Melbourne in collaboration with Melbourne City Council and Arup [5]. Five sensors were deployed at the Fitzroy Gardens, and four sensors were deployed at the Docklands Library under different urban canopy conditions for microclimate monitoring. Low-cost sensors, namely the Waspmote [34], were deployed to measure temperature, humidity, and luminosity parameters at 10-minute intervals. Two



(a) Temperature measurements of Weather station, and five sensor nodes deployed in Fitzroy Gardens



(b) Humidity measurements of four sensor nodes deployed in Dockland Library (DL)

Fig. 30. Sensor readings. (a) Temperature measurements of five sensor nodes deployed in Fitzroy Gardens (FG), and (b) humidity measurements of four sensor nodes deployed in Dockland Library (DL).

separate experiments were performed on the temperature and humidity measurements collected from the sensors deployed in the Fitzroy Gardens and Docklands Library, respectively. In the first experiment, temperature measurements collected during the 1-week period starting from December 17, 2014 to December 24, 2014 were used. The corresponding sensor node IDs are 501, 502, 505, 507, and 508, which were deployed in the Fitzroy Gardens (FG). The second experiment used humidity measurements collected between December 15, 2014 and February 25, 2015 from sensor nodes with IDs 506, 507, 508, and 509, which were deployed in the Docklands Library (DL) area. In both experiments, distributed BME and kriging were evaluated for comparison purposes.

Similar to the IBRL dataset, these data were resampled at 30-minute intervals by averaging them in 30-minute windows. The missing values of features were replaced by their nearest value. In the first experiment, the temperature measurements of a weather station, which is located nearby the FG, are considered as high-precision measurements (hard data), and the measurements of the five deployed sensors are considered as soft (interval) data. In the second experiment, all the sensor nodes are considered as soft sensors due to nonavailability of high-precision humidity sensors close to the DL region. Figure 30 shows the temperature and humidity measurements considered for these experiments.

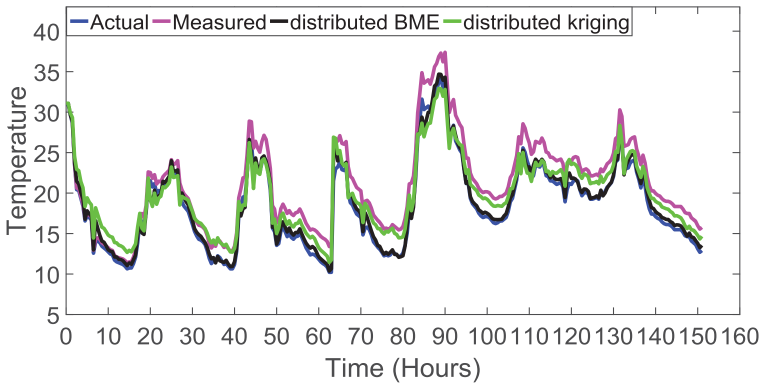


Fig. 31. Fitzroy Gardens Dataset: Actual, measured, and corrected temperature values using the BME-based and kriging-based drift detection schemes.

Table 2. Average RMSE for the Distributed BME-based and Kriging-Based Schemes on Each Dataset

Dataset	**IBRL**	**FG**	**DL**
Kriging	1.08	1.34	9.12
BME	**0.82**	**0.79**	**6.75**

Based on the interval calculation method discussed in Section 7.1.1, the interval value for each soft sensor was chosen as 0.5. The values of variogram parameters C_0, C_1, and a for FG and DL datasets were chosen as $(0, 0)$, $(0.42, 20.3297)$, and $(10.49, 29.87)$, respectively. These variogram parameters were chosen such that they well represent the data within the given geographical area.

For the FG dataset, synthetic drift having zero process mean and 0.1 variance was added randomly to one of the sensor nodes, while no synthetic drift was added to the measurements of the DL dataset. This is because humidity sensor of one of the sensor nodes deployed at the DL, namely node 506, started to report erroneous values compared to other sensors after a period of correct operation. This is shown in red in Figure 30(b). This is a typical example of a real-life drifting scenario in a sensor network. We aim to detect and correct these erroneous measurements using the BME-based algorithm and present the suitability of our scheme for IoT applications. Similar to the IBRL dataset experiment, the values of the Q and R parameters for the FG and DL datasets were chosen as $(1, 0.1)$ and $(1, 0.5)$, respectively, based on the grid search over the parameter space. The maximum number of hard and soft sensors for FG and DL were chosen as $(1, 2)$ and $(0, 2)$, respectively.

Figure 31 shows the performance of the BME and simple kriging methods in estimating and correcting the drift for temperature sensors from the FG dataset. It can be clearly observed that the corrected drift using the BME is much closer to the actual drift compared to the kriging-based method.

Figure 32 demonstrates that the BME-based scheme is capable of correcting the erroneous measurements of humidity from node 506 much better (with a significant difference in average RMSE) than the kriging-based method. The average RMSE incurred for the distributed BME and kriging-based methods, for the entire duration of the FG and DL experiments, are shown in Table 2. The

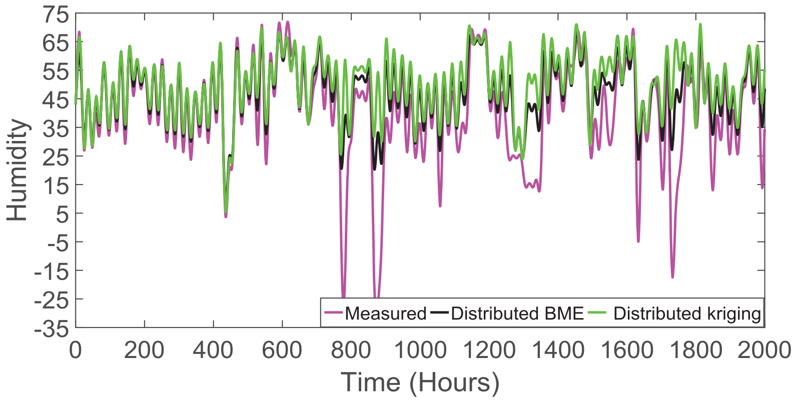


Fig. 32. Docklands Library Dataset: Actual, measured, and corrected humidity values using the distributed BME-based and kriging-based drift detection schemes.

minimum average RMSE for each dataset is shown in bold. The difference between the average RMSE is significant for the FG and DL experiments as compared to the IBRL experiment. It clearly shows the superiority of the BME-based algorithm over the kriging-based scheme for IoT applications, such as for environment monitoring.

10 DISCUSSION

The sensor drift we consider in this article is smooth, as it is the most common type of drift [22, 25]. It usually grows slowly independent of the measured variable dynamics which means the physical phenomenon being measured can change fast but change in the sensor drift must be small. The standard KF with single drift model does not efficiently respond to changes in the dynamics when the drift changes abruptly at some points. This can be observed in node 506 measurements (see Figure 30(b)), which has few surges in its drift. Such drift behavior is not well followed by the KF (see Figure 32), and due to this, the average RMSE for the DL dataset is relatively higher as compared to the IBRL and FG datasets, as shown in Table 2. Therefore, this assumption of smooth drift is necessary to justify the use of the KF for drift estimation. However, the IMM can deal with abrupt changes in drift, which can be incorporated in our algorithm, albeit with higher computational overhead. IMM has been used for unsmooth drift detection and correction in [49]. The IMM approach has been originally used in target tracking to track manoeuvring objects that show sudden changes in their dynamics [38].

11 CONCLUSIONS

Automatic detection and correction of sensor drifts in an IoT environment is important for reliable monitoring of the environment. In this article, we proposed a BBME-based and Kalman filter–based framework to automatically detect and correct the drifts that develop in a network of sensors, where a mix of low-cost, low-precision, resource-constrained sensors and high-precision, often expensive, sensors are used to deploy over a large geographical (spatial) area for monitoring purposes. Both centralized and distributed implementations of the BME-based approaches are discussed. Evaluation using real sensor network data reveals that the BME-based drift detection algorithm outperforms various kriging-based spatial estimation methods. Further, we discussed how the multivariate BME-based framework can be used to incorporate additional features (variables) for improving estimation accuracy and subsequently to improve the drift detection and correction process. The implementation of our distributed BME algorithm on a real WSN hardware demonstrated its applicability in a real-world scenario.

A APPENDIX

ALGORITHM 3: Kalman Filter

```
Input: $Q$ - Assumed process variance
       $R$ - Assumed measured variance
       $T_{measured}$ - Measured temperature
       $T_{predicted}$ - Predicted temperature
       $d_{previous}$- Previous estimated drift
       $P_{previous}$- Kalman filter parameter
Output: $P_{present}$- Kalman filter parameter
       $d_{present}$ - Estimated sensor drift at present time
//Calculate predicted drift
$d_{predict} = T_{measured} - T_{predicted}$
//Time update equation
$d_{apriori} = d_{previous}$
$P_{minus} = P_{previous} + Q$
//Measurement update equation
$K = \frac{P_{minus}}{P_{minus}+R}$
$d_{present} = d_{apriori} + K \times (d_{predict} - d_{apriori})$
$P_{present} = (1 - K) \times P_{minus}$
```

ALGORITHM 4: Bayesian Maximum Entropy Estimation

```
Input: $N_{nh}, N_{nl}$ - number of high- and low-precision sensors as neighbors, respectively,
       $S_E, S_{nh}, S_{nl}$- locations of central node, and neighboring high- and low-precision sensors,
       respectively,
       $zhlocal$ - measurement of neighboring high-precision sensors,
       $alocal, blocal$ - values of lower and upper bound of interval of neighboring low-precision
       sensors, respectively,
       $C_0, C_1, a$ - Variogram parameters, e.g., value of nugget, sill and range.
Output: $X_E$- estimated value at central node.
Setup: $X$ - unit vector of dimension of $(nhmax + nsmax)$.
//Compute the mean and variance assuming uniform distribution.
$mslocal = (alocal + blocal)/2$,   $vslocal = ((blocal - alocal).^2)/12$,   $z = [zhlocal, mslocal]$
//Compute the covariance matrix (inverse variogram) for hard sensor points.
//Compute the distance matrix between neighboring hard points.
for $i \leftarrow 1$ to $N_{nh}$ do
    for $j \leftarrow 1$ to $N_{nh}$ do
        $h[i,j] = dist(S_{nh_i}, S_{nh_j})$
    end
end
//Compute the self-variogram for hard data using Equation (7)
for $i \leftarrow 1$ to $N_{nh}$ do
    for $j \leftarrow 1$ to $N_{nh}$ do
        $\gamma_{hh}[i,j] = C_0 + C_1[1 - exp(-3h[i,j]/a)]$
    end
end
//Compute the covariance matrix (inverse variogram) of hard data using Gauss Jordan
Method [41].
$K_{hh} = Inverse(\gamma_{hh})$
//Compute the covariance matrix $K_{ss}$ for soft, $K_{kk}$ for estimation points,
cross-covariance vectors $K_{hs}$ between hard/soft points, $K_{kh}$ between hard/estimation
points, and $K_{sh}$ between soft/estimation points by following similar steps
(mentioned above).
//Build cross-covariance matrix $K_{khkh}$ for estimation+hard data, and $K_{skh}$ for
soft/estimation+hard data covariance matrix
$$K_{khkh} = \begin{bmatrix} K_{kk} & K_{kh} \\ K'_{kh} & K_{hh} \end{bmatrix}, \quad K = \begin{bmatrix} K_{hh} & K'_{sh} \\ K'_{sh} & K_{ss} \end{bmatrix}, \quad K_{skh} = \begin{bmatrix} K'_{ks} \\ K_{sh} \end{bmatrix}$$
$K_{ss}$ = add vslocal to diagonal values of $K_{ss}$ [35].
//Compute the conditional covariance matrix [46]
$InvK_{khkh} = Inverse(K_{khkh})$,   $KskhinvK_{khkh} = K_{skh} \times invK_{khkh}$,
$K_{ssifkh} = K_{ss} - KskhinvK_{khkh} \times K'_{skh}$
//Compute the estimated mean from hard and soft (centered) data using linear regression
$XtinvK = X' \times Inverse(K)$
$InvXtinvKX = Inverse(XtinvK \times X)$
$mk_{est} = InvXtinvKX \times XtinvK \times z$
// Subtract the mean from the hard and soft data
$zh_{cond} = zhlocal - mk_{est}$.   $al = alocal - mk_{est}$,   $bl = blocal - mk_{est}$
// Initialize the minimum and maximum guess values in optimization function for to
estimate value where posterior PDF is maximum, i.e., $-log(pdf)$ is minimum
$zk_{min} = minimum(zh_{cond}, a)$,   $zk_{max} = maximum(zh_{cond}, b)$
// Find the local minimum value $\in \{zk_{min}, zk_{max}\}$ for min($-log(pdf)$) using
BRENT [9] algorithm.
$x_E = localmin(@Posteriorpdf, zk_{min}, zk_{max}, zh_{cond}, al, bl, InvK_{khkh}, KskhinvK_{khkh}, K_{ssifkh})$
// Add the mean to the estimate to get final estimated value at central node
$X_E = x_E + mk_{est}$
```

ALGORITHM 5: Posterior Probability Distribution Function Calculation

```
Input:  $z_k$ - scaler input value (between $al$ and $bl$) for which the posterior PDF is computed,
        $zh_{cond}$ - vector of conditioning hard values,
        $al, bl$ - vector of lower and upper integration limits,
        $InvK_{khkh}$ - inverse of the covariance matrix for the estimation location and the hard data
        $KskhinvK_{khkh}$ - vector equal to $K_{skh} \times invK_{khkh}$, where $K_{skh}$ and $K_{khkh}$ are covariance
        matrices,
        $K_{ssifkh}$ - matrix of conditional covariances for the soft data given $z_k$ and $zhlocal$.
Output: $f$- value of $-log(pdf)$ at $z_k$ up to some constants.
Setup: $F = 0$ (initial value of PDF)
//Assume the zero mean ($\mu = 0$) and compute the positive semidefinite matrix
$sigma = (K_{ssifkh} + K'_{ssifkh})/2$
//Compute the conditional mean
$z_{kh} = [z_k, zh_{cond}]$,    $msifkh = KskhinvK_{khkh} \times z_{kh}$
//Compute the lower $X_L$ and upper $X_U$ limits for computing posterior PDF
$X_L = al - msifkh$,    $X_U = bl - msifkh$
//Standardize $X$ and sigma value
$s$ = square root of diagonal value of sigma.
$rho = sigma/s \times s'$
$X_{L0} = X_L/s$,    $X_{U0} = X_U/s$
//Build the semi-infinite hyperrectangles by defining four limits
$LL = X_{L0}$,    $LU = [X_{L0}(1), X_{U0}(2)]$
$UL = [X_{U0}(1), X_{L0}(2)]$,    $UU = X_{U0}$
//Compute the bivariate normal cumulative probability using the algorithm [17] over
the hyperrectangle with lower and upper limits defined by $X_{L0}$ and $X_{U0}$, respectively
$F = F + Bivariatepdf(LL, rho)$
$F = F - Bivariatepdf(LU, rho)$
$F = F - Bivariatepdf(UL, rho)$
$F = F + Bivariatepdf(UU, rho)$
if $F < 0$ then
    $F = 0$
end
if $F > 0$ then
    $F = 1$
end
//Make sure that log(P) does not yield $-\infty$
$Q = maximum(F, 1e^{-323})$
//Compute the value of the $-log(pdf)$
$f = 0.5 * (z'_{kh} \times InvK_{khkh} \times z_{kh}) - log(F)$
```

ALGORITHM 6: Normal Cumulative Distribution Function Calculation Using Approximation

```
**Input:** $z$- input value of random variable
**Output:** $p$- cumulative distribution function for normal distribution.
**Setup**: Approximation constants
    $a1 = 0.254829592, a2 = -0.284496736, a3 = 1.421413741,$
$a4 = -1.453152027, a5 = 1.061405429, pr = 0.3275911$
**//Compute the cdf of normal distribution [1]**
$den = abs(z)/sqrt(2)$
$t = 1.0/(1.0 + pr \times den)$
$y = 1.0 - (((((a5t + a4)t) + a3)t + a2)t + a1)t \times exp(-den^2)$
$p = 0.5. * (1.0 + sign(z) \times y)$
```

REFERENCES

[1] Milton Abramowitz and Irene A Stegun. 1964. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. Vol. 55. Courier Corporation.

[2] Ian F. Akyildiz, Mehmet C. Vuran, and Ozgür B. Akan. 2004. On exploiting spatial and temporal correlation in wireless sensor networks. In *Proceedings of WiOpt*, Vol. 4. 71–80.

[3] Larry C. Andrews. 1985. *Special Functions for Engineers and Applied Mathematicians*. Macmillan.

[4] Tom Artursson, Tomas Eklöv, Ingemar Lundström, Per Mårtensson, Michael Sjöström, and Martin Holmberg. 2000. Drift correction for gas sensors using multivariate methods. *Journal of Chemometrics* 14, 5-6 (2000), 711–723.

[5] ARUP. 2016. Retrieved from http://www.arup.com/Global_locations/Australia.aspx.

[6] Laura Balzano and Robert Nowak. 2008. Blind calibration of networks of sensors: Theory and algorithms. In *Networked Sensing Information and Control*. Springer, 9–37.

[7] Dimitri P. Bertsekas. 1999. Nonlinear programming. (1999).

[8] James C. Bezdek, Sutharshan Rajasegarar, Masud Moshtaghi, Chris Leckie, Marimuthu Palaniswami, and Timothy C. Havens. 2011. Anomaly detection in environmental monitoring networks [application notes]. *IEEE Computational Intelligence Magazine* 6, 2 (2011), 52–58.

[9] Richard P. Brent. 2013. *Algorithms for Minimization Without Derivatives*. Courier Corporation.

[10] Vladimir Bychkovskiy, Seapahn Megerian, Deborah Estrin, and Miodrag Potkonjak. 2003. A collaborative approach to in-place sensor calibration. In *Information Processing in Sensor Networks*. Springer, 301–316.

[11] G. Christakos, P. Bogaert, and M. L. Serre. 2002. Temporal GIS. Springer-Verlag, New York, N.Y., 220 p. With CD-ROM, 2002.

[12] I. Clarke. 1979. Practical geostastics. Retrieved from http://www.kriging.com/PG1979.

[13] Ivo Couckuyt, A. Forrester, Dirk Gorissen, Filip De Turck, and Tom Dhaene. 2012. Blind kriging: Implementation and performance analysis. *Advances in Engineering Software* 49 (2012), 1–13.

[14] N. Cressie. 1993. Statistics for Spatial Data: Wiley Series in Probability and Statistics. (1993).

[15] Jonny Carlos da Silva, Abhinav Saxena, Edward Balaban, and Kai Goebel. 2012. A knowledge-based system approach for sensor fault modeling, detection and mitigation. *Expert Systems with Applications* 39, 12 (2012), 10977–10989.

[16] Intel Lab Data. 2004. Homepage. Retrieved March 15, 2016 from http://db.lcs.mit.edu/labdata/labdata.html.

[17] Zvi Drezner and George O. Wesolowsky. 1990. On the computation of the bivariate normal integral. *Journal of Statistical Computation and Simulation* 35, 1–2 (1990), 101–107.

[18] Xavier Emery. 2005. Simple and ordinary multigaussian kriging for estimating recoverable reserves. *Mathematical Geology* 37, 3 (2005), 295–319.

[19] Alan Genz. 1992. Numerical computation of multivariate normal probabilities. *Journal of Computational and Graphical Statistics* 1, 2 (1992), 141–149.

[20] Alan Genz. 2004. Numerical computation of rectangular bivariate and trivariate normal and t probabilities. *Statistics and Computing* 14, 3 (2004), 251–260.

[21] Alan Genz and Frank Bretz. 2009. *Computation of Multivariate Normal and t Probabilities*. Vol. 195. Springer Science & Business Media.

[22] John-Erik Haugen, Oliver Tomic, and Knut Kvaal. 2000. A calibration method for handling the temporal drift of solid state gas-sensors. *Analytica Chimica Acta* 407, 1 (2000), 23–39.

[23] Gustavo Hernandez-Penaloza and Baltasar Beferull-Lozano. 2012. Field estimation in wireless sensor networks using distributed kriging. In *Proceedings of the IEEE International Conference on Communications (ICC)*. IEEE, 724–729.

[24] Bruce Hoadley. 1970. A Bayesian look at inverse linear regression. *J. Amer. Statist. Assoc.* 65, 329 (1970), 356–369.

[25] M. Holmberg and T. Artursson. 2002. Drift compensation, standards, and calibration methods. In *Handbook of Machine Olfaction: Electronic Nose Technology.* Wiley-VCH Verlag GmbH & Co. KGaA, 325–346.

[26] Martin Holmberg, Fabrizio A. M. Davide, Corrado Di Natale, Arnaldo D'Amico, Fredrik Winquist, and Ingemar Lundström. 1997. Drift counteraction in odour recognition applications: Lifelong calibration method. *Sensors and Actuators B: Chemical* 42, 3 (1997), 185–194.

[27] Martin Holmberg, Fredrik Winquist, Ingemar Lundström, Fabrizio Davide, Corrado DiNatale, and Arnaldo D'Amico. 1996. Drift counteraction for an electronic nose. *Sensors and Actuators B: Chemical* 36, 1 (1996), 528–535.

[28] ISSNIP. Internet of Things (IoT) for Creating Smart Cities. 2016. Retrieved from http://issnip.unimelb.edu.au/research_program/sensor_networks/Internet_of_Things.

[29] Vasanth Iyer and S. Sitharama Iyengar. 2011. Modeling unreliable data and sensors: Using F-measure attribute performance with test samples from low-cost sensors. In *Proceedings of the IEEE 11th International Conference on Data Mining Workshops (ICDMW'11)*. IEEE, 15–22.

[30] V. Roshan Joseph, Ying Hung, and Agus Sudjianto. 2008. Blind kriging: A new method for developing metamodels. *Journal of Mechanical Design* 130, 3 (2008), 031102.

[31] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering* 82, 1 (1960), 35–45.

[32] Dheeraj Kumar, Sutharshan Rajasegarar, and Marimuthu Palaniswami. 2013. Automatic sensor drift detection and correction using spatial kriging and kalman filtering. In *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 183–190.

[33] Dheeraj Kumar, Sutharshan Rajasegarar, and Marimuthu Palaniswami. 2015. Geospatial estimation-based auto drift correction in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)* 11, 3 (2015), 50.

[34] Libelium. 2016. Waspmote. Retrieved from http://www.libelium.com/products/waspmote/.

[35] BME Library. 2015. (2015). Retrieved Jan 15, 2015 from http://www.unc.edu/depts/case/BMELIB.

[36] S. N. Lophaven, H. B. Nielsen, and J. Sondergaard. 2002. DACE: A matlab kriging toolbox. 2002. *Technical University of Denmark.* (2002).

[37] Georges Matheron. 1963. Principles of geostatistics. *Economic Geology* 58, 8 (1963), 1246–1266.

[38] A. Munir and D. P. Atherton. 1995. Adaptive interacting multiple model algorithm for tracking a manoeuvring target. *IEE Proceedings-Radar, Sonar and Navigation* 142, 1 (1995), 11–17.

[39] Margaret A. Oliver and Richard Webster. 1990. Kriging: A method of interpolation for geographical information systems. *International Journal of Geographical Information System* 4, 3 (1990), 313–332.

[40] Gregory J. Pottie and William J. Kaiser. 2000. Wireless integrated network sensors. *Communications of the ACM* 43, 5 (2000), 51–58.

[41] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. 1992. Gauss-jordan elimination and gaussian elimination with backsubstitution. *The Art of Scientific Computing Sections in Numerical Recipes in Fortran:* (1992).

[42] Sutharshan Rajasegarar, Timothy C. Havens, Shanika Karunasekera, Christopher Leckie, James C. Bezdek, Milan Jamriska, Ajith Gunatilaka, Alex Skvortsov, and Marimuthu Palaniswami. 2014. High-resolution monitoring of atmospheric pollutants using a system of low-cost sensors. *IEEE Transactions on Geoscience and Remote Sensing* 52, 7 (2014), 3823–3832.

[43] Sutharshan Rajasegarar, Christopher Leckie, and Marimuthu Palaniswami. 2008. Anomaly detection in wireless sensor networks. *IEEE Wireless Communications* 15, 4 (2008), 34–40.

[44] Sutharshan Rajasegarar, Christopher Leckie, Marimuthu Palaniswami, and James C. Bezdek. 2006. Distributed anomaly detection in wireless sensor networks. In *Proceedings of the 10th IEEE Singapore International Conference on Communication Systems (ICCS'06)*. IEEE, 1–5.

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

[46] M. L. Serre and G. Christakos. 1999. Modern geostatistics: Computational BME analysis in the light of uncertain physical knowledge - The equus beds study. *Stochastic Environmental Research and Risk Assessment* 13, 1–2 (Apr 1999), 1–26.

[47] Marc L. Serre and G. Christakos. 1999. Modern geostatistics: Computational BME analysis in the light of uncertain physical knowledge–The equus beds study. *Stochastic Environmental Research and Risk Assessment* 13, 1–2 (1999), 1–26.

[48] Maen Takruri, Subhash Challa, and Rajib Chakravorty. 2008. Auto calibration in drift aware wireless sensor networks using the interacting multiple model algorithm. In *Proceedings of the International Conference on Communications, Computers and Applications, MIC-CCA*. IEEE, 98–103.

[49] Maen Takruri, Subhash Challa, and Rajib Chakravorty. 2010. Recursive bayesian approaches for auto calibration in drift aware wireless sensor networks. *Journal of Networks* 5, 7 (2010), 823–832.

[50] Maen Takruri, Sutharshan Rajasegarar, Subhash Challa, Christopher Leckie, and Marimuthu Palaniswami. 2011. Spatio-temporal modelling-based drift-aware wireless sensor networks. *IET Wireless Sensor Systems* 1, 2 (2011), 110–122.

[51] Tyrone L. Vincent and Pramod P. Khargonekar. 1999. A class of nonlinear filtering problems arising from drifting sensor gains. *IEEE Transactions on Automatic Control* 44, 3 (1999), 509–520.

[52] Hans Wackernagel. 2013. *Multivariate Geostatistics: An Introduction with Applications*. Springer Science & Business Media.

[53] Chaocan Xiang, Panlong Yang, Chang Tian, Haibin Cai, and Yunhao Liu. 2015. Calibrate without calibrating: An iterative approach in participatory sensing network. *IEEE Transactions on Parallel and Distributed Systems* 26, 2 (2015), 351–361.

[54] Xiao Xu, J. Wesley Hines, and Robert E. Uhrig. 1998. On-line sensor calibration monitoring and fault detection for chemical processes. In *Proceedings of othe Maintenance and Reliability Conference (MARCON'98)*. 12–14.

[55] Marzia Zuppa, Cosimo Distante, Pietro Siciliano, and Krishna C. Persaud. 2004. Drift counteraction with multiple self-organising maps for an electronic nose. *Sensors and Actuators B: Chemical* 98, 2 (2004), 305–317.

Received September 2016; revised March 2017; accepted April 2017