Data Fusion in Wireless Sensor Networks

Maen Takruri

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Faculty of Engineering and Information Technology
UNIVERSITY OF TECHNOLOGY, SYDNEY

March 2009

Certificate of Authorship/Originality

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text. I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Production Note:
Signature removed prior to publication.

Maen Takruri, March 2009

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Subhash Challa, whose generosity and commitment are above and beyond the call of duty; words do not describe my gratitude. I appreciate his vast knowledge and skill in many areas and his assistance in completing this thesis. I would like also to thank my co-supervisor, Dr. Tim Aubrey for his assistance and support.

I must also thank the following people from the University of Technology, Dr. Khalid Aboura for his assistance and advice in statistics, Dr. Rami Al-Hmouz, Dr. Mohammad Momani, Dr. Kais Al-Momani, Mr. Mohammad Al-Hattab and Mr. Akram AlSukker for their support and invaluable philosophical debates, exchanges of knowledge, which helped enrich the experience.

I would like to acknowledge the support of the ARC Research Network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP) through the collaborative work with A/Prof. Marimuthu Palaniswami, A/Prof. Christopher Leckie, Mr. Sutharshan Rajasegarar from the University of Melbourne, to whom I would like to express my deep and sincere gratitude for their valuable contribution in implementing Support Vector Regression with the Drift detection and correction algorithms in chapters 6 and 7 of this thesis. Thanks also go to Dr. Rajib Chakravorty from NICTA Victoria Laboratory for his assistance and advice in estimation theory.

I am deeply grateful to my sisters Ruba, Heba and Sahar and my brother Awn, for their loving support. I owe my loving thanks to my wife Ramah and my beloved daughter Tasneem. Without their love, patience and encouragements I would not have finished this thesis. I wish to extend my deep and warm gratitude to my father

Sadeq and my mother Zeinat. They raised me, taught me, and always supported and loved me. To them I dedicate this thesis.

In conclusion, I acknowledge that this research would not have been possible without the financial support of THALES, Australia through the ARC Linkage grant (LP0561200)/APAI Scholarship. To them, I express my sincere gratitude.

To my father and mother.

Contents

List of Figures

List of Tables

List of Abbreviations

DCM-WD	Drift Corrected Measurement With Drift
DCM-WOD	Drift Corrected Measurement Without Drift
EKF	Extended Kalman Filter
EnKF	Ensemble Kalman Filter
FFT	Fast Fourier Transform
KF	Kalman Filter
IBRL	Intel Berkeley Research Laboratory
IMM	Interacting Multiple Model
R-WD	Reading With Drift
R-WOD	Reading Without Drift
SVM	Support Vector Machine
SVR	Support Vector Regression
UKF	Unscented Kalman Filter
UT	Unscented Transform
WSN	Wireless Sensor Network

Abstract

WIRELESS Sensor Networks (WSNs) are deployed for the purpose of monitoring an area of interest. Even when the sensors are properly calibrated at the time of deployment, they develop drift in their readings leading to erroneous network inferences. Traditionally, such errors are corrected by site visits where the sensors are calibrated against an accurately calibrated sensor. For large scale sensor networks, the process is manually intensive and economically infeasible. This imposes finding automatic procedures for continuous calibration. Noting that a physical phenomenon in a certain area follows some spatio-temporal correlation, we assume that the sensors readings in that area are correlated. We also assume that measurement errors due to faulty equipment are likely to be uncorrelated. Based on these assumptions, we follow a Bayesian framework to solve the drift and bias problem in WSNs.

In the case of densely deployed WSN, neighbouring sensors are assumed to be close to each other that they observe the same phenomenon. Hence, the average of their corrected readings is taken as a basis for each sensor to self-assess its measurement, estimate its drift and to correct the measurement using a Kalman Filter (KF) in the case of smooth drift, and the Interacting Multiple Model algorithm (IMM) in the case of unsmooth drift. The solutions are computationally simple, decentralised and also scalable. Any new node joining the neighbourhood needs only to obtain the corrected readings of its neighbours to find the average and apply the KF iterative procedure.

On the other hand, when the sensors are not densely deployed, Support Vector Regression (SVR) is used to model the interrelationships of sensor measurements

in a neighbourhood. This enables the incorporation of the spatio-temporal correlation of neighbouring sensors, to predict future measurements. The SVR predicted value is used by a KF to estimate the actual drift and correct the measurement. Unfortunately, the KF introduces some system errors when used with nonlinear systems. The use of Unscented Kalman filter (UKF) instead, considerably reduces the system error and results in a better drift correction. The use of IMM with the SVR-UKF framework allows for reducing the sampling rate which eventually reduces the communication overhead among the sensors and saves the communication energy.

In this thesis, we present several solutions for the random and systematic (drift and bias) errors in sensors measurements, for different sensor deployment scenarios. We also consider two drift scenarios, namely smooth and unsmooth drifts. We evaluate the presented algorithms on simulated and real data obtained from the Intel Berkeley Research Laboratory sensor deployment. The results show that our algorithms successfully detect and correct systematic errors (drift and bias) developed in sensors and filters out the noise. Thereby, prolonging the effective lifetime of the network.

Chapter 1

Introduction

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

On the down side, the wireless sensors are usually left unattended for long periods of time in the field, which makes them prone to failures. This is due to either sensors running out of energy, ageing or harsh environmental conditions surrounding them. Besides the random noise, these cheap sensors tend to develop drift in their measurements as they age. We define the drift as a slow, unidirectional long-term change in the sensor measurement. This poses a major problem for end applications, as the data from the network becomes progressively useless. An early detection of such drift is essential for the successful operation of the sensor network. In this process, the sensors, which otherwise would have been deemed unusable, can continue to be used, thus prolonging the effective life span of the sensor network and optimising the cost effectiveness of the solutions.

A common problem faced in large scale sensor networks is that sensors can suffer from bias in their measurements [3]. The bias and drift errors (systematic

errors) have a direct impact on the effectiveness of the associated decision support systems. Calibrating the sensors to account for these errors is a costly and time consuming process. Traditionally, such errors are corrected by site visits where an accurate, calibrated sensor is used to calibrate other sensors. This process is manually intensive and is only effective when the number of sensors deployed is small and the calibration is infrequent. In a large scale sensor network, constituted of cheap sensors, there is a need for frequent recalibration. Due to the size of such networks, it is impractical and cost prohibitive to manually calibrate them. Hence, there is a significant need for auto calibration [4] in sensor networks.

The sensor drift problem and its effects on sensor inferences have not been addressed thoroughly in the literature. We address this problem under the assumption that neighbouring sensors in a network observe correlated data, i.e., the measurements of one sensor is related to the measurements of its neighbours. Furthermore, the physical phenomenon that these sensors observe also follows some spatial correlation. Moreover, the faults of the neighbouring nodes are likely to be uncorrelated [5]. Hence, in principle, it is possible to predict the data of one sensor using the data from other closely situated sensors [5],[4]. This predicted data provides a suitable basis to correct anomalies in a sensor's reported measurements. At this point, it is important to differentiate between the measurement of the sensor or the reported data which may contain bias and/or drift, and the corrected reading which is evaluated by the error correction algorithms. The early detection of anomalous data enables us not only to detect drift in sensor readings, but also to correct it.

In this thesis, we present general and comprehensive frameworks for detecting and correcting both the systematic (drift and bias) and random errors in sensor measurements. The solutions address different sensor deployment scenarios. They also consider two types of drift, namely, smooth and unsmooth drifts. Statistical modelling rather than physical modelling is used to model the spatio-temporal cross correlations among sensors' measurements. This makes the frameworks presented here likely to be applicable to most sensing problems with minor changes.

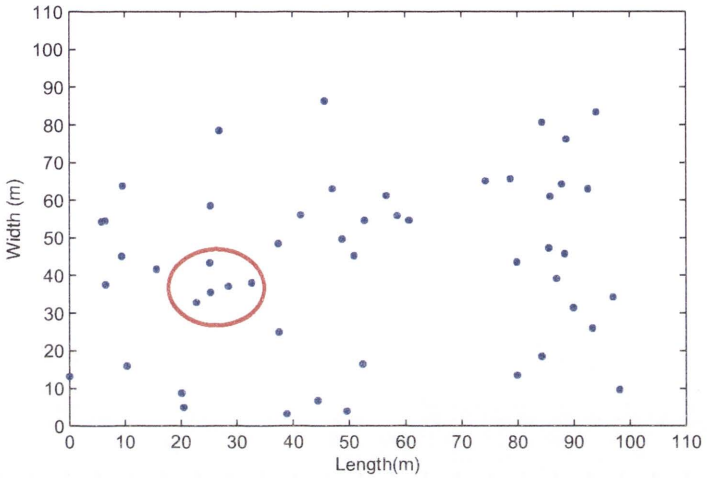


Figure 1.1: Wireless sensor area with encircled sub-network

1.1 Problem Statement

Consider a wireless sensor network with a large number of sensors distributed randomly in a certain area of deployment such as the one shown in Figure 1.1. The sensors are grouped in clusters (sub-networks) according to their spatial proximity. Each sensor measures a phenomenon such as ambient temperature, chemical concentration, noise or atmospheric pressure. The measurement, say temperature, is considered to be a function of time and space. As a result, the measurements of sensors that lie within the same cluster can be different from each other. For example, a sensor closer to a heat source or near direct sunlight will have readings higher than those in a shaded region or away from the heat source. An example of a cluster is shown using a circle in Figure 1.1. The sensors within the cluster are considered to be capable of communicating their readings among each other.

As time progresses, some nodes may start experiencing drift in their readings. If these readings are collected and used from these nodes, they will cause the users of the network to draw erroneous conclusions. After some level of unreliability is reached, the network inferences become untrustworthy. Consequently, the sensor network becomes useless. In order to mitigate this problem of drift, each sensor

node in the network has to detect and correct its own drift using the feedback obtained from its neighbouring nodes. This is based on the principle that the data from nodes that lie within a cluster are correlated, while their faults or drifts instantiations are likely to be uncorrelated. The ability of the sensor nodes to auto-detect and correct their drifts helps to extend the effective (useful) lifetime of the network. In addition to the drift problem, we also consider the inherent bias that may exist within some sensor nodes. There is a distinct difference between these two types of errors. The former changes with time and often becomes accentuated, while the latter, is considered to be a constant error from the beginning of the operation. This error is usually caused by a possible manufacturing defect or a faulty calibration.

The sensor drift that we consider in this work is of two types. The first is slow smooth drift that we model as linear and/or exponential function of time. The second type is unsmooth drift or drift with jumps. It is similar to the first; however, it suffers from sudden changes, surges or sharp peaks. Both of them are dependent on the environmental conditions, and strongly relate to the manufacturing process of the sensor. It is highly unlikely that two electronic components fail in a correlated manner unless they are from the same integrated circuit. Therefore, we assume that the instantiations of drifts are different from one sensor to another in a sensor neighbourhood or a cluster. Figures 1.2 and 1.3 show examples of the theoretical drift models for smooth drift and drift with jumps, respectively.

Consider a sensor sub-network that consists of n sensors deployed randomly in a certain area of interest. Without loss of generality, we choose a sensor network measuring temperature, even though this is generally applicable to all other types of sensors that suffer from drift and bias problems. Let T be the groundtruth temperature. T varies with time and space. Therefore, we denote the temperature at a certain time instance and sensor location as $T_{i,k}$ where i is the sensor number and k is the time index. At each time instant k, node i in the sub-network measures a reading $r_{i,k}$ of $T_{i,k}$. It then estimates and reports a *drift corrected* value $x_{i,k}$ to its neighbours. The corrected value $x_{i,k}$ should ideally be equal to the groundtruth temperature $T_{i,k}$. If all nodes are perfect, $r_{i,k}$ will be equal to the $T_{i,k}$, and the re-

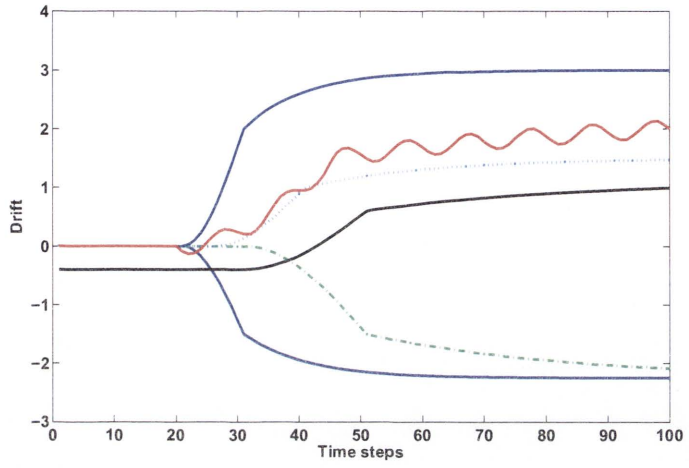


Figure 1.2: Examples of smooth drifts

ported values will ideally be equal to the readings, i.e., $x_{i,k} = r_{i,k}$.

To estimate the corrected value $x_{i,k}$, each node i first finds a predicted value $\widetilde{x}_{i,k}$ for its temperature as a function of the corrected measurements collected from itself and its neighbours in the previous time step using $\widetilde{x}_{i,k} = f(\{x_{j,k-1}\}_{j=1,j\neq i}^{n})$. Then it fuses this predicted value together with its measurement $r_{i,k}$ and the projected drift $d_{i,k}$ to result in a drift corrected sensor measurement $x_{i,k}$. In practice, each sensor reading comes with an associated random reading error (noise), and a drift $d_{i,k}$. This drift may be null or insignificant during the initial period of deployment, depending on the nature of the sensor and the deployment environment. The problem we address here is how to account for the drift in each sensor node i, using the predicted value $\widetilde{x}_{i,k}$, so that the reading $r_{i,k}$ is corrected and reported as $x_{i,k}$.

In chapters 3,4 and 5 we use the *average* of the neighbouring sensor corrected measurements to compute the $\widetilde{x}_{i,k}$, based on the assumption that the sensors are spatially close enough to have the same measurements (the network is densely deployed). We relax this assumption in the following chapters where $\widetilde{x}_{i,k}$ is computed using a support vector regression (SVR) modelled function that takes into account the temporal and spatial correlations of the sensor measurements. The SVR approximates $\widetilde{x}_{i,k}$ in chapter 6, using the current and previous corrected readings of

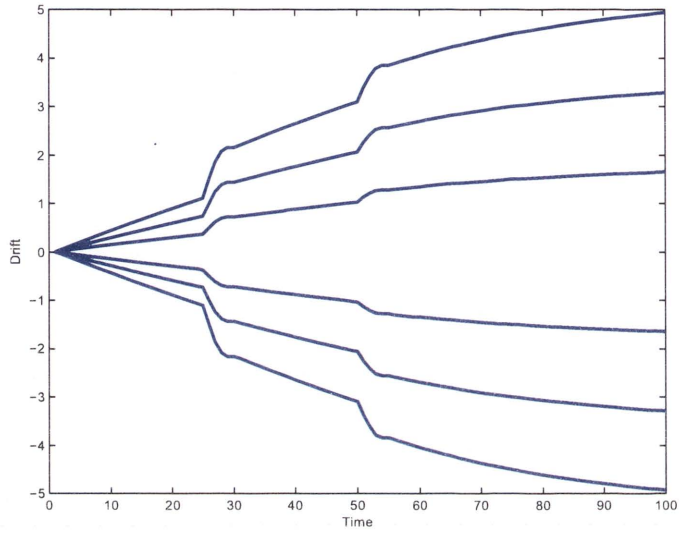


Figure 1.3: Examples of drifts with jumps and sudden changes

the neighbours $\widetilde{x}_{i,k} = f(\{x_{j,k-1}, x_{j,k}\}_{j=1,j\neq i}^{n})$, whereas in chapters 7 and 8, $\widetilde{x}_{i,k}$ is approximated using the previous corrected readings of all the sensors in the neighbourhood (cluster) excluding the sensor itself $\widetilde{x}_{i,k} = f(\{x_{j,k-1}\}_{j=1,j\neq i}^{n})$.

In the following section we explain the thesis structure and summarise our contributions.

1.2 Thesis structure and contributions

The structure of the thesis is organised as follows:

Chapter 2 defines WSNs, their applications and the most important challenges facing their wide use. It then surveys sensor faults, defines the drift, the bias, the calibration process and then reviews the single sensor based calibration techniques. An extensive survey of methods for detecting and correcting sensor faults in WSN is given, and a comparison between the algorithms available in literature is made.

Chapter 3 addresses the problem of systematic errors (drift and bias) in sensor measurements and explains their effects on the inferences made by the sensor net-

works. It also emphasises that the special characteristic of WSNs: that they consist of cheap unreliable sensors and establishes the need for frequent calibration. The chapter argues that using the correlation among neighbour sensor readings and depending on the assumption that sensor drifts are unlikely to be correlated; an autocalibration algorithm can be devised.

The main contributions of this chapter are:

- Introducing a simple algorithm for autocalibration of WSNs sensors under the simple scenario of close deployment of the sensors to observe the same data.
- Showing how the use on an autocalibration algorithm will extend the effective life of the network.
- Showing the effect of the communication channel reliability, the number of sensors in the cluster and the drift timing on the performance of the drift correction algorithm.

Chapter 4 presents a formal Bayesian framework for estimating sensor bias and smooth drift, and correcting the sensor measurements in a densely deployed WSN. The proposed solution exploits the notion that measurement errors due to faulty equipment are likely to be uncorrelated. The sensors in the neighbourhood are assumed to be densely deployed. Hence, the average of their corrected readings is taken as a basis for each sensor to self assess. The Bayesian formulation leads to a decentralised and computationally simple single state Kalman Filter (KF) iterative procedure, allowing its implementation in WSNs. The presented solution is also scalable. A new node joining the neighbourhood only needs to obtain information from its neighbouring sensors and then find the average and apply the KF based iterative procedure. Similarly, any sensor excluded from the neighbourhood will not affect the others as they use the average.

The main contributions of this chapter are:

- Introducing a formal Bayesian framework for estimating sensor bias and smooth drift, and correcting the sensor measurements in a densely deployed WSN.

- Showing that the Bayesian formulation leads to a scalable decentralised and computationally simple single state KF iterative procedure that uses the average of the sensors readings in the cluster as a basis for assessing the measurements of the sensors in that cluster.

Chapter 5 introduces a formal statistical procedure for estimating unsmooth drifts and correcting sensor measurements in a densely deployed WSN. As the sensors in the neighbourhood are assumed to be densely deployed, the average of their corrected readings is taken as a basis for each sensor to self-assess. The essential idea brought forth by the solution presented in this chapter, is the use of the Interacting Multiple Model algorithm (IMM) instead of KF to catch the fast changes and the jumps in the drift behaviour. The solution is designed to capture both smooth drifts and unsmooth drifts that have jumps and sudden escalations. It is also computationally simple as it is decentralised dealing with single state IMM based algorithm. The presented solution is also scalable. Any new node joining the neighbourhood only needs to obtain information from its neighbouring sensors and then find the average and apply the IMM based iterative procedure. Similarly, any sensor excluded from the neighbourhood will not affect the others as they use the average. The chapter shows that the IMM based algorithm performs better than the KF based solution in detecting and correcting both the smooth drift and unsmooth one, however, at the cost of the computational complexity.

The main contributions of this chapter are:

- Introducing the use of the IMM algorithm to better deal with fast changes and the jumps in the drift behaviour, as the standard KF does not efficiently respond to fast changes in the dynamics.
- Showing that the IMM based algorithm performs better (in our context) than the KF based solution in estimating unsmooth drift, however, at the cost of the computational complexity.

Chapter 6 presents a formal statistical procedure for estimating sensor drifts and correcting sensor measurements in a non-densely deployed WSN. The solution introduces the novel idea of using Support Vector Regression (SVR) to model the

interrelationships of sensor measurements in a neighbourhood in a non-densely deployed WSN. This enables the incorporation of the spatio-temporal correlations of neighbouring sensors to predict future measurements. The prediction is used by a KF to estimate the drift in the reading of the sensor under consideration. The algorithm runs recursively and is fully decentralised. Chapter 6 also demonstrates using real data obtained from the Intel Berkeley Research Laboratory (IBRL) that the algorithm successfully suppresses the drifts developed in sensors. However, it also shows that the KF introduces some system error that accumulates with time and refers that error to the use of the KF with nonlinear system.

The main contributions of this chapter are:

- Introducing the novel idea of using SVR to model the interrelationships of sensor measurements in a neighbourhood in a non-densely deployed WSN. This enables the incorporation of the spatio-temporal correlations of neighbouring sensors, to predict future measurements. The prediction is used by a KF to estimate the drift in the reading of the sensor under consideration.
- Using the general framework of statistical modelling (using SVR) rather than physical modelling, to model the spatio-temporal cross correlations among sensors, which in principal, seems to be applicable to most sensing problems with minor changes.

Chapter 7 introduces a solution for the non-densely deployed WSN that solves the problem of system error accumulation of chapter 6. The solution may seem to be similar to the SVR-KF framework of chapter 6, whereas, it is substantially different. Instead of the KF, Unscented Kalman Filter (UKF) is used. The use of UKF reduces the system error noticed in the SVR-KF evaluations since it is a better method for estimating the mean and the variance of a random variable propagating through nonlinear systems. The estimated variable here is the actual temperature whereas in chapter 6 it is the drift. Similar to chapter 6, SVR is used to model the interrelationships of sensor measurements in a neighbourhood. This enables the incorporation of the spatio-temporal correlations of neighbouring sensors, to predict future measurements. The prediction is used by a UKF to estimate the actual value of the

measured variable (here temperature) at the sensor under consideration. The algorithm runs recursively and is fully decentralised. No assumptions regarding the linearity of drift or the density (closeness) of sensor deployment are made. Evaluations for the performance of the algorithm using real data obtained from the IBRL show that the algorithm successfully suppresses the drifts developed in sensors and thereby prolongs the effective life of the network. However, it is noticed that a major source of error in temperature estimation and drift correction method (though small) is the sudden steep change in the measured variable. This is dealt with in this chapter by increasing the sampling rate of the data in a trade-off with the communication overhead.

The main contributions of this chapter are:

- Introducing the use of UKF to reduce the accumulating system error noticed in the SVR-KF evaluations since UKF is a better method for propagating the mean and the variance of a random variable through nonlinear systems.
- Extensive evaluation of the algorithm under several scenarios showing that the algorithm is reliable and successfully suppresses the drifts developed in sensors and thereby prolongs the effective life of the network.

Chapter 8 proposes the use of the IMM algorithm with the SVR-UKF framework presented in chapter 7, to overcome the error in the temperature estimation caused by the sudden steep changes in the measured data. The IMM-SVR-UKF framework serves as an alternative for increasing the sampling rate of the data, in order to reduce the effect of the jumps on the accuracy of the estimated readings. Besides its ability to follow (track) data that suffer from sharp changes and sudden jumps, the IMM-SVR-UKF framework can deal with jumps in the readings caused by lowering the sampling rate. Consequently, this will cause less communication overhead between sensors to maintain the calibration, and eventually reduce the energy consumed from the batteries. The evaluations conducted using IBRL data show that the IMM-SVR-UKF performs better than SVR-UKF for half sampling rate of the data.

The main contributions of this chapter are:

- Introducing the use of the IMM algorithm with the SVR-UKF framework to overcome the error in the temperature estimation caused by the sudden steep changes in the measured data.
- Allowing for reducing the sampling rate and thereby reducing the energy consumed in communication among the neighbouring sensors.

Finally, chapter 9 concludes the thesis and presents the future research directions.

1.3 Publications arising from this thesis

Journal Papers

- M. Takruri, S. Rajasegarar, S. Challa, C. Leckie, and M. Palaniswami, "Spatio-Temporal Modelling Based Drift Aware Wireless Sensor Networks," to appear in the International Journal in Distributed Sensor Networks, 2010.
- M. Takruri, S. Challa, and R. Chakravorty, "Recursive Bayesian Approaches for Auto Calibration in Drift Aware Wireless Sensor Networks," to appear in the the Journal of Networks, Academy Publisher, 2010.

Book Chapter

- M. Takruri, K. Aboura, and S. Challa, "Distributed recursive algorithm for auto calibration in drift aware wireless sensor networks," in Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering (K. Elleithy, ed.), pp. 21-25, Springer, 2008.

Conferences

- M. Takruri, S. Challa and R. Yunis, " Data Fusion Techniques for Auto Calibration in Wireless Sensor Networks," Proceedings of the 12th International conference on information fusion (Fusion 2009), Seattle, USA, July, 2009.
- M. Takruri, S. Challa, and R. Chakravorty, "Auto calibration in drift aware wireless sensor networks using the interacting multiple model algorithm," Mosharaka International Conference on Communications, Computers and Applications MIC-CCA 2008, Amman, Jordan, August, 2008.

- M. Takruri, S. Rajasegarar, S. Challa, C. Leckie, and M. Palaniswami, "Online drift correction in wireless sensor networks using spatio-temporal modelling," Proceedings of the 11th International conference on information fusion (Fusion 2008), Cologne, Germany, July, 2008.
- M. Takruri and S. Challa, "Drift aware wireless sensor networks," Proceedings of the 10th the International conference on information fusion (Fusion 2007), Quebec City, Canada, July, 2007.

Chapter 2
Literature Review

THIS chapter introduces Wireless Sensor Networks (WSNs) and some of their intended military and civilian applications along with some discussion on some of the most important challenges facing the wide spread of these networks. The chapter focuses on the fault tolerance as a major problem in WSNs. It reviews most of the common sensors faults addressed in literature, especially, the drift and bias. It then defines the calibration process and summarises some single sensor based calibration techniques. Finally it surveys several methods for detecting/ detecting and correcting sensor faults in WSN and gives comparisons among the algorithms available in literature.

2.1 Wireless Sensor Networks

WSNs are an important and promising field of research that have a lot of prospective applications in all aspects of our lives. A WSN usually consists of cheap sensors with limited processing capabilities and energy resources. However, as a group they can accomplish more complex sensing tasks than the expensive individual traditional sensors. Moreover, they can be deployed in inaccessible areas without the need for carefully engineering their position of deployment and their communication topology [2].

Wireless sensor nodes are small in size and able to sense, process data, and communicate with each other to transfer information to the interested users. Typically, a sensor node consists of four sub-systems [6, 7]:

- Computing sub-system (processor and memory): responsible for processing the collected data, controlling the sensors and the execution of communication protocols.
- Communication sub-system (transceiver): used to communicate with neighbouring nodes and the outside world.
- Sensing sub-system (sensing elements) and
- Power supply sub-system (battery): supplies power to the node.

Sensor networks are usually composed of a large number of sensor nodes, which are deployed randomly either inside the phenomenon or very close to it [2]. They represent a collection of self-organised sensor nodes that form a temporary network. Neither pre-defined network infrastructure nor centralised network administration exists. This also requires that sensor network protocols and algorithms possess self-organising capabilities. The wireless nodes communicate with each other via radio links. Due to the limited transmission range, distant nodes wishing to communicate with other nodes employ a multi-hop strategy for communicating, causing each node to simultaneously act as a router and as a host [2, 8].

The inferences made by the WSN are a result of the cooperative effort of sensor nodes [9]. Sensor nodes collect data from their respective environments and send their measurements after being half-processed, to a fusion centre where the inferences and conclusions are made and the corresponding actions are decided [10]. Processing the data by the sensor node before being sent to the fusion centre reduces the communication overhead and the bandwidth required for communication in the network.

WSNs have many prospective civilian and military applications. They can be deployed in controlled environments, for applications such as sensing the atmosphere in buildings and factories [11], or they can be spread in hazardous and hostile environments such as battlefields [2]. Originally motivated by surveillance in battlefields for the military, interest in WSNs spread over a wide range of applications, from scientific exploration and monitoring, for example, the deployment of a WSN on an active volcano [12, 13], to monitoring the microclimate throughout the

volume of redwood trees [14], to building and bridge monitoring [15,16], to healthcare monitoring [17] and a number of other applications such as home automation [18], target tracking [19], detecting and monitoring car thefts [20], vehicle tracking and detection [21]. A key application for WSNs is environmental monitoring [2,22].

Currently, the trend is to move from the centralised ordinary WSNs to the distributed and decentralised WSNs [10]. Distributed and decentralised WSNs outweigh ordinary sensor networks by the fact that being less centralised, more processing, analysis and understanding of observed phenomena are conducted on the sensor nodes level without referring to the server. This results in reducing the bandwidth and energy requirements. Examples of such networks are given in [23–25]. A review of recent developments in distributed WSNs relating to the network structure, the data processing paradigm, the sensor fusion algorithms and the optimal sensor deployment strategies is given in [26].

Typically, a WSN consists of large number of sensor nodes left unattended for long periods of time. This makes them prone to failures due to either lack of energy or due to the harsh environmental conditions surrounding them [4]. This emphasises the importance of implementing fault tolerant algorithms for the successful deployment of WSNs [5]. On the other hand and due to the need to take more data for a certain phenomenon, sensor nodes are to be added to the network. In both cases the network should be able to adapt with the changes. This highlights scalability as a key issue in WSN research [22]. Another important challenge is the limited energy resources [2].

To address the energy problem in WSNs, researchers suggested equipping the sensor nodes with power scavenging methods [27] such as solar cells, or implementing decentralised and distributed algorithms [28–30]. Scalability of WSN received special attention in WSN literature. Weng et al proposed a method for dynamic adaptation of sensor node architectures to improve the scalability of sensor networks [31]. Chiassserini and Rao tackled the scalability problem from another direction. They introduced a collaborative computational algorithm and communication scheme that makes sensors operate as a distributed digital signal processor.

The algorithm was implemented on a WSN to calculate Fast Fourier Transform (FFT) in collaboration among all sensors of the network. This resulted in overcoming the energy and computational limitations of an individual sensor and improving significantly the energy efficiency of the overall network [24]. A more complex system named *IRISNET* was implemented by Intel Research Pittsburgh. It clearly showed the effect of decentralisation of the WSN on improving the scalability in high bit rate multimedia sensors (mainly video cameras) [25, 32, 33]. The fault tolerance in WSNs is the main focus of this thesis. The coming sections will give an overview of sensor faults and the available techniques in literature addressing sensor calibration both on the sensor scale and the WSN scale.

2.2 Sensor Faults, Drift, Bias and the Calibration Problem

The purpose of wireless sensor networks is to deploy low cost sensors with sufficient computing and communication capabilities to support networked sensing applications. The emphasis on lower cost led to sensors that are less accurate and less reliable and suffer from several types of faults. This affects the inferences and the decisions made by the network and shortens its operational life. For a better performance of the sensor network, the sensors should be continuously calibrated. The calibration process is mapping the sensor reading to the true value of the measured phenomenon by knowing the function that relates them. The function can be linear or nonlinear depending on the characteristics of the sensor itself. The sensors failures are assumed to be uncorrelated. This is a reasonable assumption since these failures are primarily due to imperfections in the manufacturing process and not a function of the nodes' spatial deployment [5]. Following, we give an overview of these failures form the sensors literature:

Koushanfar et al. in [34] introduced taxonomy for classification of faults in sensor networks. First they defined the bias fault as the offset in the readings and then they defined drift as a generalisation of the bias model were the correct value of the measurement is subject to alteration that is time invariant function of the

correct value. Other types of error models they mentioned are the error when the reading of a sensor is frozen on a particular reading and when the sensor is dead and does not report any values. They also mentioned another two error models and specified them as more complex error models. The first is characterised by the degradation in the sensor readings variance often due to ageing of the sensor and the other error model is characterised by the transformation of the sensor readings by a particular function that may or may not be dependent on the sensor readings.

Laura Balzano in her master thesis [35] categorised the sensor faults into 5 types, namely, offset fault, gain fault, variance degradation fault, stuck-at-fault and static fault. She modelled the relationship between the value reported by the sensor and the actual value (groundtruth) as linear relationship given in (2.1).

$$y = \beta_0 + \beta_1 x + \epsilon \tag{2.1}$$

Where y is the sensor reading, x is the actual value and ϵ is an additive noise. β_0 represents the offset fault and β_1 represents the gain fault. The variance degradation fault is caused by the increase in the sensor readings variance due to the degradation of the sensing element. It is modelled in (2.1) by increasing the variance of the noise component. Similar to the frozen fault in [34], the stuck-at-fault is characterised by the no-change state in the readings irrespective of the changing measured variable. Finally, in the static fault, the sensor reports highly noisy measurements that are not related at all to the measured phenomenon.

An experimental study by [36], on the porous silicon humidity sensors showed that they suffer from long term drifts in their measurements. The authors related the drift to the ageing in the sensor's material which is an irreversible change in the material properties over time. Unlike the assumption in [35], the results showed that on the long run, the drift has a nonlinear response with time. However, on the short term, they stated that it can be modelled as linear.

Similarly, in another two studies on the tin oxide gas sensors [37, 38], it was stated that tin oxide gas sensors suffer from long term drift that causes significant temporal variations in the sensor response when exposed to the same gases under

identical conditions. The two studies modelled the drift in tin oxide gas sensors as linear function with time.

In this work we address two types of sensor faults, namely the drift fault and the bias fault. We define the drift as an irreversible long term change in the sensor readings that is a function of time. It is usually caused by the ageing of the sensing element or by the harsh environmental conditions the sensor is deployed in. The drift usually is a slow function of time. We call this type of drift as smooth drift and we model it as mixture of slow exponential functions, linear and/or slow sinusoidal functions that can be linearized for small time intervals. Examples of smooth drifts are shown in figure 1.2. Another type of drift we consider is the unsmooth drift. We sometimes refer to it as smooth drift with jumps. It is similar to the smooth drift; however, it suffers from sudden changes, surges or sharp peaks. Examples of unsmooth drifts are shown in figure 1.3. As mentioned earlier, the drifts are dependent on the environmental conditions, and strongly related to the manufacturing process of the sensor. It is highly unlikely that two electronic components fail in a correlated manner unless they are from the same integrated circuit. This implies that drifts of the neighbouring sensors in a sensor network may be considered uncorrelated, or in other words, the instantiation of drift differs from one sensor to another.

The inherent bias is the error in the sensor reading at the very first instant of deployment. The distinct difference between the drift error and the bias error is that the former changes with time and often becomes accentuated, while the latter, is considered to be a constant error from the beginning of the operation. Therefore, it can be said that the drift at time zero is equal to the bias.

Unlike Balzano's assumption in [35] of the constant offset term in equation (2.1), we consider the offset (here the drift) to be a nonlinear function of time and we introduce frameworks to evaluate it continuously. In [35] the offset and gain are evaluated once and the relationship is used to map the readings into the true value. This is a great solution if the deviations are linear; however, if the long term relationship is nonlinear as in [36], it will give erroneous results.

Although the focus of this thesis is on finding auto calibration solutions for sensors in the context of wireless sensor networks, we survey below some solutions for calibrating individual sensors. These solutions are interesting to us here since they use machine learning techniques for the continuous calibration of the sensors.

The authors of [37] proposed a solution for the classification of gases using a self organising map neural network with Tin Oxide gas sensor. The network was trained on the sensor response for several gases and was able to classify them into distinct clusters. However, when the neural network was tested on drifting sensors, it resulted in erroneous classification of the gases. To compensate for the drift effect, the authors proposed that keeping the neural network to learn during the normal operation phase (testing phase) with very low learning rate and zero neighbourhood extension would dramatically improve the recognition of the gases.

Having studied the response of the porous silicon humidity sensors noticing that it suffers from nonlinear long term drift and other nonlinearities, the authors of [36] used a neural network to account for the drift and recover the groundtruth humidity values. The neural network was trained (in a supervised fashion) with the actual sensor output as an input, and the linear drift free response as the desired output. The training was repeated on sensor data that suffered from drift. Finally the trained network was tested and showed the ability to successfully recover the actual humidity from the drifting sensor measurements

Wang and Ye in [39] stated that the porous silicon humidity sensors usually developed errors in their measurements due to the hysteresis effect and nonlinearities (including nonlinear drift) in their responses. To solve the problem, they introduced a framework comprising a cascade of two Support Vector Regression (SVR) blocks. The first SVR block was used to compensate for the hysteresis effect whereas the second was used to compensate for nonlinearities. The authors claimed that the periodic training of the SVR resulted in compensating the long term drift.

Having introduced sensor faults, the calibration problem and addressed our drift and bias problem, we give in the following section a survey of the techniques

and methods found in literature that address the problems of fault detection and fault detection/correction in sensor networks.

2.3 Related Work

On their work addressing sensor faults, Koushanfar et al. introduced in [34] a cross validation based technique for online detection (but not correction) of sensor faults. They applied the approach on different fault models. The key idea of their approach was to compare the results of multi-sensor fusion with and without each of the sensors involved, and to use non-parametric statistical techniques to identify sensors that had the highest probability to be faulty. They modelled the multi-modal sensor fusion by a set of nonlinear equations and then translated the system into an instance of nonlinear function minimisation and obtained a solution by applying Powell method. The method was applied on a network of light sensors and in the presence of random noise. It showed better detection results for low number of faulty sensors.

Another work addressing sensor faults in wireless sensor networks was given in [40]. The paper proposed a way for identifying faulty (crashed) nodes through the information gathered by the operational sensors of the network, which can be used then by the algorithm to detect the faulty sensors. According to the scenario given in the paper, the detected faulty sensors had to either be replaced or their batteries be changed by the network operator. Thus, extending the lifetime of the network. The solution given in [40] seems to be impractical to many large-scale sensor networks applications where the sensors are left unattended and are required to automatically detect and correct their errors.

Closely related areas to the context of this thesis in wireless sensor networks literature are anomaly or outlier detection and trust management in sensor networks. [41] gives a recent survey on anomaly detection and discusses the state-of-the art techniques used in the field. The paper highlights the importance of detecting anomalies for tasks such as intrusion detection, monitoring applications and de-

tecting faulty sensors. It also emphasises that the key challenge for the anomaly detection techniques is minimising the energy consumption and the communication requirements by exploiting distributed in-network processing. Several techniques are currently used for anomaly detection in sensor networks. Of these are data clustering approaches [42, 43] and support vector machine approach [44]. The most relevant application of anomaly detection to our work is detecting faulty sensors and the most relevant technique is the use of support vector machines to detect sensors that report faulty data.

The trust problem focuses on detecting sensors that report malicious data or do not report at all due to either a fault in the sensor itself or a problem in the communication link. It builds a reputation for each sensor along time. It also labels sensors with malicious behaviours and then removes them from the network. Most of the work done in the field of trust management in sensor networks considered the effect of communication on trust. This resulted in binary modelling of the trust depending on whether the communication link is reliable or not [45, 46]. Besides the communication effect, Momani et al. in [47] emphasised that the major component of trust in sensor networks is related to the reliability of the data reported by the sensors. They added that this is substantial to the sensor networks as the inferences made by the network not only depend on the communication among the sensors, but also on how true the reported data are. Therefore, they presented in [48] a novel methodology that combines the data component and the communication component in one continuos model as opposed to the previous binary models.

An algorithm for detecting and correcting faults in sensor measurements in an application specific context was introduced by Krishnamachari and Iyengar in [5]. They proposed a distributed Bayesian algorithm for fault-tolerant event region detection in WSNs. They addressed the challenge of distinguishing between faulty sensor measurements and unusual environmental conditions by exploiting the notion that measurement errors due to faulty equipment are likely to be uncorrelated, while environmental conditions are spatially correlated. To simplify the problem, they considered a binary fault-event disambiguation problem i.e. a node either

reports a reading indicating a normal value or an event. The reported value of an event can be an actual event or a false alarm caused by a sensor fault. In order to decide whether its reading is true or not, the sensor collects the neighbouring sensors readings. If the number of neighbouring sensors (or the average of the neighbouring sensors readings) reporting the same value is above than a certain threshold, then the sensor reading is true. Otherwise, the sensor is considered faulty and its reading is disregarded. The optimal threshold decision scheme for their average-correlation model was proved to be that half of the neighbouring sensors. The authors suggested that algorithm could be generalised to the correction of real-valued sensor measurement errors i.e. nodes in a sensor network should be able to exploit the spatial correlation of environmental readings to correct for the noise in their readings considering continuous noise models not binary 0-1 failures.

The sensor bias and drift problems and their effects on sensor inferences have not been addressed thoroughly in the sensor networks literature. In contrast, the bias correction problem has been well studied in the context of the multi-radar tracking problem. In the target tracking literature the problem is usually referred to as the *registration problem* [49],[50]. When the same target is observed by two sensors (radars) from two different angles, the data from those two sensors can be fused to estimate the bias in both sensors. In the context of image processing of moving objects, the problem is referred to as *image registration*, which is the process of overlaying two or more images of the same scene taken at different times, from different viewpoints, and/or by different cameras. It geometrically aligns two images: the reference and sensed images [51]. Image registration is a crucial step in all image analysis tasks in which the final information is gained from the combination of various data sources like in image fusion [52]. That is, in order to fuse two sensor readings, in this case two images, the readings must first be put into a common coordinates systems before being fused. The essential idea brought forth by the solution to the registration problem is the augmentation of the state vector with the bias components. In other words, the problem is enlarged to estimate not only the states of the targets, using the radar measurements for example, but also the bi-

ases of the radars. This is the approach we consider in the case of sensor networks. Target tracking filters, in conjunction with sensor drift models are used to estimate the sensor drift in real time. The estimate is used for correction and as a feedback to the next estimation step. The presented methodology is a robust framework for auto calibration of sensors in a WSN.

A straightforward approach to bias calibration is to apply a known stimulus to the sensor network and measure the response. Then comparing the groundtruth input to the response will result in finding the gain and offset for the linear drifts case [53]. This method is referred to by [54] as non-blind calibration since the groundtruth is used to calibrate the sensors. Another form of non-blind calibration is manually calibrating a subset of sensors in the sensor network and then allowing the non-calibrated sensors to adjust their readings based on the calibrated subset. The calibrated subset in this context form a reference point to the groundtruth [3, 55]. The above mentioned methods are impractical and cost prohibitive in the case of large scale sensor networks.

The calibration problem of the sensor network was also tackled by [54, 56] in a different fashion. They stated that after sensors are calibrated to the factory settings, when deployed, their measurements will differ linearly from the groundtruth by certain gains and offsets for each sensor. They presented a method for estimating these gains and offsets using subspace matching. The method only requires routine measurements to be collected by the sensors and does not need groundtruth measurements for comparison. They referred to this problem as blind calibration of sensor networks. So by identifying these gains and offsets (which they assumed to be constant for each sensor), the future readings of the sensors can be mapped to the true values. The method does not require dense deployment of the sensors or a controlled stimulus. However, It requires that the sensor measurements are at least slightly correlated over space i.e. the network over samples the underlying signals of interest. The theoretical analysis of their work does not take noise into consideration and assumes linear calibration functions. Therefore, the solution may not be robust in noisy conditions and will probably result in wrong estimates if applied in

a scenario where the relationship between the measurement and the groundtruth is nonlinear. The evaluations they presented showed that the method worked better in a controlled environment.

An earlier work on blind calibration of sensor nodes in a sensor network was presented in [3,55]. They assumed that the sensors of the network under consideration were sufficiently densely deployed that they observed the same phenomenon. They used the temporal correlation of signals received by neighbouring sensors when the signals were highly correlated to derive a function relating the bias in their amplitudes. Another method for calibration was considered by [57]. They used geometrical and physical constraints on the behaviour of a point light source to calibrate light sensors without the need for comparing the measurement with an accurate sensor (ground truth). They assumed that the light sensors under consideration suffered form a constant bias with time.

The authors in [58, 59] argued that calibrating the sensors in sensor networks is a problematic task since it comprises large number of sensor that are deployed in partially unobservable and dynamic environments and may themselves be unobservable. They suggested that the calibration problem in sensor/actuator networks should be expressed as a parameter estimation problem on the network scale. Therefore, instead of calibrating each sensor individually to optimise its measurement, the sensors of the network are calibrated to optimise the overall response of the network. The joint calibration method they presented calibrated sensors in a controlled environment. The method was tested on an ad-hoc localisation system and resulted in reducing the error in the measured distance from 74.6% to 10.1%. The authors claimed that the joint calibration method could be transformed into an auto calibration technique for WSNs in an uncontrolled environment i.e. some form of blind calibration where the value of the groundtruth measurement (here the distance) is unknown. They formulated the problem as a quadratic programming problem. Similar to [58,59], blindly calibrating range measurements for localisation purposes between sensors using received signal strength and/or time delay were considered in [60,61].

The work of [62] aimed to reduce the uncertainties in the sensors readings. It introduced a Bayesian framework for online cleaning of noisy sensor data in WSNs. The solution was designed to reduce the influence of random errors in sensors measurements on the inferences of the sensor network but did not address systematic errors. The framework was applied in a centralised fashion and on synthetic data set and showed promising results.

The author of [35] described a method for in-situ blind calibration of moisture sensors in a sensor network. She used the Ensemble Kalman Filter (EnKF) to correct the values measured by the sensors, or in other words, to estimate the true moisture at each sensor. The state equation was governed by a physical model of moisture used in environmental and civil engineering and the measurements were assumed to be related to the real state by a certain offset and gain. The state (moisture) vector was augmented with the calibration parameters (gain and offset) and then the gains and offsets were estimated to recover the correct state from the measurements.

Another method for detecting a single sensor failure that is a part of an automation system (a sort of wired sensor network) was proposed by [63]. Using the incoming sensor measurement, a model for the sensor behaviour was constructed and then optimised using an online maximum likelihood algorithm. Sensor readings were compared with the model. In event that the sensor reading deviated from the modelled value by a certain threshold, the system labelled this sensor as faulty. On the other hand, when the difference was small, the system automatically adapted to it. This made the system capable of adapting to slow drifts.

A neural network-based instrument surveillance, calibration and verification system for a chemical processing system (a sort of wired sensor network) was introduced in [64]. The neural network used the correlation in the measurements of the interconnected sensors to correct the drifting sensors readings. The sensors that were discovered to be faulty were replaced automatically with the best neural network estimate thus restoring the correct signal. The performance of the system depended on the degree of correlation of the sensors readings. It was also found that the robustness of the monitoring network was related to the amount of signal

redundancies and the degree of signal correlations. The authors concluded that their system could be used to continuously monitor sensors for faults in a plant. However, they noted that retraining the entire network may be necessary for major changes in plant operating conditions

Support Vector Machines (SVM) were used in [44] to detect anomalies and faulty sensors of a sensor network. The data reported by the sensors were mapped from the input space (the space where the features are observed) to the feature space (higher dimensional space) using kernels. The projected data were then classified into clusters and the data points that did not lie in a normal data cluster were considered anomalous. The sensor that always reported anomalous data was considered faulty.

The authors of [65] presented a method for in-network modelling of sensor data in a WSN. The method used kernel linear regression to fit functions to the data measured by the sensors along a time window. The basis functions used were known by the sensors. Therefore, if a sensor knew the weights of its neighbour, it would be able to answer any query about the neighbour within the time window. So instead of sending the measured data of the whole window period from one sensor to another, sending the weights would considerably reduce the communication overhead. This was one of the aims of the method. The other aim was to enable any sensor in the network to estimate the measured variable at points within the network where there were no sensors using the spatial correlation in the network. An application for the introduced method is computing contour levels of sensor values as in [66]. Even that the work in [65] considered the unreliable communication between distant sensors and the noise in sensor readings, it did not address the systematic errors (drift and bias) which can build up along time and propagate among sensors causing the continuously modelled functions to produce estimates that deviate from the groundtruth values.

In addition to its superb capabilities in generalisation, function estimation and curve fitting, SVR is used in other applications such as forecasting and estimating the physical parameters of a certain phenomenon. Wang et al. in [67] used SVR

in medical imaging for nonlinear estimation and modelling of functional magnetic resonance imaging (fMRI) data to reflect their intrinsic spatio-temporal autocorrelations. Moreover, Kashif et al. in [68] used SVR to successfully predict the ground moisture at a site using meteorological parameters such as relative humidity, temperature average solar radiation, and moisture measurements collected from spatially distinct locations. A similar experiment to predict ground moisture was conducted by Gill et al. in [69]. In addition to using the SVR to predict the moisture measurements ahead in time, they introduced the use of an EnKF to correct or match the predicted values with the real measurements at certain points of time (whenever measurements are available) to keep the predicted values close to the measurements taken on site and eventually reduce the prediction error.

The above survey, has introduced most of the work undertaken in the area of fault detection and fault detection/correction in wireless sensor networks. This research approaches the problem in a more comprehensive manner resulting in several novel solutions for detecting and correcting drift and bias in WSNs. It does not assume linearity of the sensor faults (drift) with time and addresses smooth drifts and drifts with sudden changes and jumps. It also considers the cases when the sensors of the network are densely and non densely deployed. Moreover, it introduces recursive online algorithms for the continuous calibration of the sensors. In addition to all of that, the solutions presented are decentralised to reduce the communication overhead. Some of the papers that have arisen from this research are surveyed below: The idea of drift aware wireless sensor network that detects and corrects sensors drifts and eventually extends the functional life time of the network was first introduced in [4]. A formal statistical procedure for tracking and detecting smooth sensors drifts using decentralised Kalman Filter (KF) algorithm in a densely deployed network was introduced in [70,71]. The sensors of the network were close enough to have similar temperature readings and the average of their measurements was taken as a sensible estimate to be used by each sensor to self-assess. As an upgrade for this work, the KFs were replaced in [71,72] by interacting multiple model (IMM) based filters to deal with unsmooth drifts. A more general

solution was considered in [73]. The assumption of dense sensor deployment was relaxed. Therefore, each sensor in the network ran an SVR algorithm on its neighbours' corrected readings to obtain a predicted value for its measurements. It then used this predicted data to self-assess its measurement, detect (track) its drift using a KF and then correct the measurement.

Finally, a more robust and reliable decentralised algorithm for online sensor calibration in wireless sensor networks was presented in [74]. The algorithm represents a substantial improvement of method in [73]. By using an Unscented Kalman Filter (UKF) here instead of the KF, the bias in the estimated temperature (system error) was dramatically reduced compared to that reported in [73]. This is justified by the fact that UKF is a better approximation method for propagating the mean and covariance of a random variable through a nonlinear transformation than the KF is. Unlike the work in [35], statistical modelling rather than physical relations was used to model the spatio-temporal cross correlations among the sensors measurements. Similar to [73], statistical modelling was achieved by applying SVR. This in principal made the framework applicable to most sensing problems without needing to find the physical model that describes the phenomenon under observation, and without the need to abide by the constraints of that physical formulation. The algorithm runs recursively and is fully decentralised. It does not make assumptions regarding the linearity of the drifts as opposed the work in [54] . The implementation of the algorithm on real data obtained from the Intel Berkeley research laboratory (IBRL) showed a great success in detecting and correcting sensors drifts and extending the functional lifetime of the network.

As previously mentioned the idea of using regression to model spatio-temporal correlation among the sensors of the network was used by [65]. This is in principal similar to the approach followed in [74] and [73]. Contrary to them, the work in [65] ignored the systematic errors that are inherent in this type of sensors. In fact, the main purpose of the methods presented in [74] and [73] was to detect and correct these errors.

The work in [74] and [73] that is based on an SVR-UKF framework may also seem to be similar the research of Gill et al. in [69]; however, the objective is totally different. In their work, Gill et al. used SVR to predict ground moisture measurements ahead in time and used the EnKF to correct or match the predicted values with the real measurements at certain points of time (whenever measurements are available) to keep the predicted values close to the measurements taken on site and eventually reduce the prediction error. In contrast, [74] and [73] dealt with a WSN that comprises a number of cheap sensors and tend to develop drift. Their use of an SVR-UKF framework was to correct the readings of the sensors using the predicted values given by a well-trained SVR. So in principal, the scheme of Gill et al. trusted the measurements of the sensors more than the predicted values, whereas, in the case of [74] and [73] where a wireless sensor network was used, they did not totally rely on the values measured by the sensors because they usually tend to develop drifts with time.

Chapter 3

Drift Aware Wireless Sensor Networks

WIRELESS sensor networks comprise low cost sensors with sufficient computing and communication capabilities to support networked sensing applications. The emphasis on lower cost led to sensors that are less accurate and less reliable than their wired sensor counterparts. Sensor measurements usually suffer from both random and systematic errors. Even when the sensors are properly calibrated at the time of their deployment, they develop drift in their readings leading to biased sensor measurements. This poses a real problem from the end application point of view as the data from the network becomes progressively useless. To solve this problem, an automatic way for calibrating these sensors has to be found.

The focus of this chapter is to give a quantitative overview of the drift problem, familiarise the reader of the effects of the drift problem on the inferences made by the network, highlight the importance of finding a solution for it and show how neighbouring sensor can collaborate to detect the drift and partially correct the reported measurements.

The method presented here is rather simple. It assumes that the sensor network is densely deployed, the phenomenon under consideration does not vary with time and that the instantiations of drifts are uncorrelated. The evaluation results of the presented algorithm emphasise that detecting and correcting drift/bias extend the operational life time of the network.

3.1 A Simple drift detection and correction algorithm

The sensor network under consideration consists of N sensors deployed randomly in a certain area A. The sensors are grouped into clusters according to their spatial distances. Each sensor considers its $n-1$ closest sensors as its neighbours in the cluster, i.e., the cluster size is n. Without loss of generality we have chosen to apply the algorithm on a temperature sensor network even though it is applicable to all other types of sensors that suffer from drift and bias errors. The reading $r_{i,k}$ of each sensor is given by:

$$r_{i,k} = T_k + d_{i,k} + w_{i,k} \quad w_{i,k} \sim N(0, R_{i,k}) \tag{3.1}$$

$$d_{i,k} = \begin{cases} 0 & if \quad k < DT_i \\ d & if \quad k \geq DT_i \end{cases} \quad DV_i \sim U(0, a) \tag{3.2}$$

Where T_k is the groundtruth temperature and is assumed to be constant in the cluster $T_k = 25C^o$, $d_{i,k}$ is the drift of sensor i at time instant k. It is a unidirectional (positive) long-term change in the sensor measurement. $w_{i,k}$ is the measurement random error (noise). It is assumed to be Gaussian noise with mean $\mu_{i,k} = 0$ and variance $R_{i,k} = 0.09$, i.e., the standard deviation is $\sigma = 0.3C^o$. DT_i is the drift starting time. d is the drift amplitude. It follows a uniform random function with maximum possible value of a. The steps of algorithm are explained below:

Each sensor in cluster or the sub network calculates the average value of the neighbouring sensors that report their measurements. We assume it has reliable communication channels with them. The average value calculated by sensor i at time step k, $\bar{r}_{i,k}$ is given by:

$$\bar{r}_{i,k} = \begin{cases} \frac{\sum_{j=1}^{n} I_{ji,k} r_{j,k} - I_{ii,k} r_{i,k}}{\sum_{j=1}^{n} I_{ji,k} - I_{ii,k}} & if \quad n \neq 1 \\ r_{i,k} & if \quad n = 1 \end{cases} \tag{3.3}$$

Where i is the number of the sensor calculating the average, $r_{j,k}$ is the reading of sensor j at time step k, n is the number of sensors in the cluster and $I_{ji,k}$ is the communication channel reliability index between sensors j and i at time step k.

$I_{ii,k}$ is always equal to 1. $I_{ji,k}$ takes two values either 1 or 0 indicating that the wireless channel is reliable (sensors i and j can communicate with each other) or not, respectively. The value of $I_{ji,k}$ is simulated in our work according to equation (3.4).

$$I_{ji,k} = \begin{cases} 0 & if \quad \alpha \sim U(0,1) \geq reliability_{ji} \\ 1 & if \quad \alpha \sim U(0,1) < reliability_{ji} \end{cases} \tag{3.4}$$

Where $reliability_{ji}$ is the reliability of the wireless channel between the two sensors i and j, α is the realisation of a uniform random variable defined between 0 and 1. When the value of α is less than $reliability_{ji}$, the communication channel is considered reliable ($I_{ji,k} = 1$), and node j measurement is counted in the average in equation (3.3). The case when reliability of the communication channels of all neighbours of sensor i is equal to zero is equivalent to the case when $n = 1$. After finding the average of its neighbours, each sensor calculates the difference between its measurement and the neighbours average; if the absolute value of the difference is greater or equal to 3σ, the sensor considers itself to have developed a considerable drift error and takes the drift value to be equal to the difference. At this point a correction bias has to be added to the reading of the drifting sensor to compensate for the drift. Each sensor calculates the correction bias $CB_{i,k}$ according to (3.5) and then adds the correction bias to its reading (3.6).

$$CB_{i,k} = \begin{cases} 0 & if \quad |\overline{r}_{i,k} - r_{i,k}| < 3\sigma \\ \overline{r}_{i,k} - r_{i,k} & if \quad |\overline{r}_{i,k} - r_{i,k}| \geq 3\sigma \end{cases} \tag{3.5}$$

$$r_{i,k} = r_{i,k} + CB_{i,k} \tag{3.6}$$

The continuous bias additions to compensate for the drift in sensors readings will result in moving the total average value NET_k of all sensors up. This means that not only the sensors which develop drift will result in wrong temperature measurements, but also the conclusions and actions taken by the whole network will be affected. Since the inferences and conclusions made by the network may be dependent on fusing data coming from all sensors and in some applications on the

average of their readings (as in [5]), a threshold on the average of all sensors of the network NET_k has to be considered. We set the threshold to 3σ. The total average NET_k is calculated by the network's centre node. If the total average exceeds this threshold, the network is considered to have broken down and is deemed to be useless.

The total average value at the time of deployment NET_0 should be taken as a reference since all the sensors at time $k = 0$ are considered to be drift free and correctly calibrated. As time passes, some sensors will start to develop drift. If no correction operation is conducted, the total average of network will rapidly deviate from the initial average as more drifts may develop. However, if the correction operation is conducted, the deviation will be slower. The network is considered to be working properly as long as its total average does not deviate by 3σ from the initial total average. If the total average passes the 3σ threshold, the network is considered to have broken down as follows:

$$NET_k = \frac{\sum_{i=1}^{N} r_{i,k}}{N} \tag{3.7}$$

$$|NET_k - NET_0| \geq 3\sigma \Longrightarrow Breakdown \tag{3.8}$$

A note on the observability of the system

The system constituting of state equation (3.2) and measurement equation (3.1) is completely observable at the deployment stage of the sensor network, when no sensors are drifting. However, as the sensors start to develop drift there is a chance that the system becomes unobservable or partially observable. However, if we detect and correct drifts as soon as the drifts are developed we continue to prolong the state of no drift. One of the explicit assumptions we have made in this thesis is to assume that the instantiations of drifts are random, i.e., the time at which the drifts are generated are random and it is very likely to be different for different sensors. However, if multiple drifting sensors start to drift simultaneously, the system becomes progressively unobservable. Thus when employing the algorithms intro-

duced in this thesis, the drifts are detected and consequently the readings of the sensors are corrected to become close to the ground truth. This together with that the probability of sensors start drifting simultaneously is low, enhance our ability to extend the period of observability of the system. Hence, extending the useful time of the sensor network and giving us the opportunity for making the most use of the network. Here we present a simple case to provide an insight into observability and partial observability of such systems.

In this chapter we have assumed that the temperature T_k is constant. At the deployment of the sensor network, the sensors are also assumed to be properly calibrated and therefore, the initial drift for all the sensors is zero. For the sake of simplicity, we use the same drift value d for all sensors in the neighbourhood but persist with the realistic assumption that the instantiations of drifts are random. While simplifying the discussion and making a point on the nature of observability, it does not diminish the generality of the concept.

As stated before in equation (3.1), the measurement (reading) of sensor i is given by:

$$r_{i,k} = T_k + d_{i,k} + w_{i,k} \quad w_{i,k} \sim N(0, R_{i,k})$$

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

We also define another variable $x_{i,k}$, the corrected measurement of sensor i at time instant k. $x_{i,k}$ is never sensed but calculated. It is the difference between the sensor reading and the drift and is calculated by $x_{i,k} = r_{i,k} - d_{i,k}$ to result in $x_{i,k} = T_k + w_{i,k}$. Therefore, the average of the corrected (bias free) readings of the nodes in the neighbourhood of sensor i is taken as an estimate for the (unbiased) ground truth temperature $\overline{x}_{i,k} = E\{T_k\} = \hat{T}_k$ as $w_{i,k}$ is a zero mean process.

The average of the readings of the sensors neighbouring to sensor i will be:

$$\begin{aligned} \overline{r}_{i,k} &= \hat{T}_k + \overline{d}_{i,k} \\ &= \hat{T}_k + \frac{n_d}{n-1} d \end{aligned}$$

where n_d is the number of drifting sensors in the neighbourhood of sensor i and n is the number of sensors in the neighbourhood including sensor i. d is the amplitude of drift which we previously assumed to be equal for all sensors whereas the instantiations of these drifts are random. This leads to:

$$\hat{T}_k = \overline{r}_{i,k} - \frac{n_d}{n-1}d \tag{3.9}$$

Since $d_{i,k} = d$, equation (3.1) can be written as:

$$r_{i,k} = \hat{T}_k + d + w_{i,k} \quad w_{i,k} \sim N(0, R_{i,k}) \tag{3.10}$$

Substituting equation (3.9) into equation (3.10) leads to:

$$\begin{aligned} r_{i,k} &= \overline{r}_{i,k} - \frac{n_d}{n-1}d + d + w_{i,k} \quad w_{i,k} \sim N(0, R_{i,k}) \\ &= \overline{r}_{i,k} - (1 - \frac{n_d}{n-1})d + w_{i,k} \quad w_{i,k} \sim N(0, R_{i,k}) \end{aligned} \tag{3.11}$$

We also define $y_{i,k}$, the drift measurement, as the difference between the average corrected reading $\overline{x}_{i,k}$ and the reading $r_{i,k}$ as follows:

$$y_{i,k} = r_{i,k} - \overline{x}_{i,k} \tag{3.12}$$

At early stages of the deployment, when no sensor has started to develop drift ($n_d = 0$), it can be found from equation (3.9) that $\hat{T}_k = \overline{r}_{i,k}$. Also given that, as mentioned earlier, $\overline{x}_{i,k} = \hat{T}_k$, then substituting equation (3.11) into equation (3.12) results in:

$$y_{i,k} = d + \eta_{i,k} \quad \eta_{i,k} \sim N(0, \delta_{i,k}) \tag{3.13}$$

Where $\eta_{i,k}$, as will be shown in chapter 4, is a Gaussian noise with zero mean and variance $\delta_{i,k}$. It can be noticed from equation (3.13) that the drift of sensor i is totally observable and measurable. However, when the sensors in the neighbourhood start drifting and taking $\overline{r}_{i,k} \approx \hat{T}_k$ as $r_{i,k}$ is the only really measured quantity by the

sensors, equation (3.13) becomes:

$$y_{i,k} = (1 - \frac{n_d}{n-1})d + \eta_{i,k} \quad \eta_{i,k} \sim N(0, \delta_{i,k}) \tag{3.14}$$

Which means that if all the neighbouring sensors are drifting ($n_d = n - 1$), $y_{i,k}$ will be equal to the noise component leading to the conclusion that the drift is completely unobservable. Whereas if ($n_d < n - 1$) the measured drift will have a value that is a fraction of the true drift value d and therefore we consider the drift as partially observable.

In addition to the above we show here that the probability that many sensors start drifting at the same point of time is low. Lets define $P_d(i)$ as the probability that sensor i starts developing drift in the time slot ($k-1$ *to* k). Assuming that the instantiations of drifts in the sensors are random, and independent, then the probability that sensors i and j start developing drift together in the time slot ($k-1$ *to* k) will be given as the product of the two probabilities $p_d(i,j) = p_d(i).p_d(j)$. Similarly, the probability that all sensors start developing drift at the time slot ($k-1$ *to* k) will be:

$$p_d(all) = \prod_{i=1}^{n} p_d(i) \tag{3.15}$$

Since $P_d(i) < 1$ then $P_d(all) << 1$ which is one of the main conditions for our algorithms in this chapter and the following chapters to be able to accurately detect the drift in the sensor's reading and correct it. It is shown in the evaluation section of chapter 3 how having different instantiations of drifts in sensors enables our algorithm to extend the functional life of the network.

The above discussion have showed that at the deployment stage of the sensor network, when no sensors are drifting, the system is completely observable and the ground truth temperature can be found from the readings of the sensors. However, as the sensors start to develop drift the system becomes partially observable. When our algorithm is implemented it detects these drifts and correct the readings of the sensors to the ground truth value leaving n_d very small for longer period of time. This together with that the probability of many sensors start drifting simul-

taneously is low, enhance our ability to extend the period of observability of the system. Thus, extending the useful time of the sensor network and giving us the opportunity for making the most use of the network.

3.2 Evaluation

In first part of our simulation we consider a sensor network consisting of $N = 15$ sensor distributed randomly in an area A. The temperature of the surroundings is $T_k = 25C^o$ over the whole area. The cluster size is $n = 10$, the communication channels reliability is 1, the standard deviation of the sensors noise is $\sigma = 0.3C^o$ and the number of drifting sensors is $n_d = 5$. We compare two situations: the first is when the network corrects the drifting sensors and we call it Drift Aware Sensor Network ($DASN$). The other is the situation when the network does not correct the drifting sensors and we call it Non Drift Aware Sensor Network ($NDASN$). Referring to equation (3.8) it can be easily shown that an average drift of $(3N\sigma/n_d)$ per sensor or more, for n_d number of sensors, will certainly result in raising the average of sensors readings by 3σ in the case of NDASN causing breakdown. It is worth noting here that all the evaluations given in this chapter are base on 100 Montecarlo simulations. Figure 3.1 shows the comparison between the two types of networks under two drift timing scenarios when applying a drift of $(3N\sigma/n_d) = 2.7C^o$ per sensor for $nd = 5$.

From figure 3.1.a and 3.1.c it is obvious that in the case of the NDASN, the network will breakdown if the drift value for each of the five sensors is equal to $2.7C^o$ irrespective of drift times. On the contrary, in the case of DASN see fig 3.1.b and 3.1.d, a breakdown will only occur if the n_d drifts occur in the same time slot $k = DT = 50$ which is practically improbable. Hence the DASN can be considered to prolong the life of wireless sensor networks.

In the second part we consider a sensor network with 100 sensors ($N = 100$) deployed randomly in an area A. The reliability of the communication channel is modelled as a uniform random variable. Drift times are random. The relationship

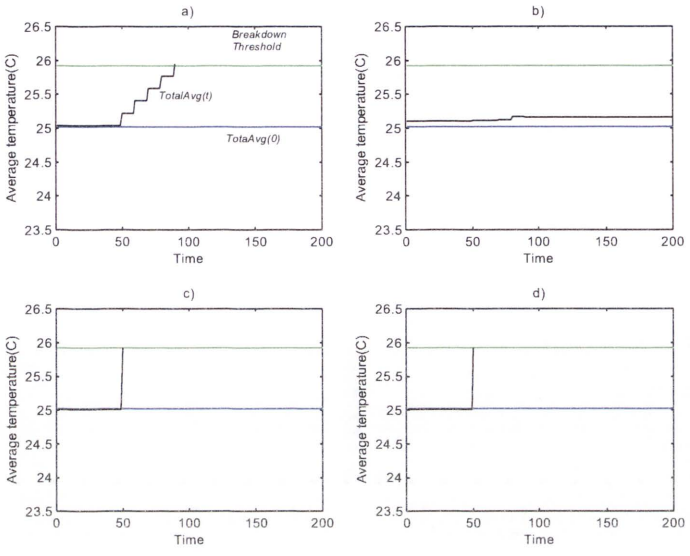


Figure 3.1: A comparison between NDASN and DASN under two time scenarios: a) NDASN,$DT = 50, 60, 70, 80, 90$ b) DASN, $DT = 50, 60, 70, 80, 90$, C) NDASN, $DT = 50$ d) DASN , $DT = 50$

between the number of sensors in each cluster and the probability of breakdown of the network is shown in figure 3.2 for different drift scenarios ($a = 1.5, 3, 4.5, 6$). The reliability of communication channels in this case is fixed at 1. Looking at figure 3.2, it can be clearly seen that the probability of Network breakdown decreases as the cluster size n increases. It is obvious that for each drift scenario, there is a cluster size after which the network does not breakdown unless sensors develop more drifts, some other faults occur or some of the sensors' batteries die.

Figure 3.3 shows the relationship between the reliability of communication channels between the sensors and the probability of network breakdown for different drift scenarios ($a = 1.5, 3, 4.5, 6$) when the cluster size is fixed at $n = 10$. The probability of network breakdown decreases as the reliability of the communication channels increases. It can be seen that for each drift scenario and for a certain cluster size, there is a communication channel reliability after which the network will not breakdown unless sensors develop more drifts, some other faults occur or some of the sensors' batteries die.

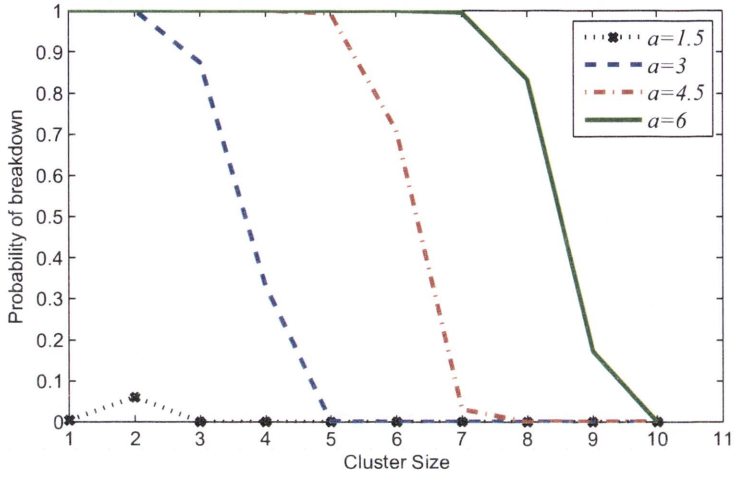


Figure 3.2: Probability of network breakdown VS. Cluster size (n) for different drift scenarios and fixed communication channel reliabilities =1

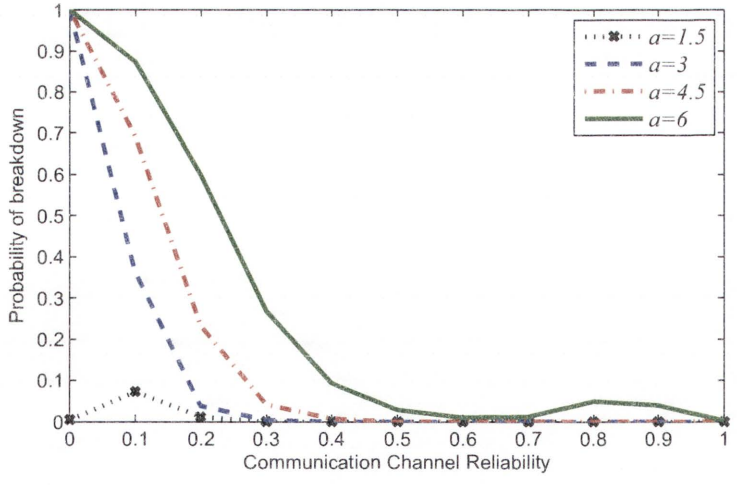


Figure 3.3: Probability of network breakdown VS. Communication channel reliability for different drift scenarios and fixed cluster size $n = 10$

Looking at both figure 3.2 and figure 3.3 we notice an analogy between the effect of cluster size and communication channel reliability on the probability of networks breakdown. This can be justified by noticing that reducing the communication channel reliability is actually reducing the number of sensors a sensor can communicate with to use their measurement in the averaging process. In other words, it means reducing the size of the cluster.

Figure 3.4 shows how the network breakdown time is affected by the number of sensors in the cluster for a communication channel reliability of 1. It can be seen in figures 3.4.b, 3.4.c that the breakdown time increases with n, and that for $n = 10$ in figure 3.4.d no breakdown happens causing the network to continue operating properly unless it develops more drifts. Similarly, no breakdown is noticed in figure 3.4.a since the drift in this case is zero.

Figure 3.5 shows how the network breakdown time is affected by the communication channel reliability for a cluster size of 10. It can be seen in figures 3.5.b, 3.5.c that the breakdown time increases with reliability, and that for $reliability = 1$ in figure 3.5.d no breakdown happens causing the network to continue operating properly unless it develops more drifts. Similarly, no breakdown is noticed in figure 3.5.a since the drift in this case is zero. Comparing figure 3.4.b and figure 3.5.b emphasises our reasoning that for the same drift scenario, the cases when $n = 1$, and the case when $reliability = 0$, are exactly equivalent. In either of these cases the network is like a NDASN.

Figure 3.6 shows the relationship between the cluster size and the probability of network breakdown at a certain drift level $a = 4.5$ and for different number of sensors N in the network. It can be seen that N has a negligible effect on the probability of network breakdown, when taking the threshold for breakdown to be a probability of 0.01. So for a cluster size $n = 8$ or higher, the network can be considered to have approximately zero probability of breakdown due to drift, irrespective of the value of N.

Finally, figure 3.7 shows the relationship between the communication channel reliability and the probability of network breakdown at a certain drift level and for

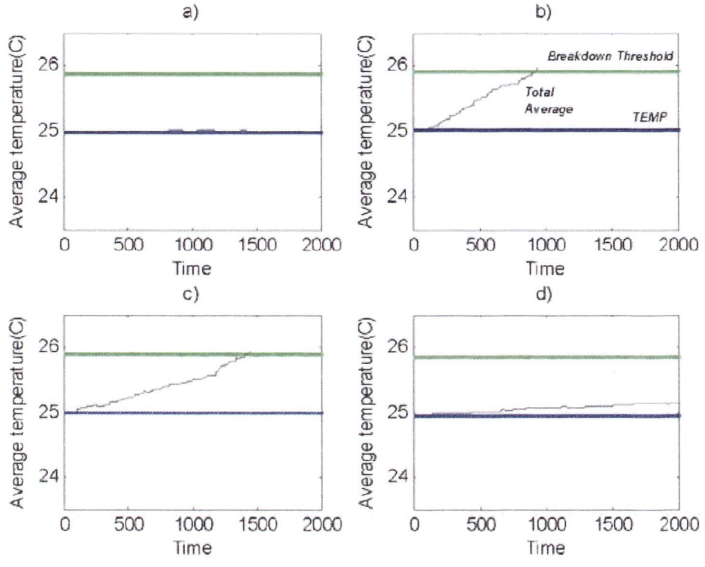


Figure 3.4: Breakdown time for different cluster sizes and different drift scenarios for $N = 100$ and *reliability* $= 1$: a) no drift, b) $a = 6$, $n = 1$,c) $a = 6$, $n = 5$,d) $a = 6$, $n = 10$

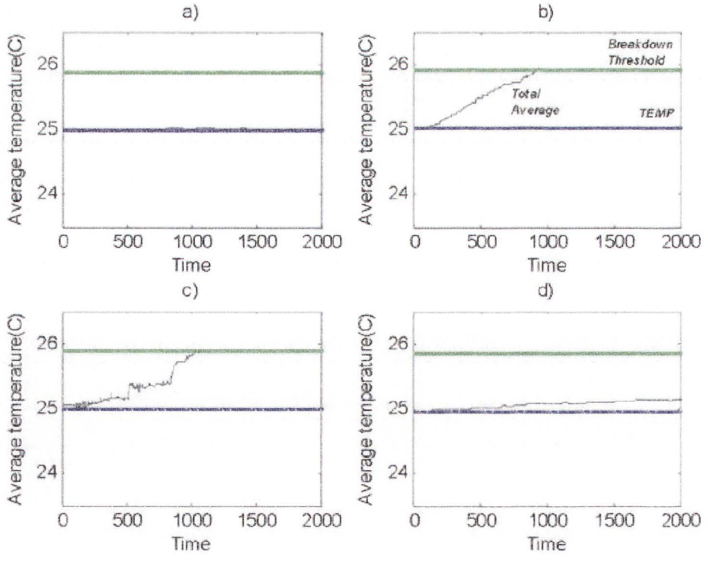


Figure 3.5: Breakdown time for different communication channel reliabilities and different drift scenarios for $N = 100$, $n = 10$: a) no drift, b) $a = 6$, *reliability* $= 0$, c) $a = 6$, *reliability* $= 0.3$, d) $a = 6$, *reliability* $= 1$

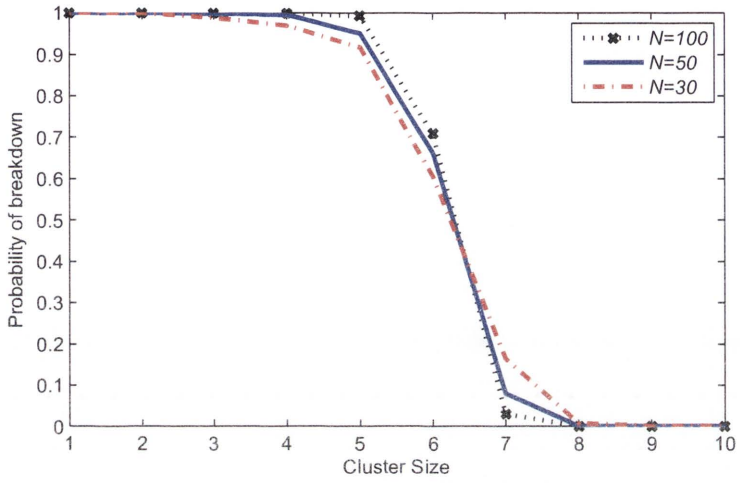


Figure 3.6: Probability of network breakdown VS. Cluster size for $N = (100, 50, 30)$, *reliability* $= 1$ and $a = 4.5$

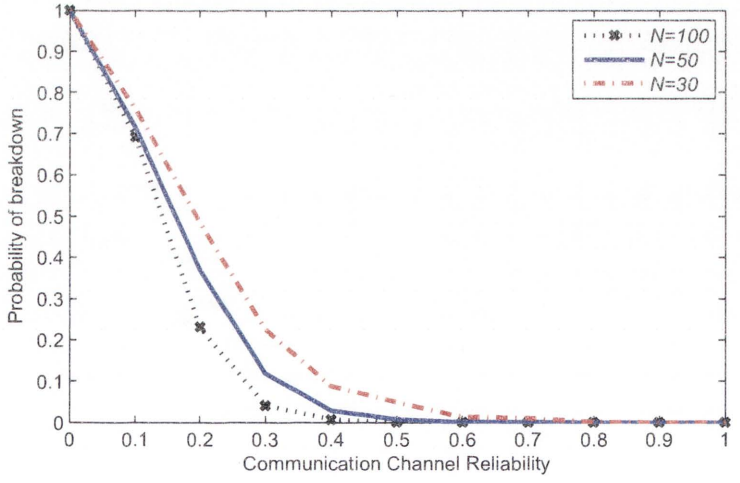


Figure 3.7: Probability of network breakdown VS. Communication channel reliability for $N = (100, 50, 30)$, $n = 10$ and $a = 4.5$

different number of sensors N in the network. It can be seen that N has a negligible effect on the probability of network breakdown provided that the threshold for breakdown is taken to be a probability of 0.01. So for a communication channel reliability $reliability = 0.7$ or higher, the network can be considered to have approximately zero probability of breakdown due to drift, irrespective of the value of N.

As a summary, our simulation results showed that:

- Breakdown depends on the value of the drift in each sensor, the time spacing between drift occurrences, the communication channel reliability and the number of sensors in each cluster.
- The case when $n = 1$ or $reliability = 0$ is equivalent to the case when the network is non drift aware since no drift correction is performed in both cases.
- The number of sensors in the network has a negligible effect on the probability of network breakdown.
- Therefore we can generalise that for a certain drift level, there is a certain cluster size and communication channel reliability that will prevent the network breakdown, due to drift, no matter what the size of the network is.

3.3 Conclusion

We have addressed the problem of systematic errors (drifts and bias) in sensor measurements and explained their bad effects on the inferences made by the sensor networks. We argued that the nature of deployment of the sensors in the sensor network imposes finding an algorithm for auto calibrating these sensors. Therefore, we introduced a simple algorithm for auto calibration of these sensors under the simple scenario of dense deployment of the sensors to observe the same data. Each sensor uses the average value of it's neighbours as a basis to self-assess its readings and then detect and correct its drift. Another important assumption that supports the solution is that the instantiation of drift in one sensor is unlikely to be the same as the other. The simulation results showed that using the algorithm

results in increasing the effective life span of the network. We have also showed that the number of the sensors in the neighbourhood, the communication channel reliability, the instantiation and amplitude of drifts are important factors affecting the performance of the solution.

The solution introduced in this chapter is rather simple. The scenario of the constant temperature is simplistic. However, the objective is to demonstrate and show the effect of the drift problem on sensor networks and to demonstrate how auto calibrating the sensors will extend the lifetime of the network. In the next chapter, we propose a new algorithm to deal with more realistic situations where the temperature or any other parameter of interest is variable time in the area of deployment of the sensor network.

Chapter 4

Correcting Measurement Errors under Smooth Drift Scenario

THIS chapter presents a formal statistical framework for correcting both systematic and random sensor measurement errors in a WSN. The solution presented in this chapter is inspired by the solution of the Registration Problem in target tracking, where a target may be observed by a group of sensors (radars), from different angles. The observations of these sensors are reported, and fused, to estimate not only the true target location, but also the biases of the sensors [49, 50].

In addressing the sensor measurement errors problem, we follow a Bayesian reasoning that leads us to a Kalman Filter (KF) based solution for estimating the drift. The estimate is used for correcting the measurements and as a feedback to the next estimation step. The solution is designed for smooth drifts i.e. drifts that are slowly varying with time. The neighbouring sensors in the network are assumed to have correlated measurements and the instantiation of drift in a sensor is assumed to be uncorrelated with other sensors . The sensors in the neighbourhood (cluster) are assumed to be densely deployed with similar measurements. That is, the measurements in the cluster are space invariant. Hence, the average of the sensors' corrected readings in the cluster, is taken as a basis for each sensor to self-assess. The presented methodology is a robust framework for auto calibration of sensors in a WSN.

In the following section, we introduce a Bayesian formulation for the sensor measurement errors detection and correction problem. The evaluation of the new solution is presented in section 4.3 followed by conclusion.

4.1 Smooth drifts estimation and measurements correction algorithm

In this section we introduce a Bayesian approach to solve the sensor measurement errors problem in WSN, assuming that the drifts are smooth (see figure 1.2) and that sensor nodes are densely deployed. Under the dense deployment assumption, all the sensors nodes in a cluster are assumed to measure the same value. Therefore, the average of corrected sensor measurements $\overline{x}_k$ is considered as a good estimate for the expectation of the ground truth value $E\{T_k\}$ in the cluster. It is also considered as a good basis for the sensors to self-assess their measurements.

Let us assume that at time instant k, a measurement or a reading $r_{i,k}$ is made by node i. Rather than sending that value to its neighbours, the node is aware of its drift, and has an estimate for it at this time instant. It is a projected value from an estimate of the drift made at the previous time instant. Using this estimate of the drift, the node i computes its corrected measurement $x_{i,k}$ and sends it to its neighbouring nodes. This applies to all the nodes in the neighbourhood. Each node then collects all the neighbourhood sensors corrected measurements $\{x_{i,k}\}_{i=1}^{n}$, and computes the average $\overline{x}_k = \sum_{i=1}^{n} x_{i,k}/n$. At this point each sensor computes the drift measurement. We define the drift measurement as the difference between the sensor measurement and the average value computed by that sensor. We denote the drift measurement of node i at time instant k by $y_{i,k}$. The drift measurement is used by a KF to estimate the drift. The problem is formulated mathematically as follows:

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

$$d_{i,k} = d_{i,k-1} + v_{i,k} \quad v_{i,k} \sim N(0, Q_{i,k}) \tag{4.1}$$

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

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

by:

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

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

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

Since the sensors are densely deployed and the instantiations of drifts in the sensors are random, we use the average of corrected sensors' measurements close to node i as an estimate for the expectation of actual (ground truth) value $\overline{x}_k = E\{T_k\}$. We also define $y_{i,k}$ in equation (4.2) as the difference between the measurement $r_{i,k}$ and the average of corrected sensors measurements $\overline{x}_k$ and refer to $y_{i,k}$ as the drift measurement of node i at time instant k.

$$y_{i,k} = r_{i,k} - \overline{x}_k \tag{4.2}$$

At early stages of deployment of the sensor network when very few sensors have started to develop drift, and given that the instantiations of drifts in all the sensors are random, we assume that $E\{T_k\} = T_k$. Substituting $r_{i,k}$ into equation (4.2) results in:

$$\begin{aligned} y_{i,k} &= T_k + d_{i,k} + w_{i,k} - E\{T_k\} - \frac{1}{n}\sum_{j=1}^{n} w_{j,k} \\ &= d_{i,k} + w_{i,k} - \frac{1}{n}\sum_{j=1}^{n} w_{j,k} \\ &= d_{i,k} + \psi_{i,k} \quad \psi_{i,k} \sim N(0, \delta_{i,k}) \end{aligned} \tag{4.3}$$

where $\psi_{i,k} = w_{i,k} - \frac{1}{n}\sum_{j=1}^{n} w_{j,k}$ is the drift measurement noise and is actually a mixture of Gaussians. It is well known in literature [75, 76] that a Gaussian mixture can be approximated by a Gaussian $\psi_{i,k} \sim N(\pi_{i,k}, \delta_{i,k})$ with the mean found by the

weighted sum of the means of the original Gaussians:

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

and the variance found by:

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

Assuming that all sensors are neighbours in the cluster and that they can report to each other, then equation (4.1) and equation (4.3) can be written in vector form for all the sensors of the cluster as follows:

$$D_k = FD_{k-1} + V_k \quad V_k \sim N(0, Q_k) \tag{4.4}$$

$$Y_k = HD_k + \Psi_k \quad \Psi_k \sim N(0, \Delta_k) \tag{4.5}$$

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

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

expanded by Bayes rule as follows:

$$\begin{aligned} p\left(D_k|Y^k\right) &= p\left(D_k|Y_k, Y^{k-1}\right) \\ &= \frac{p\left(Y_k|D_k, Y^{k-1}\right) \cdot p\left(D_k|Y^{k-1}\right)}{p\left(Y_k|Y^{k-1}\right)} \end{aligned} \tag{4.6}$$

Assuming the measurement noise is white Gaussian; i.e. not correlated in time, then the current measurements do not depend on the previous measurements and (4.6) reduces to:

$$p\left(D_k|Y^k\right) = \frac{\overbrace{p\left(Y_k|D_k\right)}^{likelihood} \cdot \overbrace{p\left(D_k|Y^{k-1}\right)}^{predicted\ density}}{\underbrace{p\left(Y_k|Y^{k-1}\right)}_{normalisation}} \tag{4.7}$$

The Likelihood $p(Y_k|D_k)$ for sensor i can be obtained from the measurement equation (4.5) where Ψ_k is a noise vector, assumed to be Gaussian with zero means and covariance Δ_k. Given D_k, the probability of obtaining a drift measurement vector Y_k should be equal to the probability of the noise with mean HD_k:

$$p\left(Y_k|D_k\right) = N(HD_k, \Delta_k) \tag{4.8}$$

The predicted density predicts the current state D_k of the sensors based on the old measurements. We expand it here by Chapman-Kolmogorov identity (an approach used by [77]) as follows:

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

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

$$p\left(D_k|Y^{k-1}\right) = \int p(D_k|D_{k-1}) p(D_{k-1}|Y^{k-1})\ dD_{k-1} \tag{4.9}$$

To evaluate the predicted density $p\,(D_k|Y^{k-1})$ we have to evaluate $p(D_k|D_{k-1})$ and $p(D_{k-1}|Y^{k-1})$ and then substitute them into (4.9). From (4.4) and similar to the likelihood:

$$p\,(D_k|D_{k-1}) = N(FD_{k-1}, Q_k) \tag{4.10}$$

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

$$p\,(D_{k-1}|Y^{k-1}) = N(\hat{D}_{k-1|k-1}, P_{k-1|k-1}) \tag{4.11}$$

Substituting into (4.9) and evaluating the integral as given in [77] we get:

$$\begin{aligned} p\,(D_k|Y^{k-1}) &= \int N(\hat{D}_{k-1|k-1}, P_{k-1|k-1}) N(FD_{k-1}, Q_k)\; dD_{k-1} \\ &= N(\hat{D}_{k|k-1}, P_{k|k-1}) \end{aligned} \tag{4.12}$$

where

$$\hat{D}_{k|k-1} = F\hat{D}_{k-1|k-1} \tag{4.13}$$

$$P_{k|k-1} = FP_{k-1|k-1}\,F^T + Q_k \tag{4.14}$$

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

$$\begin{aligned} p\,(Y_k|Y^{k-1}) &= \int p(Y_k|D_k, Y^{k-1}) p(D_k|Y^{k-1}) dD_k \\ &= \int p(Y_k|D_k) p(D_k|Y^{k-1}) dD_k \end{aligned} \tag{4.15}$$

Substituting the likelihood and the predicted density in the integral of (4.15) results in:

$$\begin{aligned} p\,(Y_k|Y^{k-1}) &= \int N(HD_k, \Delta_k) N(\hat{D}_{k|k-1}, P_{k|k-1}) dD_k \\ &= N(H\hat{D}_{k|k-1}, S_k) \end{aligned} \tag{4.16}$$

where

$$S_k = HP_{k|k-1}H^T + \Delta_k \tag{4.17}$$

Putting all the terms together in (4.7) and evaluating using an identity given in the appendix of [77] results in:

$$\begin{aligned} p\left(D_k|Y_k\right) &= \frac{N(HD_k, \Delta_k)N(\hat{D}_{k|k-1}, P_{k|k-1})}{N(H\hat{D}_{k|k-1}, S_k)} \\ &= N(DD_k, P_{k|k}) \end{aligned} \tag{4.18}$$

where

$$K = P_{k|k-1}H^T(HP_{k|k-1}H^T + \Delta_k)^{-1} \tag{4.19}$$

$$\hat{D}_{k|k} = \hat{D}_{k|k-1} + K(Y_k - H\hat{D}_{k|k-1}) \tag{4.20}$$

$$P_{k|k} = (1 - KH)P_{k|k-1} \tag{4.21}$$

Equations sets (4.13-4.14) and (4.19-4.21) represent a KF framework [78], [79] for n sensor nodes in the cluster. Since F and H are identity matrices, the system above can be solved as an n-dimensional KF (by a central node and requires high computational capability) or as n 1-dimensional KFs solved by each sensor in the cluster. The first solution is centralised, whereas the latter is decentralised and requires no special processing power by the sensor nodes. We adopt the decentralised solution in this work. F and H are taken to be equal to one. This leads to the probabilistic solution for drift $d_{i,k} \sim N(\hat{d}_{i,k|k}, P_{i,k|k})$ with mean and variance:

$$\hat{d}_{i,k|k} = \hat{d}_{i,k-1|k-1} + K(y_{i,k} - \tilde{d}_{i,k}) \tag{4.22}$$

$$P_{i,k|k} = (P_{i,k-1|k-1} + Q_{i,k})(1 - K) \tag{4.23}$$

$$K = \frac{P_{i,k-1|k-1} + Q_{i,k}}{P_{i,k-1|k-1} + Q_{i,k} + \delta_{i,k}} \tag{4.24}$$

The above equations are obtained by substituting the prediction equations of the KF (4.13-4.14) into the update equations (4.19-4.21). $\tilde{d}_{i,k}$ is the predicted drift at the beginning of stage k, before the correction. In this case, $\tilde{d}_{i,k} = F\hat{d}_{i,k-1|k-1} =$

$\hat{d}_{i,k-1|k-1}$, a straightforward prediction given by the KF solution. The variances $Q_{i,k}$, $\delta_{i,k}$, $P_{i,k-1|k-1}$ and $P_{i,k|k}$ are numbers, and therefore the solution is easy to compute. Once $\hat{d}_{i,k|k}$ is obtained, it is used as the predicted drift $\tilde{d}_{i,k+1}$ for the next stage. This allows for the correction of reading $r_{i,k+1}$. The solution is implemented in a decentralised iterative procedure i.e. it is run in each node and at each time step to estimate its drift $d_{i,k}$. Using this estimation; $r_{i,k+1}$ is corrected to $x_{i,k+1}$ and the drift $d_{i,k+1}$ is estimated again and so on. A block diagram describing the algorithm is shown in figure(4.1).

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

Decentralised error correction algorithm for smooth drifts

For each node i

- At step k, the predicted drift $\widetilde{d}_{i,k} = \hat{d}_{i,k-1|k-1}$ and the previous time step process variance $P_{i,k-1|k-1}$ are available.
- Each node i obtains its reading $r_{i,k}$
- The corrected reading is calculated, $x_{i,k} = r_{i,k} - \widetilde{d}_{i,k}$ and then transmitted to the neighbouring nodes.
- Each node computes the average $\overline{x}_k$.
- The Drift measurement $y_{i,k} = r_{i,k} - \overline{x}_k$ is computed.
- Substituting into (4.22-4.24) results in the current time step estimates $\hat{d}_{i,k|k}$ and $P_{i,k|k}$.
- The projected drift $\widetilde{d}_{i,k+1} = \hat{d}_{i,k|k}$ is obtained and the algorithm iterates.

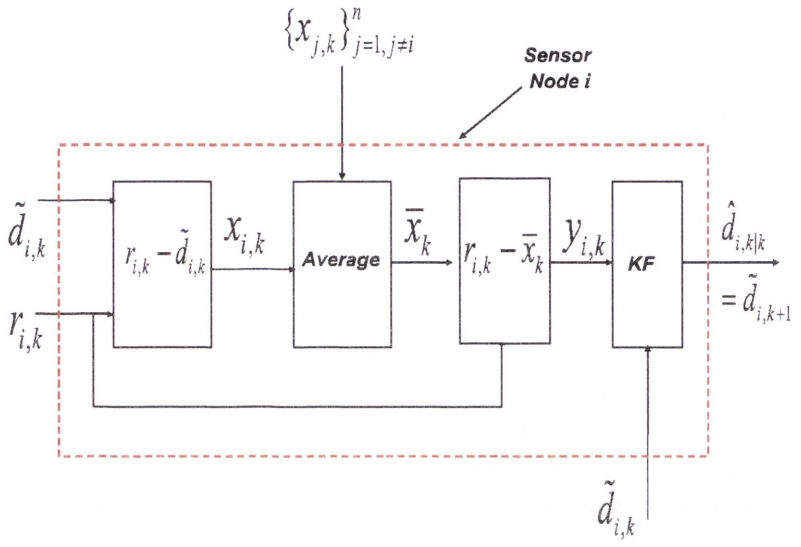


Figure 4.1: A block diagram for the smooth drift estimation and measurement correction algorithm

4.2 Complexity analysis

The computational complexity and memory complexity of our smooth drift correction algorithm is mainly contributed by the KF computations. The KF estimates the drift at every time step using equations (4.22-4.24). This involves real number computations rather than matrix computations. Hence the computational complexity is only contributed by the number of multiplications and divisions of real numbers. Memory complexity is mainly contributed by storing the values of the estimated drift of the current and previous instances, current reading, and the calculated average. The communication overhead in the network is contributed by the reporting of x among the neighbours.

4.3 Evaluation

Our aim is to evaluate the ability of our proposed framework to correct the drift experienced in a sensor node using the information gathered from the nearest neigh-

bouring nodes. We simulate a small sub-network of 10 densely deployed sensor nodes measuring the temperature in a certain area. We assume that 2 sensors are developing smooth drifts of the forms shown in figure 1.2. The measurement noise is taken to be Gaussian with zero mean and variance 0.01. The measurement variance $\delta_{i,k}$ for each node is chosen from [0.005-0.01] and the state variance $Q_{i,k}$ is taken to be 0.001. The results of the KF drift tracking algorithm are shown in figures 4.2 and 4.3. It is clear from figure 4.2 that our algorithm is capable of estimating the drifts accurately. We see that the estimated drifts of sensor 1 and 2 are very close to the actual drifts. It is also clear that the algorithm corrects the erroneous measurements to become very close to the actual temperature as seen in figure 4.3.

Figures 4.4 and 4.5 show the results of the KF drift tracking algorithm when seven sensors out of ten have developed drift. It is clear that the system can still detect and correct the drift even when 70% of the sensors are drifting. However, when comparing figure 4.4 of the seven drifting sensors with figure 4.2 of the two drifting sensors, we see that the error increases as the percentage of drifting sensors increases.

Looking at figures 4.3 and 4.5, It is clear that in both cases (when 2 sensors and when 7 sensors are drifting), the KF drift tracking algorithm extends the effective operational life time for node 1. If we ignore the transient overshoot period (29-35) when the drift starts to develop, and assume that for our application the maximum tolerable temperature error in node's 1 reading is 1 C^o, then the life of node 1 is extended from 27 time units when there is no drift correction (Reading of node 1 curve), to at least 100 time units for both cases (Corrected reading curve). This applies to all of the network's sensors that develop drift. Hence, the life of the network will be extended by applying the drift tracking and correction algorithms. The overshoot in the transient period is due to the fast change in the drift when it starts developing. A slower drift will cause a smaller overshoot. This can be solved by adding another component to the state vector, namely, the speed of the drift or by using the Interacting multiple model algorithm as will be seen in the next chapter.

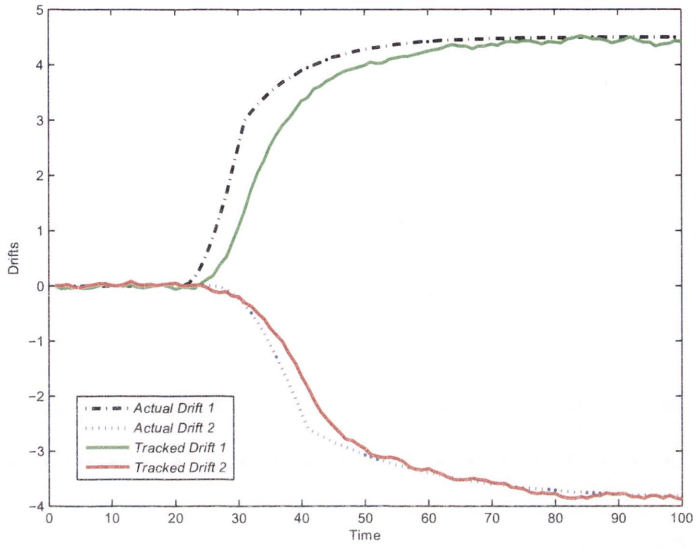


Figure 4.2: Actual and estimated drifts in nodes 1 and 2 for when 2 sensors are drifting

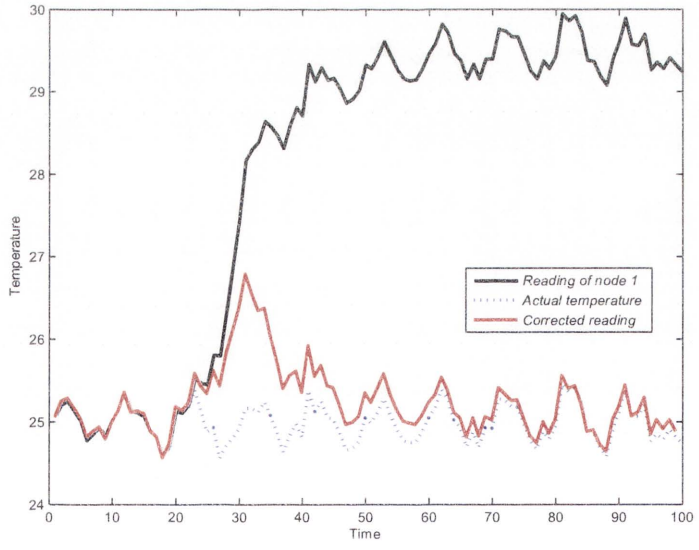


Figure 4.3: The reading of node 1, the corrected reading and the actual temperature when 2 sensors are drifting

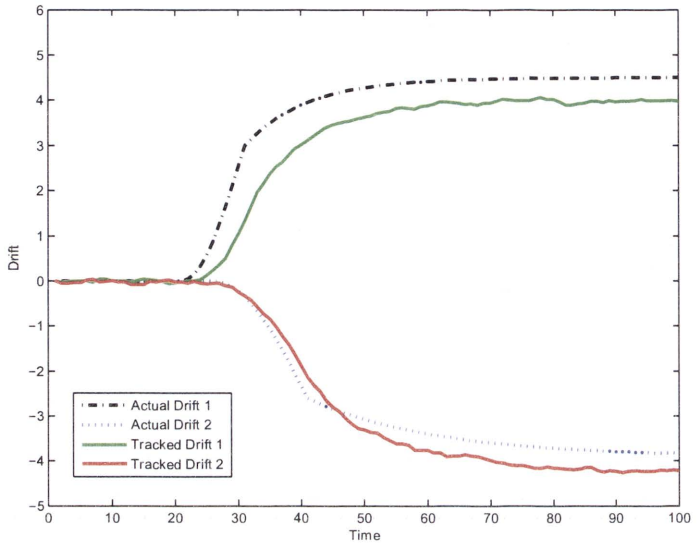


Figure 4.4: Actual and estimated drifts in nodes 1 and 2 when 7 sensors are drifting

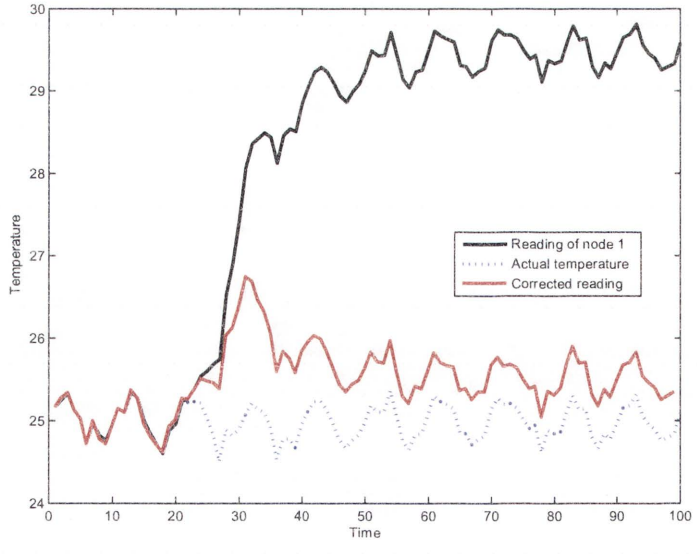


Figure 4.5: The reading of node 1, the corrected reading and the actual temperature when 7 sensors are drifting

The average error between the reading of node 1 and the actual temperature after the transient period is 4.4 C^o. The average error between the estimated temperature and the actual temperature is 0.2 C^o when two sensors are drifting and 0.8 C^o when seven sensors are drifting. This means that the average error has been reduced by $\frac{4.4-0.2}{4.4}\% = 95.5\%$ for the case of two drifting sensors and by $\frac{4.4-0.8}{4.4}\% = 81.8\%$ for the case of seven drifting sensors. Therefore; it can be said that the intuitive idea of correcting sensor errors immediately and in a distributed procedure, prolongs the effective life of the sensor network.

In addition to the drift, we also address the bias problem. As we mentioned earlier, the bias is the initial reading error or in other words, the drift at time zero. We assume that the sensor nodes are factory calibrated before deployment. Therefore, it is unlikely that a high percentage of the sensors will suffer from bias. In a sensible situation, if most of the sensors are without bias at time 0, then such a bias can be captured by the proposed solution as a constant drift of a certain amplitude. Figures 4.6 shows the KF algorithm results when both sensors 1 and 2 suffer from both bias and drift. The results demonstrate that the algorithm efficiently captures and corrects the bias. It is important to note here that if many sensors suffer from the bias, then our solution will not be accurate, as we get little help from the neighbouring sensors to correct the bias. Furthermore, it is unrealistic to have many sensors with initial bias as the sensors are supposed to be factory calibrated before deployment.

We conducted several simulations and observed that the method worked as long as not all sensor start drifting at the same instant of time. Generally speaking, we noticed that the performance of the algorithm is dependent on the number of drifting sensors, the amplitude and the sign of drifts or biases and the starting times of the drifts. If all the sensors suffer from considerable biases at time zero the method will not work accurately. The dependence on the amplitude and the sign of the drift will be taken care of by using machine learning techniques to find the relation among the sensors in the cluster instead of using the average. Using machine learning for this purpose will be the topic of future chapters.

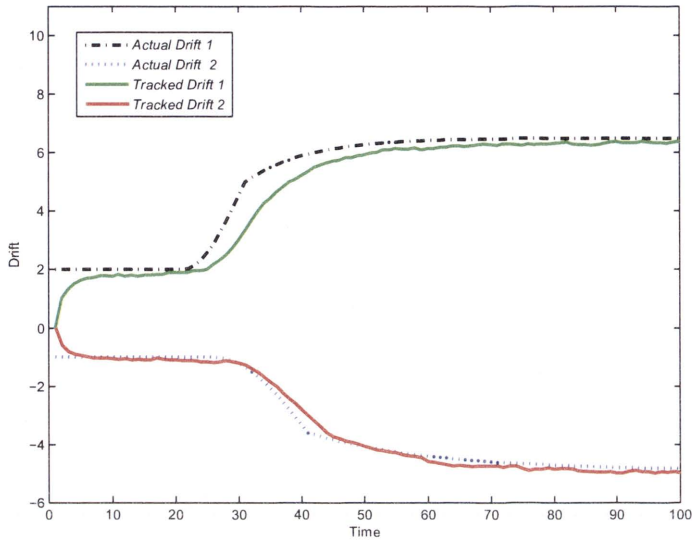


Figure 4.6: Actual and estimated biases/drifts in nodes 1 and 2

4.4 Conclusion

In this chapter we have proposed a formal Bayesian framework for estimating sensor bias and smooth drift and correcting measurements errors in a WSN. Our proposed solution exploits the notion that measurements errors due to faulty equipment are likely to be uncorrelated. The sensors in the neighbourhood are assumed to be densely deployed. Hence, the average of their corrected readings is taken as a basis for each sensor to self assess. The solution is computationally simple using an averaged sensing value and a single state KF iterative procedure, allowing its implementation in a WSN. The proposed solution is completely decentralised - each node only needs to obtain information from its neighbouring sensors and then apply the KF iterative procedure.

We consider at first a smooth drift model. The simulations conducted showed that, with our distributed recursive algorithm, sensors readings errors can be reduced by more than 81 percent when 70 percent of the sensors in the sub-network are drifting and more than 95 percent when fewer sensors are developing drift.

In the coming chapters, we introduce probabilistic models that capture drifts and errors of a different nature. Drifts that have surges and sudden escalations are considered in (Chapter 5). We will also address the problem when the phenomenon under consideration (here temperature) in the sub-network varies with distance and time (Chapters 6, 7).

Chapter 5

Correcting Measurement Errors under Unsmooth Drift Scenario

THIS chapter introduces another solution for correcting both systematic and random errors inherent in sensors measurements in a WSN. The algorithm presented here is capable of detecting and correcting both smooth and unsmooth drifts i.e. drifts with abrupt changes or jumps. The jumps are either inherent in the drift itself or caused by under sampling the data received by each node, or in other words, reducing the frequency of reporting the corrected readings among the neighbours in order to reduce the communication overhead and save the battery life. The jumps differ in their amplitudes and instantiation. Both the amplitudes and the instantiations of the drifts are considered to be random.

Such drift behaviour is not followed well by the KF algorithm given in the previous chapter and [70]. The standard KF with single drift model is limited in performance since it does not efficiently respond to rapid changes in the dynamics as the drift changes abruptly at some points. Alternatively, adaptive estimation techniques such as the adaptive Multiple Model (MM) approaches have the potential to correct them. The adaptive MM approaches have been successful in dealing with similar problems in tracking manoeuvring targets [75]. They are based on the fact that the behaviour of a state cannot be characterised at all times by a single model, and that finite number of models are needed to adequately describe its behaviour in different regimes [80].

The optimal multiple model solution for the problem is computationally infeasible [80]. It requires an exponential computational complexity $O(S^T)$ for a length T

and M models [81]. As a result, many suboptimal approximations such as the Generalised Pseudo Bayesian algorithms (GPBs) and the Interacting Multiple Model algorithm (IMM) have emerged. IMM was first introduced by [82] as a filtering algorithm for linear discrete time system with markovian coefficients. It consists of a bank of M number of filters (KF here) working in parallel to result in M number of estimate pairs (states and covariances) each corresponding to a state model. The models estimates are then fused together taking the models probabilities into account to result in the resultant state estimate. The attraction of IMM comes from the fact that its computational requirements are nearly linear in the size of the problem (number of models) while its performance is almost the same as that of an algorithm with quadratic complexity [80]. Moreover, It can be set up using KF or EKF as its building blocks to account for nonliearities [80]. Therefore, it is used in the algorithm that is presented in this chapter. We show the derivation of the IMM algorithm in the next section following the steps of [75].

5.1 Derivation of the IMM Algorithm

The state (here the drift with abrupt changes) is modelled the as a jump markovian system. It is a system whose parameters evolve according to the realisation of a finite state markov chain [81]. The jump markovian systems have been used to model systems in which the system behaviour switches between number of models. A wide variety of control systems lie in this category. Mathematically, the state dynamics of jump markovian systems are assumed to belong to the set of models defined by:

$$d_k = f_{m_k}(d_{k-1}, u_k) + v_{m_k} \quad m_k \in \{1, 2, \ldots, M\} \tag{5.1}$$

where v_{m_k} is the process noise and assumed to be Gaussian with zero mean and covariance Q_{m_k}. m_k is assumed to be a random variable satisfying a homogeneous discrete-time Markov chain with state space $\{1, 2, \ldots, M\}$ and a transition matrix Γ where

$$\Gamma_{\alpha\theta} = p(m_k = \theta | m_{k-1} = \alpha) \tag{5.2}$$

with initial condition $p(m_0 = \theta) = \pi_0(\theta)$. The measurements are assumed to be model dependent and related to the true state through:

$$y_k = h_{m_k}(d_k) + \psi_{m_k} \tag{5.3}$$

where ψ_{m_k} is the process noise and assumed to be Gaussian with zero mean and covariance δ_{m_k}. Since the model jumping process is assumed to be a random process with an underlying Markov chain, the true model can never be known. However, it can be estimated by treating it as a joint random variable with the state d_k, forming a hybrid state (d_k, m_k).

The joint probability function $p(d_k, m_k|y^k)$ can be decomposed into M components since m_k is a discrete random variable as follows:

$$p(d_k, m_k = \theta|y^k)\ \theta = \{1, 2, \ldots, M\}$$

The density of interest in our application is the posterior conditional density $p(d_k|y^k)$. It can be obtained by summing the individual components of the joint density as follows:

$$p(d_k|y^k) = \sum_{\theta=1}^{M} p(d_k, m_k = \theta|y^k) \tag{5.4}$$

Using the conditional probability lemma, the joint density in the right hand side of (5.4) can be broken into two components:

$$p(d_k, m_k = \theta|y^k) = p(d_k|m_k = \theta, y^k)p(m_k = \theta|y^k)$$

Letting $\mu^{\theta}_{k|k} = p(m_k = \theta|y^k)$ then the posterior density equation becomes:

$$p(d_k|y^k) = \sum_{\theta=1}^{M} p(d_k|m_k = \theta, y^k)\mu^{\theta}_{k|k} \tag{5.5}$$

The optimal Bayesian recursion for the first component can be derived by first

expanding the measurements y^k into $\{y^{k-1}, y_k\}$ and then using Bayes' rule:

$$\begin{aligned} p(d_k|m_k = \theta, y^k) &= p(d_k|m_k = \theta, y_k, y^{k-1}) \\ &= \frac{p(y_k|d_k, m_k = \theta, y^{k-1})}{p(y_k|m_k = \theta, y^{k-1})} p(d_k|m_k = \theta, y^{k-1}) \end{aligned} \tag{5.6}$$

where $p(d_k|m_k = \theta, y^{k-1})$ is the predicted density, $p(y_k|d_k, m_k = \theta, y^{k-1})$ is the likelihood function and $p(y_k|m_k = \theta, y^{k-1})$ is the normalisation factor. The posterior model probability $\mu^{\theta}_{k|k} = p(m_k = \theta|y^k)$ can also be recursively calculated as follows:

$$\mu^{\theta}_{k|k} = p(m_k = \theta|y_k, y^{k-1}) = \frac{p(y_k|m_k = \theta, y^{k-1})}{p(y_k|y^{k-1})} p(m_k = \theta|y^{k-1}) \tag{5.7}$$

Letting $\mu^{\theta}_{k|k-1} = p(m_k = \theta|y^{k-1})$ then the probability recursion can be rewritten as:

$$\mu^{\theta}_{k|k} = \frac{p(y_k|m_k = \theta, y^{k-1})}{p(y_k|y^{k-1})} \mu^{\theta}_{k|k-1}$$

where $\mu^{\theta}_{k|k-1}$ is the predicted model probability, $p(y_k|m_k = \theta, y^{k-1})$ is the likelihood function and $p(y_k|y^{k-1})$ is the normalisation factor.

The IMM derivation conditions on m_{k-1}. Therefore, using the laws of probability, the term $p(d_k|m_k = \theta, y^{k-1})$ is expanded as follows:

$$\begin{aligned} p(d_k|m_k = \theta, y^{k-1}) &= \sum_{\alpha=1}^{M} p(d_k, m_{k-1} = \alpha|m_k = \theta, y^{k-1}) \\ &= \sum_{\alpha=1}^{M} p(d_k|m_{k-1} = \alpha, m_k = \theta, y^{k-1}) p(m_{k-1} = \alpha|m_k = \theta, y^{k-1}) \\ &= \sum_{\alpha=1}^{M} p(d_k|m_{k-1} = \alpha, m_k = \theta, y^{k-1}) \mu^{\alpha|\theta}_{k-1|k} \end{aligned} \tag{5.8}$$

where $\mu^{\alpha|\theta}_{k-1|k} = p(m_{k-1} = \alpha|m_k = \theta, y^{k-1})$ are called the mixing probabilities [83]. The mixing probabilities relate to the transition probabilities as follows:

$$\begin{aligned} \mu^{\alpha|\theta}_{k-1|k} &= p(m_{k-1} = \alpha|m_k = \theta, y^{k-1}) \\ &= \frac{p(m_k = \theta|m_{k-1} = \alpha, y^{k-1}) p(m_{k-1} = \alpha|y^{k-1})}{p(m_k = \theta|y^{k-1})} \end{aligned}$$

$$
\begin{aligned}
&= \frac{p(m_k = \theta | m_{k-1} = \alpha)\, p(m_{k-1} = \alpha | y^{k-1})}{\sum_{\beta=1}^{M} p(m_k = \theta | m_{k-1} = \beta, y^{k-1})\, p(m_{k-1} = \beta | y^{k-1})} \\
&= \frac{p(m_k = \theta | m_{k-1} = \alpha)\, p(m_{k-1} = \alpha | y^{k-1})}{\sum_{\beta=1}^{M} p(m_k = \theta | m_{k-1} = \beta)\, p(m_{k-1} = \beta | y^{k-1})} \\
&= \frac{\Gamma_{\alpha\theta}\, \mu_{k-1|k-1}(\alpha)}{\sum_{\beta=1}^{M} \Gamma_{\beta\theta}\, \mu_{k-1|k-1}(\beta)}
\end{aligned}
\tag{5.9}
$$

Introducing the prior state d_{k-1} in (5.8), the prediction density can be expanded into:

$$
p(d_k | m_k = \theta, y^{k-1}) = \sum_{\alpha=1}^{M} \int p(d_k, d_{k-1} | m_{k-1} = \alpha, m_k = \theta, y^{k-1})\, \mu_{k-1|k}^{\alpha|\theta}\, \mathrm{d}d_{k-1}
$$

Invoking the conditional density lemma on the joint density inside the integrand, and using the fact that the state d_{k-1}, at time $k-1$, is independent of the model m_k, at time k, the prediction can be decomposed into:

$$
\begin{aligned}
p(d_k | m_k = \theta, y^{k-1}) &= \sum_{\alpha=1}^{M} \int p(d_k | d_{k-1}, m_{k-1} = \alpha, m_k = \theta, y^{k-1}) \times \\
&\quad p(d_{k-1} | m_{k-1} = \alpha, m_k = \theta, y^{k-1}) \mu_{k-1|k}^{\alpha|\theta}\, \mathrm{d}d_{k-1} \\
&= \sum_{\alpha=1}^{M} \int p(d_k | d_{k-1}, m_k = \theta, y^{k-1}) p(d_{k-1} | m_{k-1} = \alpha, y^{k-1}) \times \\
&\quad \mu_{k-1|k}^{\alpha|\theta}\, \mathrm{d}d_{k-1} \\
&= \int p(d_k | d_{k-1}, m_k = \theta, y^{k-1}) \sum_{\alpha=1}^{M} p(d_{k-1} | m_{k-1} = \alpha, y^{k-1}) \times \\
&\quad \mu_{k-1|k}^{\alpha|\theta}\, \mathrm{d}d_{k-1}
\end{aligned}
$$

The first integrand can be now identified as the transition density. It can be derived from the state dynamical equation:

$$
p(d_k | d_{k-1}, m_k = \theta, y^{k-1}) = p_{v_\theta}(d_k - f_\theta(d_{k-1}, u_k))
$$

where $v_\theta = v_{m_k=\theta}$ and $f_\theta = f_{m_k=\theta}$. The function f_θ is taken to be linear ($f_\theta = F_\theta d_{k-1} + u_\theta$). Since v_θ is modelled as Gaussian noise with zero mean and covariance

Q_θ , the transition density is:

$$p(d_k|d_{k-1}, m_k = \theta, y^{k-1}) = N(F_\theta d_{k-1} + u_\theta, Q_\theta) \tag{5.10}$$

According to the IMM approach, the second term in the integral is approximated by a normal distribution as follows: $p(d_{k-1}|m_{k-1} = \alpha, y^{k-1})$ is approximated by a Gaussian

$$p(d_{k-1}|m_{k-1} = \alpha, y^{k-1}) = N(\hat{d}^{\alpha}_{k-1|k-1}, P^{\alpha}_{k-1|k-1})$$

and the resulting mixture of Gaussians is also approximated by a Gaussian

$$\sum_{\alpha=1}^{M} p(d_{k-1}|m_{k-1} = \alpha, y^{k-1})\, \mu^{\alpha|\theta}_{k-1|k} = N(\hat{d}^{0\theta}_{k-1|k-1}, P^{0\theta}_{k-1|k-1})$$

where

$$\hat{d}^{0\theta}_{k-1|k-1} = \sum_{\alpha=1}^{M} \hat{d}^{\alpha}_{k-1|k-1}\, \mu^{\alpha|\theta}_{k-1|k} \tag{5.11}$$

$$\begin{aligned} P^{0\theta}_{k-1|k-1} = & \sum_{\alpha=1}^{M} \mu^{\alpha|\theta}_{k-1|k} \{ P^{\alpha}_{k-1|k-1} + \\ & [\hat{d}^{\alpha}_{k-1|k-1} - \hat{d}^{0\theta}_{k-1|k-1}][\hat{d}^{\alpha}_{k-1|k-1} - \hat{d}^{0\theta}_{k-1|k-1}]^T \} \end{aligned} \tag{5.12}$$

The prediction step can be carried out now:

$$\begin{aligned} p(d_k|m_k = \theta, y^{k-1}) &= \int p(d_k|d_{k-1}, m_k = \theta, y^{k-1}) \sum_{\alpha=1}^{M} p(d_{k-1}|m_{k-1} = \alpha, y^{k-1}) \times \\ &\quad \mu^{\alpha|\theta}_{k-1|k}\, \mathrm{d}d_{k-1} \\ &= \int N(F_\theta d_{k-1} + u_\theta, Q_\theta) N(\hat{d}^{0\theta}_{k-1|k-1}, P^{0\theta}_{k-1|k-1})\, \mathrm{d}d_{k-1} \\ &= N(\hat{d}^{\theta}_{k|k-1}, P^{\theta}_{k|k-1}) \end{aligned} \tag{5.13}$$

where the mean and covariance are given by:

$$\hat{d}^{\theta}_{k|k-1} = F_\theta \hat{d}^{0\theta}_{k-1|k-1} + u_\theta \tag{5.14}$$

$$P^{\theta}_{k|k-1} = F_\theta P^{0\theta}_{k-1|k-1} F^T_\theta + Q_\theta \tag{5.15}$$

The predicted model probability can be obtained by expanding $\mu^{\theta}_{k|k-1}$ as follows:

$$\begin{aligned}
\mu^{\theta}_{k|k-1} &= p(m_k = \theta | y^{k-1}) \\
&= \sum_{\alpha=1}^{M} p(m_k = \theta | m_{k-1} = \alpha, y^{k-1}) p(m_{k-1} = \alpha | y^{k-1}) \\
&= \sum_{\alpha=1}^{M} \Gamma_{\alpha\theta} p(m_{k-1} = \alpha | y^{k-1}) \\
&= \sum_{\alpha=1}^{M} \Gamma_{\alpha\theta} \mu^{\alpha}_{k-1|k-1} \qquad (5.16)
\end{aligned}$$

Measurements are assumed to be linear functions of the state and independent of the modal state m_k. Under these assumptions (5.3) reduces to $y_k = Hd_k + \psi_k$, and the first likelihood function $p(y_k|d_k, m_k = \theta, y^{k-1})$, simplifies to $p(y_k|d_k)$. As a result the first likelihood becomes:

$$p(y_k|d_k, m_k = \theta, y^{k-1}) = p(y_k|d_k) = N(Hd_k, \delta_k)$$

The second likelihood function $p(y_k|m_k = \theta, y^{k-1})$ needs further simplification;

$$\begin{aligned}
p(y_k|m_k = \theta, y^{k-1}) &= \int p(y_k, d_k|m_k = \theta, y^{k-1}) \, \mathrm{d}d_k \\
&= \int p(y_k|d_k, m_k = \theta, y^{k-1}) p(d_k|m_k = \theta, y^{k-1}) \, \mathrm{d}d_k \\
&= \int N(Hd_k, \delta_k) N(\hat{d}^{\theta}_{k|k-1}, P^{\theta}_{k|k-1}) \, \mathrm{d}d_k \\
&= N(H\hat{d}^{\theta}_{k|k-1}, HP^{\theta}_{k|k-1}H^T + \delta_k) = \lambda_k(\theta) \qquad (5.17)
\end{aligned}$$

$p(y_k|m_k = \theta, y^{k-1}) = \lambda_k(\theta)$ is also the normalisation factor of (5.6). The second normalisation factor given in (5.7) is evaluated using the derivation of the predicted model $p(m_k = \theta|y^{k-1})$ given in (5.16) as follows:

$$\begin{aligned}
p(y_k|y^{k-1}) &= \sum_{\theta=1}^{M} p(y_k|m_k = \theta, y^{k-1}) p(m_k = \theta|y^{k-1}) \\
&= \sum_{\theta=1}^{M} \lambda_k(\theta) \sum_{\alpha=1}^{M} \Gamma_{\alpha\theta} \, \mu_{k-1|k-1}(\alpha) \qquad (5.18)
\end{aligned}$$

Putting all the terms together in (5.6), the conditional density becomes:

$$\begin{aligned} p(d_k|m_k=\theta,y^k) &= \frac{p(y_k|d_k,m_k=\theta,y^{k-1})\,p(d_k|m_k=\theta,y^{k-1})}{p(y_k|m_k=\theta,y^{k-1})} \\ &= \frac{N(Hd_k,\delta_k)\,N(\hat{d}^{\theta}_{k|k-1},P^{\theta}_{k|k-1})}{N(H\hat{d}^{\theta}_{k|k-1},HP^{\theta}_{k|k-1}H^T+\delta_k)} \\ &= N(\hat{d}^{\theta}_{k|k},P^{\theta}_{k|k}) \end{aligned}$$

where the mean and the covariance are:

$$\hat{d}^{\theta}_{k|k} = \hat{d}^{\theta}_{k|k-1} + P^{\theta}_{k|k-1}H^T(HP^{\theta}_{k|k-1}H^T+\delta_k)^{-1}(y_k - H\hat{d}^{\theta}_{k|k-1}) \tag{5.19}$$

$$P^{\theta}_{k|k} = P^{\theta}_{k|k-1} - P^{\theta}_{k|k-1}H^T(HP^{\theta}_{k|k-1}H^T+\delta_k)^{-1}HP^{\theta}_{k|k-1} \tag{5.20}$$

substituting (5.16), (5.17) and (5.18) into (5.7) the conditional model probability simplifies to:

$$\mu_{k|k}(\theta) = \frac{\lambda_k(\theta)\sum_{\alpha=1}^{M}\Gamma_{\alpha\theta}\,\mu_{k-1|k-1}(\alpha)}{\sum_{\beta=1}^{M}\lambda_k(\beta)\sum_{\phi=1}^{M}\Gamma_{\phi\beta}\,\mu_{k-1|k-1}(\phi)} \tag{5.21}$$

The posterior density becomes a Gaussian mixture given by:

$$p(d_k|y^k) = \sum_{\theta=1}^{M} p(d_k|m_k=\theta,y^k)\mu_{k|k}(\theta)$$

The conditional mean $\hat{d}_{k|k}$ and covariance $P_{k|k}$ of this Gaussian mixture are given by:

$$\hat{d}_{k|k} = \sum_{\theta=1}^{M}\hat{d}^{\theta}_{k|k}\,\mu_{k|k}(\theta) \tag{5.22}$$

$$P_{k|k} = \sum_{\theta=1}^{M}\mu_{k|k}(\theta)\{P^{\theta}_{k|k} + [\hat{d}^{\theta}_{k|k} - \hat{d}_{k|k}][\hat{d}^{\theta}_{k|k} - \hat{d}_{k|k}]^T\} \tag{5.23}$$

The resultant estimates $\hat{d}_{k|k}$, $P_{k|k}$ can be found by evaluating the following equations in order (5.9), then (5.11-5.12), then (5.14-5.15), then (5.17) and finally (5.19-5.23).

5.2 Unsmooth drifts estimation and measurements correction algorithm

In this section we present a probabilistic approach that accounts for errors in sensors measurements and instantly captures drifts that have surges and sudden escalations. Such drift behaviour is not followed well by the KF algorithm given in the previous chapter. The standard KF with single drift model is limited in performance since it does not efficiently respond to changes in the dynamics as the drift changes abruptly at some points. Therefore, we make use of IMM in our solution since it is designed to deal with abrupt changes in the estimated states. The IMM approach is originally used in target tracking to track manoeuvring objects that show sudden changes in their dynamics [75, 76, 83, 84]. In accordance with the IMM algorithm, each sensor is assigned an M number of modes to account for the possible jumps in the drift. Our solution for the sudden step drift problem (also works for smooth drift) consists of the following iterative steps: As for the case of smooth drift, at time step k, a reading $r_{i,k}$ is made by node i. Rather than sending the reading as it is to it's neighbours, the node is aware of its drift $d_{i,k}$, and has an estimate for it at this stage. It is a projected value from an estimate of the drift made at the previous time step. Using this estimate of the drift, the node computes the corrected sensor reading $x_{i,k}$ and sends it to it's neighbours. Each sensor computes the average $\overline{x}_k = \sum_{i=1}^{n} x_{i,k}/n$ to self-assess it's measurements. To account for the possible jumps, the drift with abrupt changes is modelled as a jump markovian linear system. It is a system whose parameters evolve according to the realisation of a finite state markov chain [81]. Mathematically, we model the drift $d_{i,k}$ to belong to the set of models defined by (5.24):

$$\{d_{i,k} = d_{i,k-1} + u_i^{\theta} + v_{i,k}^{\theta}\}_{\theta=1}^{M} \quad v_{i,k}^{\theta} \sim N(0, Q_{i,k}^{\theta}) \tag{5.24}$$

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

same variance. Therefore, we denote the process variance for all the modes as $Q_{i,k}$.

Equation (5.24) represents an M number of possible drift models for each node. Each model differs from the others in the size of the jumps u_i^{θ}. The resultant estimated drift for node i at time instant k, $\hat{d}_{i,k|k}$, would be a weighted combination of the estimated drift of each model $\hat{d}_{i,k|k}^{\theta}$. The resultant estimated drift for each node $\hat{d}_{i,k|k}$ is found as will be shown later in this section by:

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

where $\mu_{i,k|k}^{\theta}$ is the model probability. It is the probability that the estimated drift $\hat{d}_{i,k|k}$ follows the drift model $\hat{d}_{i,k|k}^{\theta}$ given the measured values until the time step k.

A source of information is needed to provide input to a statistical model such as equation (5.24). Since the sensor measurement $r_{i,k}$ usually suffers from random error $w_{i,k}$ and systematic error (drift/bias) $d_{i,k}$, the reading or measurement of sensor i is given by:

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

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

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

Since the sensors are densely deployed and the instantiations of drifts in the sensors are random, we use the average of corrected sensors' measurements close to node i as an estimate for the expectation of actual (groundtruth) value $\overline{x}_k = E\{T_k\}$. We also define $y_{i,k}$ in (4.2) as the difference between the measurement $r_{i,k}$ and the average of corrected sensors measurements $\overline{x}_k$ and refer to $y_{i,k}$ as the drift measurement of node i at time instant k.

$$y_{i,k} = r_{i,k} - \overline{x}_k \tag{5.25}$$

At early stages of deployment of the sensor network when very few sensors have started to develop drift, and given that the instantiations of drifts in all the sensors are random, we assume that $E\{T_k\} = T_k$. Substituting $r_{i,k}$ into equation (5.25) results in:

$$\begin{aligned} y_{i,k} &= T_k + d_{i,k} + w_{i,k} - E\{T_k\} - \frac{1}{n}\sum_{j=1}^{n} w_{j,k} \\ &= d_{i,k} + w_{i,k} - \frac{1}{n}\sum_{j=1}^{n} w_{j,k} \\ &= d_{i,k} + \psi_{i,k} \quad \psi_{i,k} \sim N(0, \delta_{i,k}) \end{aligned} \tag{5.26}$$

where $\psi_{i,k} = w_{i,k} - \frac{1}{n}\sum_{j=1}^{n} w_{j,k}$ is the drift measurement noise and is actually a mixture of Gaussians. It is well known in literature [75, 76] that a Gaussian mixture can be approximated by a Gaussian $\psi_{i,k} \sim N(\pi_{i,k}, \delta_{i,k})$ with the mean found by the weighted sum of the means of the original Gaussians:

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

and the variance found by:

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

Referring to equations (5.24) and (5.26) we notice that they represent an M pairs of KF equations corresponding to M number of drift models (jumps). This leads according to the IMM algorithm to M number of KFs working in parallel to result in M number of estimations for drift and covariance. Each model has a probability $\mu^{\theta}_{i,k|k} = p\,(m_{i,k} = \theta | y_i^k)$ depending on the drift measurement values until that time step. Switching between models is governed by a pre-defined Markov transition

matrix Γ of dimension $M \times M$ for M models.

$$\Gamma = \begin{bmatrix} \gamma_{11} & \cdots & \gamma_{1M} \\ \vdots & \cdots & \vdots \\ \gamma_{M1} & \cdots & \gamma_{MM} \end{bmatrix} \tag{5.27}$$

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

The IMM step of our drift tracking algorithm is explained as follows: At time step k each node is supposed to know the previous time step models probabilities $\{\mu^{\theta}_{i,k-1|k-1}\}^{M}_{\theta=1}$, estimated drifts $\{\hat{d}^{\theta}_{i,k-1|k-1}\}^{M}_{\theta=1}$ and the associated covariances $\{P^{\theta}_{i,k-1|k-1}\}^{M}_{\theta=1}$. Unlike our standard KF drift tracking algorithm, the previous estimates are not used as priors for the M Kalman Filters. Instead, the predicted models probabilities $\{\mu^{\theta}_{i,k-1|k} = \sum^{M}_{\alpha=1} \gamma_{\alpha\theta}\, \mu^{\alpha}_{i,k-1|k-1}\}^{M}_{\theta=1}$ are calculated. Then the previous estimates together with $\{\mu^{\theta}_{i,k-1|k}\}^{M}_{\theta=1}$ are used in the mixing stage to calculate $\{\hat{d}^{0\theta}_{i,k-1|k-1}\}^{M}_{\theta=1}$ and $\{P^{\,0\theta}_{i,k-1|k-1}\}^{M}_{\theta=1}$. The mixing stage drift estimates and the associated covariances are then fed as priors to the corresponding M filters (substituted in KF equations (4.22-4.24)) to result in the posterior models estimates $\{\hat{d}^{\theta}_{i,k|k}\}^{M}_{\theta=1}, \{P^{\theta}_{i,k|k}\}^{M}_{\theta=1}$.

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

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

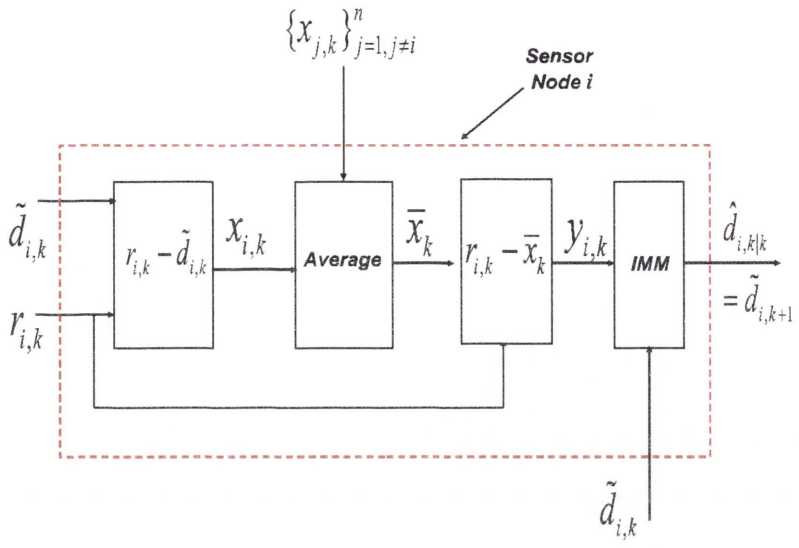


Figure 5.1: A block diagram for the unsmooth drift estimation and measurement correction algorithm

most use of the network.

Figure (5.1) shows a block diagram describing the unsmooth drift tracking algorithm. The IMM block in figure (5.1) is further broken down in figure 5.2. The steps of our unsmooth drift tracking algorithm are stated below:

Decentralised error correction algorithm for unsmooth drifts

For each node i

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

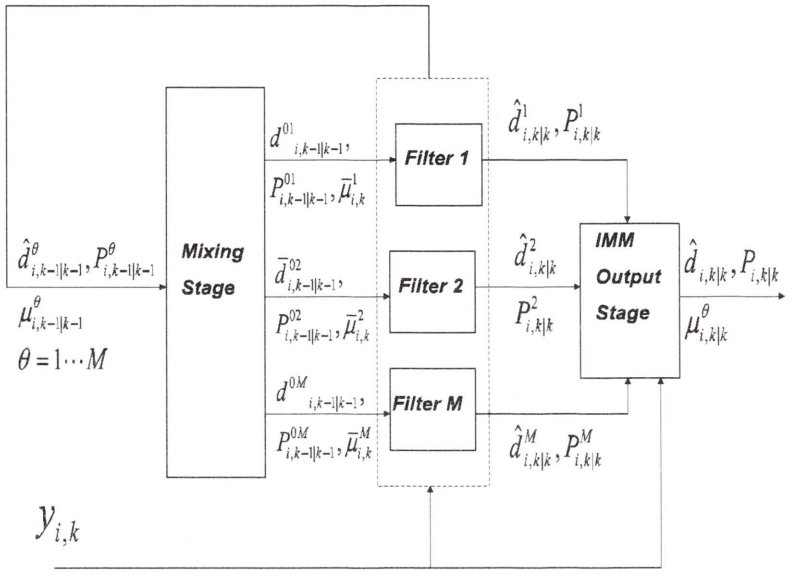


Figure 5.2: The IMM step, $\overline{\mu}^{\theta}_{i,k} = \mu^{\alpha|\theta}_{k-1|k}$

- The predicted model probabilities are calculated

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

- Mixing stage

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

$$\hat{d}^{0\theta}_{i,k-1|k-1} = \sum_{\alpha=1}^{M} \mu^{\alpha|\theta}_{k-1|k}\ \hat{d}^{\alpha}_{i,k-1|k-1}$$

$$P^{0\theta}_{i,k-1|k-1} = \sum_{\alpha=1}^{M} \mu^{\alpha|\theta}_{k-1|k} (P^{\alpha}_{i,k-1|k-1} + \{\hat{d}^{\alpha}_{i,k-1|k-1} - \hat{d}^{0\theta}_{i,k-1|k-1}\}^2)$$

- Kalman Filter update stage

$$\hat{d}^{\theta}_{i,k|k} = \hat{d}^{0\theta}_{i,k-1|k-1} + u^{\theta}_{i} + K(y_{i,k} - \hat{d}^{0\theta}_{i,k-1|k-1})$$

$$P^{\theta}_{i,k|k} = (P^{0\theta}_{i,k-1|k-1} + Q_{i,k})(1 - K)$$

$$K = \frac{P^{0\theta}_{i,k-1|k-1} + Q_{i,k}}{P^{0\theta}_{i,k-1|k-1} + Q_{i,k} + \delta_{i,k}}$$

- IMM output stage

  The Model probabilities are updated

$$\mu^{\theta}_{i,k|k} = \frac{\mu^{\theta}_{i,k-1|k}\, e^{-\frac{(y_{i,k} - \hat{d}^{0\theta}_{i,k-1|k-1} - u^{\theta}_i)^2}{2A}}}{\sum_{\theta=1}^{M} \mu^{\theta}_{i,k-1|k}\, e^{-\frac{(y_{i,k} - \hat{d}^{0\theta}_{i,k-1|k-1} - u^{\theta}_i)^2}{2A}}}$$

  where $A = P^{0\theta}_{i,k-1|k-1} + Q_{i,k} + \delta_{i,k}$. The resultant estimated drift and its associated covariance are updated as follows:

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

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

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

5.3 Complexity analysis

The computational complexity and memory complexity of our unsmooth drift correction algorithm is mainly contributed by the IMM algorithm computations. The IMM algorithm estimate of drift at each time step, involves running M KFs, evaluating the predicted model probability, the mixing stage outputs and evaluating the updated probability, the resultant drift and the resultant variance according to the equations listed in the previous section. This involves real number computations rather than matrix computations. Hence the computational complexity is only contributed by the number of multiplications and divisions of real numbers. Obviously, the computational complexity of the IMM algorithm is more than M times the computational complexity of the KF. To compare between the complexities of the two methods, we use in the next section the processing time required by each

algorithm as an indication. Memory complexity is mainly contributed by storing the values of the predicted model probability, the mixing stage outputs, the KF update stage outputs and the updated probability. The communication overhead in the network is contributed by the reporting of x among the neighbours.

5.4 Evaluation

Our aim is to evaluate the ability of our proposed framework to correct measurement errors experienced in a sensor node using the information gathered from the nearest neighbouring nodes. We simulate a small sub-network of 10 densely deployed sensor nodes measuring the temperature in a certain area. We assume that two sensors are developing smooth drifts with jumps of the forms shown in figure 1.3. We compare our IMM drift tracking algorithm with the plain KF drift tracking algorithm under the same drift and random error (noise) scenarios. The measurement noise is taken to be Gaussian with zero mean and variance 0.01. This means that the measurements will at most suffer form ± 0.3 C^o error due to noise only. The drift measurement variance $\delta_{i,k}$ for each node is chosen from [0.005-0.01] and the state variance $Q_{i,k}$ is taken to be 0.001.

The number of models we consider in our evaluation of IMM algorithm is $M = 11$. The results of the KF drift tracking algorithm are shown in figures 5.3 and 5.4, whereas, the results of the IMM drift tracking algorithm are shown in figures 5.5 and 5.6. Comparing figures 5.6 and 5.4, it is clear that both algorithms track the drift in node 1. However, the IMM drift algorithm performs considerably better. It follows the drift with jumps instantly with minimal errors and more efficiently than the plain KF drift tracking algorithm. Hence, the IMM based algorithm outperforms the KF based algorithm in terms of the speed and the accuracy of estimating the drift. It is also clear from figures 5.5 and 5.3 that both the IMM and the KF based algorithms correct the erroneous measurements by accounting for the drifts and the random noise to become very close to the actual temperature.

Looking back at figures 5.5 and 5.3, we notice that both the IMM and the KF

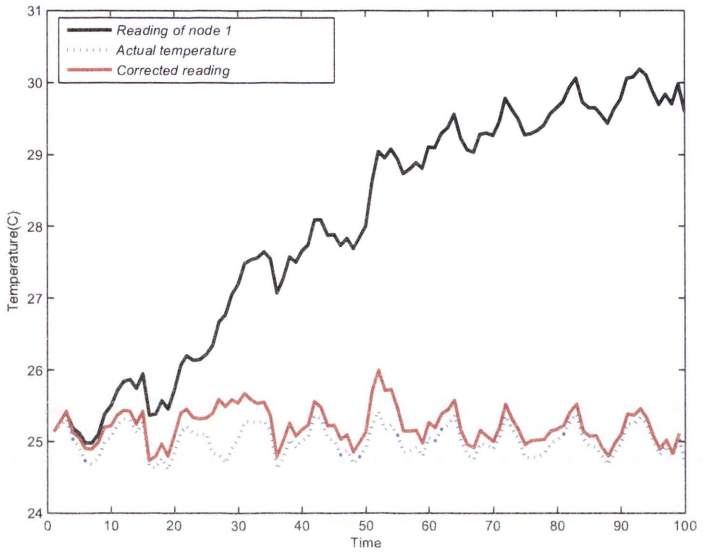


Figure 5.3: The reading of node 1, the corrected reading and the actual temperature for KF.

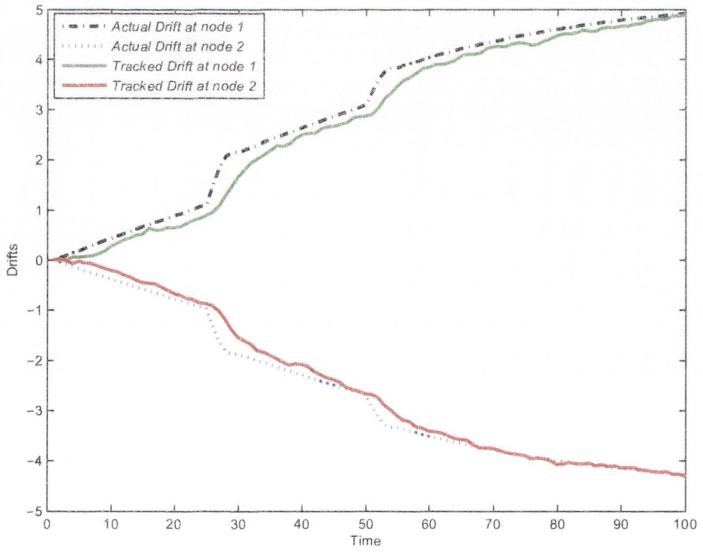


Figure 5.4: Actual and estimated drifts in nodes 1 and 2 for KF.

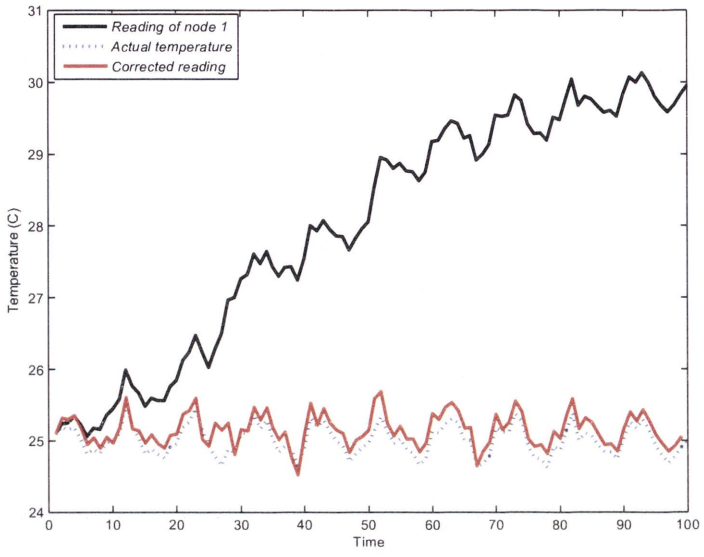


Figure 5.5: The reading of node 1, the corrected reading and the actual temperature for IMM.

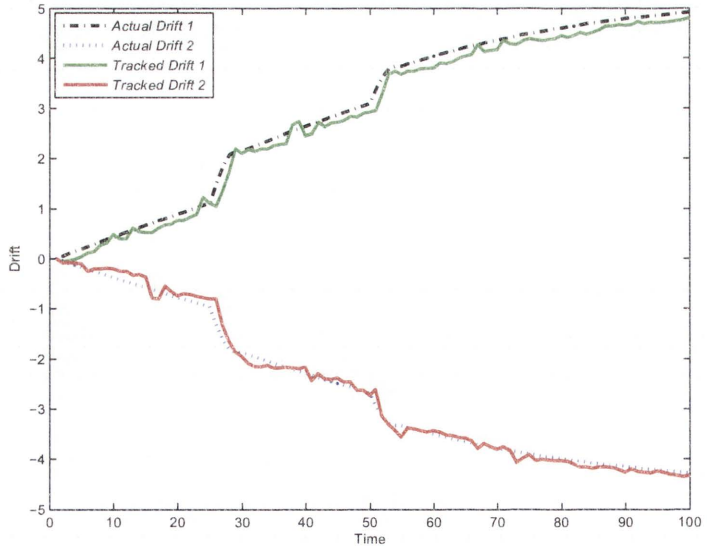


Figure 5.6: Actual and estimated drifts in nodes 1 and 2 for IMM.

drift tracking algorithms extend the effective operational life time for node 1. If we assume that for our application that the maximum tolerable temperature error in node's 1 reading is 1 C^o, then the life of node 1 is extended from 20 time units when there is no drift correction (Reading of node 1 curve) to at least 100 time units when the IMM or the KF algorithms is applied (Corrected reading curve). This applies to all of the network's sensors that develop drift. Hence, the life of the network will be extended by applying the drift tracking and correction algorithms. It is also worth noting from figures 5.5 and 5.3 that the difference between the actual and corrected reading curves tend to be in average smaller for the IMM algorithm results. This indicates that the error accumulation in the case of IMM is less and so it is expected to give longer life for the network.

In addition to the drift, we also address the bias problem. As we mentioned earlier, the bias is the initial reading error or in other words, the drift at time zero. We assume that the sensor nodes are factory calibrated before the deployment and so it is unlikely that a high percentage of the sensors will suffer from bias. In a sensible situation, if most of the sensors are without bias at time 0, then such a bias can be captured by both the KF and IMM based solutions as a constant drift of a certain amplitude. Figures 5.7 and 5.8 show the KF and IMM algorithms results when both sensors 1 and 2 suffer from bias and unsmooth drift.

It is obvious from both figures that the two algorithms efficiently capture and correct the bias. However, the IMM algorithm catches and corrects the bias faster. It is important to note here that if many sensors suffer from the bias, then both solutions will not be accurate, as we get little help from the neighbouring sensors to correct the bias. Furthermore, it is unrealistic to have many sensors with initial bias as the sensors have to be factory calibrated before deployment.

More comparisons between the two algorithms are shown in figure 5.9, when two sensors in the sub-network are subject to smooth drifts, and in figure 5.10, when the same two sensor are subject to unsmooth drifts. It is clear, in both scenarios, that the IMM drift tracking algorithm performs better than simple KF drift tracking algorithm in terms of speed of following the drift and in terms of the RMS

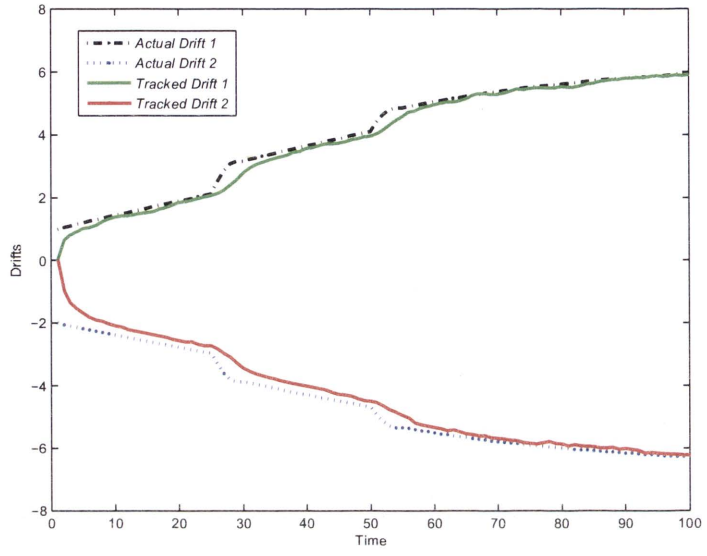


Figure 5.7: Actual and estimated biases/drifts in nodes 1 and 2 for KF.

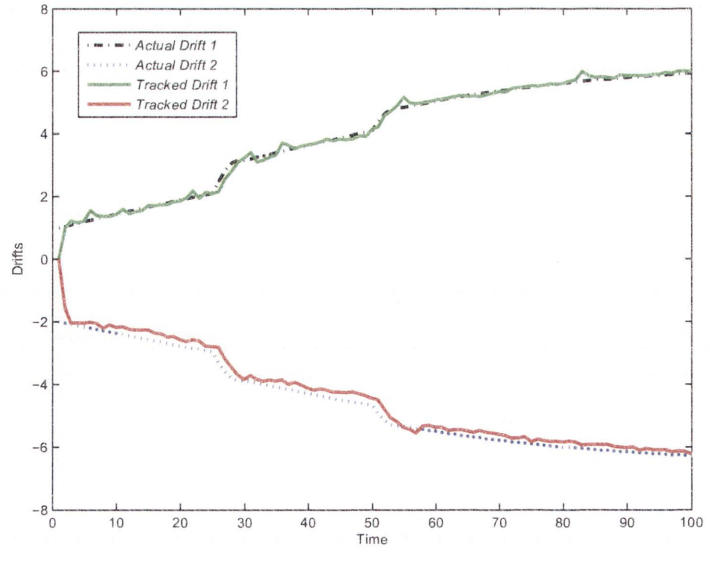


Figure 5.8: Actual and estimated biases/drifts in nodes 1 and 2 for IMM.

error between the estimated and actual drifts. However, the improved performance of IMM is at the cost of the increased computational complexity. We use the processing time required by each algorithm as an indication of its computational complexity. Table 5.1 shows the average processing time required by the KF based algorithm and the IMM based one (for different number of models) as reported by our MatLab simulations. The *Ratio* column clearly shows that the IMM based algorithm requires approximately $2M$ the time required by the KF based algorithm. Obviously, the computational complexity can be reduced by reducing the number of models M used in the IMM algorithm. Figure 5.11 shows the RMS error in the estimated drift for the KF based algorithm and the IMM based algorithm for different number of models. We notice that the RMS error reduces with M. However, the rate of change in the RMS error also decreases with M; the difference between the RMS errors for $M = 7$ and $M = 11$ is very small. In fact, it is well known in target tracking literature that using more models does not necessarily lead to better estimation, whereas it definitely increases the computational complexity [85]. Therefore, M should be chosen carefully. Alternatively, a model such as the variable structure IMM (VSIMM) which adaptively determines the minimal number models for estimating the state may be used [86].

It is important here to note that the speed of following the drift for the KF based algorithm can be increased by increasing $Q_{i,k}$. However, this will lead to more oscillatory response and will result in increasing the RMS error. Adding another component to the state vector, namely, the speed of the drift while maintaining the same value of $Q_{i,k}$, will result in faster tracking of the drift with less RMS error. Unfortunately, this will increase the mathematical complexity, as the problem of estimating the drift will then involve matrix multiplications and inversions, which is not desirable in a wireless sensor with limited computational capability.

We conducted several simulations and observed that the method worked as long as not all sensors start drifting at the same instant of time. Generally speaking, we noticed that the performance of both algorithms is dependent on the number of drifting sensors, the amplitude of drifts or biases and the starting times of the

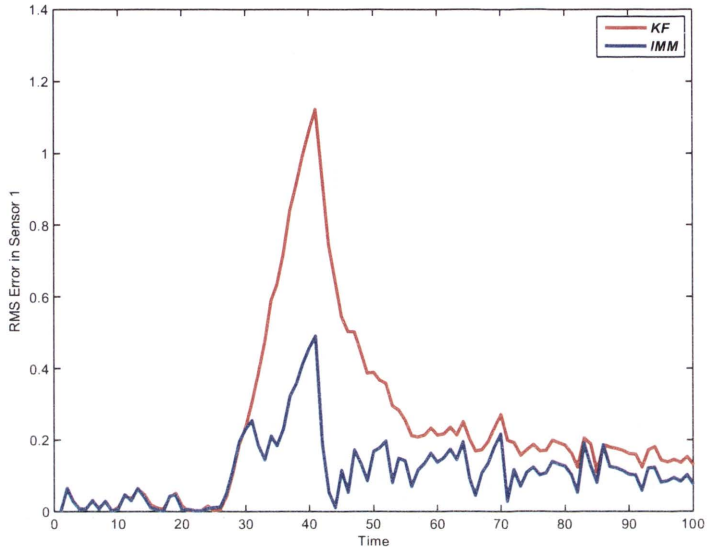


Figure 5.9: RMS error for both algorithms under smooth drift scenario.

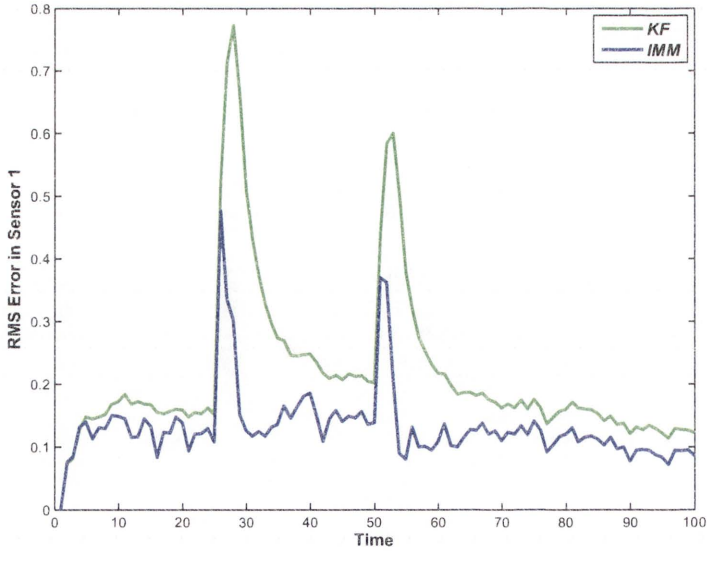


Figure 5.10: RMS error for both algorithms under unsmooth drift scenario.

Table 5.1: Processing times required by KF based and IMM based drift estimation and correction algorithms.

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

drifts. If all the sensors suffer from considerable biases at time zero the method will not work accurately. We went as far as 70% of the sensors drifting, and the corrections still made the network system data validity hold. It can be higher than this, but it depends on the error amplitudes.

5.5 Conclusion

In this chapter we have introduced a formal statistical procedure for correcting sensor errors in a WSN based on the assumption that neighbouring sensors have cor-

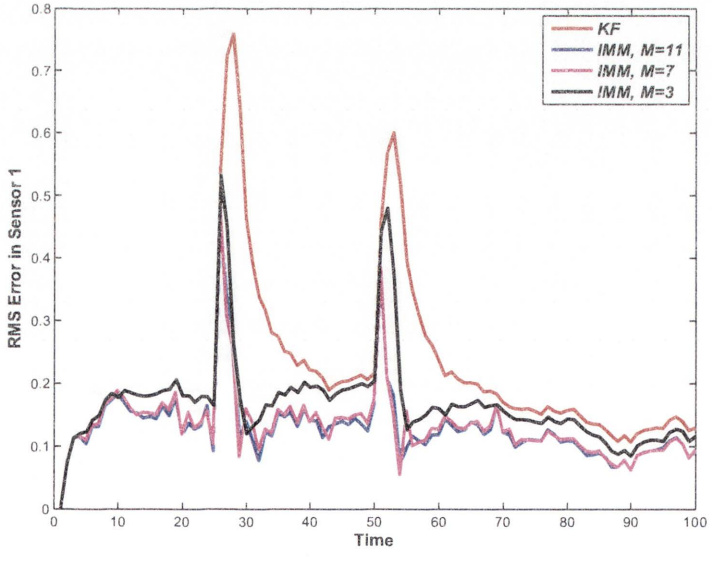


Figure 5.11: RMS error under unsmooth drift scenario for different number of models.

related measurements and that the instantiation of drift in a sensor is uncorrelated with other sensors. The sensors in the neighbourhood are assumed to be densely deployed. Hence, the average of their corrected readings is taken as a basis for each sensor to self-assess. The solution is designed to capture both smooth drifts and unsmooth drifts that have jumps and sudden escalations. It is also computationally simple as it is decentralised dealing with single state IMM based algorithm. Compared to KF solution introduced in the previous chapter, the IMM based algorithm performs better in detecting and correcting both the smooth drift and unsmooth one, however, at the cost of the computational complexity. It is important here to note that in the case of smooth drift, including the speed of the drift in the state model of the KF, would result in better estimate of the drift than that of the IMM solution (interms of accuracy and speed). However, that would involve the evaluation of matrix operations such as inversion which is not desirable in WSNs scenario.

Next chapter, we will address the problem when the sensors are not densely deployed and therefore the average cannot be used by the sensors to self-assess their readings. In this case each sensor is required to find a way to express it's reading in terms of its neighbour's measurements. As will been seen this is done using support vector regression.

Chapter 6

Spatio-Temporal Modelling of Measurements in Wireless Sensor Networks

IN this chapter, we introduce a more general solution for the problem of sensor measurement errors WSNs. We relax the assumption of the dense deployment of the sensor network. As a result, the measurements of the sensors within a cluster are assumed to be variable with distance and time. This means that the average of the sensors reported corrected readings cannot be used by the sensors to self assess their readings. Instead, the solution adopts the novel idea of using Support Vector Regression (SVR) by each sensor to model the spatio-temporal correlation among sensors in the cluster. SVR implements this in two phases, namely the *training phase* and the *running phase*. During the training phase, sensor measurements collected during the initial deployment period (training data set) are used to model a function which expresses the reading of one sensor in terms of the readings of its neighbours. The sensors at this stage are assumed to be correctly calibrated. During the *running phase*, each sensor uses its trained model to find a predicted value for its measurement in terms of the corrected measurements of its neighbours. It then uses this predicted value to self-assess its measurement and estimate its drift using a KF and then correct its measurement as was done in chapter 4.

Similar to the previously proposed solution in chapter 4, the solution presented here is designed for smooth drifts and is based on the assumption that neighbouring sensors have correlated measurements and that the instantiation of drift in a

sensor is uncorrelated with other sensors. The use of statistical modelling rather than physical relations to model the spatio-temporal cross correlations among sensors makes the framework presented here, in principle, applicable to most sensing problems. The algorithm also runs recursively and is fully decentralised. We demonstrate using real data obtained from the Intel Berkeley Research Laboratory that the algorithm successfully suppresses drifts developed in sensors and thereby prolongs the effective lifetime of the network.

6.1 Modelling and predicting measurements using SVR

Our aim is to predict the actual sensor measurements $\widetilde{x}_{i,k}$ of a sensor node i at time instant k using the corrected measurements from neighbouring sensors. Our intention is to learn a model function $f(.)$ that can be used for predicting the subsequent actual sensor measurements through out the whole period of the experiment. SVR implements this in two phases, namely the *training phase* and the *running phase*. During the training phase, sensor measurements collected during the initial deployment period (training data set) are used to model the function $f(.)$. During the *running phase*, the trained model $f(.)$ is used to predict the subsequent actual sensor measurements $\widetilde{x}_{i,k}$. Below we describe the mathematical formulation of SVR.

For simplicity, we consider a training data set represented by $X_s = \{(x_k, \acute{z}_k) : k = 1...m\}$ in the *input space*, where m is the number of training vectors. The data vectors X_s are mapped to a *feature space* via a non linear function $\phi(.) : \Re^d \rightarrow \Re^p$, resulting in image vectors $X = \{(\phi(x_k), z_k) : k = 1...m\}$, where p is the dimension of the feature space and d is the dimension of the input space. Our aim is to fit a function $f(\phi(x))$ to the data set in the feature space that has at most ϵ deviations from the targets z_k of the training input [87]. This means that errors less than ϵ are ignored and that errors greater than ϵ are penalised. For the linear case, the function $f(\phi(x))$ is given as:

$$f(\phi(x)) = \langle \omega.\phi(x) \rangle + \mathbf{b} \tag{6.1}$$

where $\langle \, . \, \rangle$ denotes the dot product in X, $\omega \in \Re^p$ and $\mathbf{b} \in \Re$ is the bias. Choosing a function $f(\phi(x))$ for X in the feature space that ensures a smooth function in the input space can be obtained by minimising the norm $\|\omega\|^2$ [87,88]. This leads to the following optimisation problem for support vector regression with an ϵ-intensive loss function [87,89].

$$\begin{aligned} \min \quad & \frac{1}{2}\|\omega\|^2 + C\sum_{k=1}^{m}(\xi_k + \xi_k^*) \\ \text{subject to:} \quad & z_k - \langle \omega.\phi(x)\rangle - \mathbf{b} \leq \epsilon + \xi_k \\ & \langle \omega.\phi(x)\rangle + \mathbf{b} - z_k \leq \epsilon + \xi_k^* \\ & \xi_k, \xi_k^* \geq 0, \; k = 1...m \end{aligned} \tag{6.2}$$

where $\{\xi_k, \xi_k^* : k = 1...m\}$ are the slack variables that allow some errors to be tolerated in approximating the function $f(\phi(x))$ using the input X (see Figure 6.1). The constant $C > 0$ determines the trade off between the complexity of the function $f(\phi(x))$ and the degree to which deviations more than ϵ are tolerated (see Figure 6.1). Using the Lagrange technique, a *dual* for the above *primal* optimisation problem (6.2) can be obtained. The Lagrange function for this optimisation problem is as follows:

$$\begin{aligned} L \;=\; & \frac{1}{2}\|\omega\|^2 + C\sum_{k=1}^{m}(\xi_k + \xi_k^*) - \sum_{k=1}^{m}(\eta_k\xi_k + \eta_k^*\xi_k^*) \\ & - \sum_{k=1}^{m}\alpha_k(\epsilon + \xi_k - z_k + \langle \omega.\phi(x)\rangle + \mathbf{b}) \\ & - \sum_{k=1}^{m}\alpha_k^*(\epsilon + \xi_k^* + z_k - \langle \omega.\phi(x)\rangle - \mathbf{b}) \end{aligned} \tag{6.3}$$

where $\{\alpha_k, \alpha_k^*, \eta_k, \eta_k^* \geq 0 : \forall k\}$ are the Lagrange multipliers. Equating the partial derivatives of L with respect to $\mathbf{b}$, ω, ξ_k and ξ_k^* to zero yields:

$$\frac{\partial L}{\partial \mathbf{b}} = 0 \;\Rightarrow\; \sum_{k=1}^{m}(\alpha_k - \alpha_k^*) = 0 \tag{6.4}$$

$$\frac{\partial L}{\partial \omega} = 0 \;\Rightarrow\; \omega - \sum_{k=1}^{m}(\alpha_k - \alpha_k^*)\phi(x_k) = 0 \tag{6.5}$$

$$\frac{\partial L}{\partial \xi_k} = 0 \quad \Rightarrow \quad C - \alpha_k - \eta_k = 0 \tag{6.6}$$

$$\frac{\partial L}{\partial \xi_k^*} = 0 \quad \Rightarrow \quad C - \alpha_k^* - \eta_k^* = 0 \tag{6.7}$$

Substituting (6.4) - (6.7) in (6.3) results in

$$\begin{aligned} L \;=\; & -\frac{1}{2}\sum_{k=1,l=1}^{m}(\alpha_k - \alpha_k^*)(\alpha_l - \alpha_l^*)\,\langle \phi(x_k).\phi(x_l)\rangle \\ & + \sum_{k=1}^{m} z_k(\alpha_k - \alpha_k^*) - \epsilon \sum_{k=1}^{m}(\alpha_k + \alpha_k^*) \end{aligned} \tag{6.8}$$

Therefore, the dual problem is as follows:

$$\begin{aligned} \max \quad & L \\ \text{subject to:} \quad & \sum_{k=1}^{m}(\alpha_k - \alpha_k^*) = 0, \\ & 0 \le \alpha_k, \alpha_k^* \le C,\ k = 1...m \end{aligned} \tag{6.9}$$

Further, using (6.5) and (6.1), we can obtain

$$\omega \;=\; \sum_{k=1}^{m}(\alpha_k - \alpha_k^*)\phi(x_k) \tag{6.10}$$

$$f(\phi(x)) \;=\; \sum_{k=1}^{m}(\alpha_k - \alpha_k^*)\,\langle \phi(x_k).\phi(x)\rangle + \mathbf{b} \tag{6.11}$$

This dual problem (6.9) is a quadratic optimisation problem with linear constraints. The solution can be obtained using widely available convex optimisation techniques such as the interior point methods [87,90,91].

From the Karush-Kuhn-Tucker (KKT) conditions [87,88,90] for the optimal solution of the dual problem (6.9), the data vectors that satisfy $|f(\phi(x_k)) - z_k| < \epsilon$ will have their $\{\alpha_k, \alpha_k^*\}$ as zero. Only data vectors that fall outside or on the ϵ-tube (i.e., $|f(\phi(x_k)) - z_k| \ge \epsilon$) will have nonzero $\{\alpha_k, \alpha_k^*\}$. These data vectors are called *support vectors*. Hence, only the support vectors, rather than the whole data set, are required to approximate the function $f(\phi(x))$. Therefore the approximate function

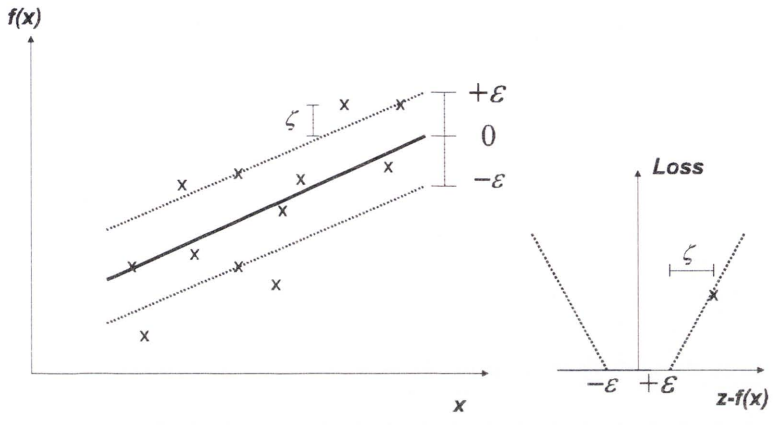


Figure 6.1: Support vector regression framework [91].

that models the input vectors becomes $f(\phi(x)) = \sum_{k=1}^{m_{sv}} (\alpha_k - \alpha_k^*) \langle \phi(x_k).\phi(x) \rangle + \mathbf{b}$, where $m_{sv} < m$ is the number of support vectors.

The bias $\mathbf{b}$ can be computed using support vectors that fall *on* the ϵ-tube [92] using the equation $\mathbf{b} = \text{average}_l\{\delta_l + z_l - \sum_s (\alpha_k - \alpha_s*) \langle \phi(x_s).\phi(x_l) \rangle)\}$, where $\delta_l = \epsilon \text{ sign}(\alpha_l - \alpha_l^*)$. In the case of using an interior point optimisation technique for optimising (6.9), $\mathbf{b}$ happens to be the by-product of that optimisation process [87].

Using Mercer kernels, the dot product computations of the image vectors in the feature space can be computed in the input space without any knowledge of the non-linear function $\phi(.)$. Hence the optimisation problem (6.2) can be expressed in terms of a Mercer *kernel function* $K(u, v) = \phi(u)\phi(v)^T$, by replacing the dot products with the kernel function. This is called the kernel-trick [89]. This provides a way to fit non linear regression to the data in the input space. Hence the function approximation (6.11) now becomes:

$$f(x) = \sum_{k=1}^{m_{sv}} (\alpha_k - \alpha_k^*) K(x_k, x) + \mathbf{b} \tag{6.12}$$

In this chapter, we use the Gaussian kernel $K(u, v) = exp(-\frac{\|u-v\|^2}{2\gamma_G^2})$ for the SVR with γ_G as the kernel width parameter. There are three modelling parameters that

need to be considered in using SVR to fix a model to the data, namely C,ϵ and γ_G. In practice, the parameter values are obtained by grid-search and cross-validation techniques [68,91]. Further, the number of support vectors produced also depends on these parameter settings [88,91], thereby providing flexibility in terms of obtaining the desired number of support vectors during training.

We use sensor measurements collected during the initial periods of deployment of the network as our training data set. We assume that this data is void of any drift and can be used for training the SVR at each node. This is a reasonable assumption in practice, as the sensors are usually calibrated before deployment to ensure that they are working in order. Hence the training data set we consider at each node i is given by $X_s = (TrX, TrZ)$, where $TrX = \{x_{j,k}, x_{j,k-1} : j = 1...n-1, k = 1...m, j \neq i\}$, $TrZ = \{x_{i,k} : k = 1...m\}$ and m is number of training data vectors.

After the training phase, each node i has only to keep in memory the support vectors, the non-zero Lagrange multipliers $(\alpha_k - \alpha_k*)$ and the bias **b**. This model obtained via SVR training will then be used during the *running phase* for predicting the subsequent actual measurements $\widetilde{x}_{i,k}$ using (6.12). The output of SVR and the readings $r_{i,k}$ are used by the *KF* to estimate the *drift* experienced in that sensor node and correct the reading error. Below we explain the KF formulation that utilises the SVR prediction for online error correction in sensor networks.

6.2 Iterative drift estimation and correction using SVR-KF framework

In this section, we consider a solution for the smooth drift problem under non dense deployment conditions. It is similar to the smooth drift detection and correction algorithm presented in section 4.1. However, instead of using the average of the neighbourhood $\overline{x}_{i,k}$, each sensor uses the predicted corrected reading $\widetilde{x}_{i,k}$ to evaluate the drift measurement $y_{i,k}$. $\widetilde{x}_{i,k}$ represents a better modelling for the spatio-temporal correlation of measurements within the cluster. The trained SVR of each node i finds $\widetilde{x}_{i,k}$ as an approximation to $T_{i,k}$ using the current and previous corrected

readings of the neighbours $\widetilde{x}_{i,k} = f(\{x_{j,k-1}, x_{j,k}\}_{j=1,j\neq i}^{n})$.

The solution to the smooth drift problem consists of the following iterative steps. At stage k, a reading $r_{i,k}$ is made by node i. Rather than sending that value directly to its neighbours, the node is aware of its drift, and has an estimate $\widetilde{d}_{i,k}$ for it at this stage. It is a projected value from an estimate of the drift made at the previous stage $\hat{d}_{i,k-1|k-1}$. Using this estimate of the drift, the node computes the corrected sensor measurement $x_{i,k}$ and sends it to its neighbours. Each node then collects all the neighbours' values and computes $\widetilde{x}_{i,k}$. To estimate the drift of a node, the mathematical model given in (6.13) is used, assuming smoothness in the way the drift changes.

$$d_{i,k} = d_{i,k-1} + v_{i,k} \quad v_{i,k} \sim N(0, Q_{i,k}) \tag{6.13}$$

where $v_{i,k}$ is the process noise and is taken to be a Gaussian noise with zero mean and variance $Q_{i,k}$.

A source of information is needed to provide input to a statistical model such as equation (6.13). Since the sensor measurement $r_{i,k}$ usually suffers from random error $w_{i,k}$ and systematic error (drift/bias) $d_{i,k}$, the reading or measurement of sensor i is given by:

$$r_{i,k} = T_{i,k} + d_{i,k} + w_{i,k} \quad w_{i,k} \sim N(\mu_{i,k}, R_{i,k}) \tag{6.14}$$

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

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

Since the sensors are not densely deployed, we use $\widetilde{x}_{i,k} = f(\{x_{j,k-1}, x_{j,k}\}_{j=1,j\neq i}^{n})$ (the output of the SVR modelled function) as an estimate for the expectation of actual (groundtruth) value $\widetilde{x}_{i,k} = E\{T_{i,k}\} + e_{i,k}$. $e_{i,k}$ is an error component and is ideally equal to zero when the SVR algorithm is well trained. We also define $y_{i,k}$ in (6.15) as the difference between the measurement $r_{i,k}$ and the SVR modelled value

$\widetilde{x}_{i,k}$ and refer to $y_{i,k}$ as the drift measurement of node i at time instant k.

$$y_{i,k} = r_{i,k} - \widetilde{x}_{i,k} \tag{6.15}$$

At early stages of deployment of the sensor network when very few sensors have started to develop drift, and given that the instantiations of drifts in all the sensors are random, we assume that $E\{T_{i,k}\} = T_{i,k}$. Substituting equation (6.14) into equation (6.15) results in:

$$\begin{aligned} y_{i,k} &= T_{i,k} + d_{i,k} + w_{i,k} - \widetilde{x}_{i,k} \\ &= T_{i,k} + d_{i,k} + w_{i,k} - E\{T_{i,k}\} - e_{i,k} \\ &= d_{i,k} + w_{i,k} - e_{i,k} \\ &= d_{i,k} + \psi_{i,k} \end{aligned} \tag{6.16}$$

If we assume $\psi_{i,k}$ is Gaussian ($\psi_{i,k} \sim N(0, \delta_{i,k})$), then $d_{i,k}$ can be estimated accurately by a KF [78]. Else the estimated drift will build error along time. In this chapter we adopt the assumption that $\psi_{i,k}$ is Gaussian ($\psi_{i,k} \sim N(0, \delta_{i,k})$) and therefore, we use the KF to estimate $d_{i,k}$. It can be seen from equations (6.13) and (6.16) that the state transition matrix F and the observation matrix H of the KF are equal to an identity matrix of dimension (1x1). This dramatically reduces the computational complexity of the KF, as no matrix multiplication or inversion is required for the evaluation of innovation covariance and the Kalman gain. As shown in section 4.1, this leads to the probabilistic solution for drift $d_{i,k} \sim N(\hat{d}_{i,k|k}, P_{i,k|k})$ with mean and variance given by:

$$\hat{d}_{i,k|k} = \hat{d}_{i,k-1|k-1} + K(y_{i,k} - \tilde{d}_{i,k}) \tag{6.17}$$

$$P_{i,k|k} = (P_{i,k-1|k-1} + Q_{i,k})(1 - K) \tag{6.18}$$

$$K = \frac{P_{i,k-1|k-1} + Q_{i,k}}{P_{i,k-1|k-1} + Q_{i,k} + \delta_{i,k}} \tag{6.19}$$

$\tilde{d}_{i,k}$ is the predicted drift at the beginning of stage k, before the correction. In this

case, $\tilde{d}_{i,k} = \hat{d}_{i,k-1|k-1}$, a straightforward prediction given by the KF solution. Note that $(y_{i,k} - \tilde{d}_{i,k}) = x_{i,k} - \widetilde{x}_{i,k}$. The variances $Q_{i,k}$, $\delta_{i,k}$, $P_{i,k-1}$ and $P_{i,k}$ are numbers, and therefore the solution is easy to compute in a sensor network with a very small complexity. Once $\hat{d}_{i,k|k}$ is evaluated, it is used as the predicted drift $\tilde{d}_{i,k+1}$ for the next stage. This allows for the correction of reading $r_{i,k+1}$.

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

Figure 6.2 shows a block diagram that summarises our SVR-KF drift correction framework in sensor i. The steps of our smooth drift tracking algorithm are also stated below:

Decentralised error correction using the SVR-KF framework

For each node i

- At step k, the predicted drift $\widetilde{d}_{i,k} = \hat{d}_{i,k-1|k-1}$ and the previous time step process variance $P_{i,k-1|k-1}$ are available.
- Each node i obtains its reading $r_{i,k}$
- The corrected reading is calculated, $x_{i,k} = r_{i,k} - \widetilde{d}_{i,k}$ and then transmitted to the neighbouring nodes.
- Each node computes $\widetilde{x}_{i,k}$ using the SVR equation (6.12).
- The drift measurement $y_{i,k} = r_{i,k} - \widetilde{x}_{i,k}$ is computed.
- Substituting into (6.17-6.19) results in the current time step estimates $\hat{d}_{i,k|k}$ and $P_{i,k|k}$.
- The projected drift $\widetilde{d}_{i,k+1} = \hat{d}_{i,k|k}$ is obtained and the algorithm iterates.

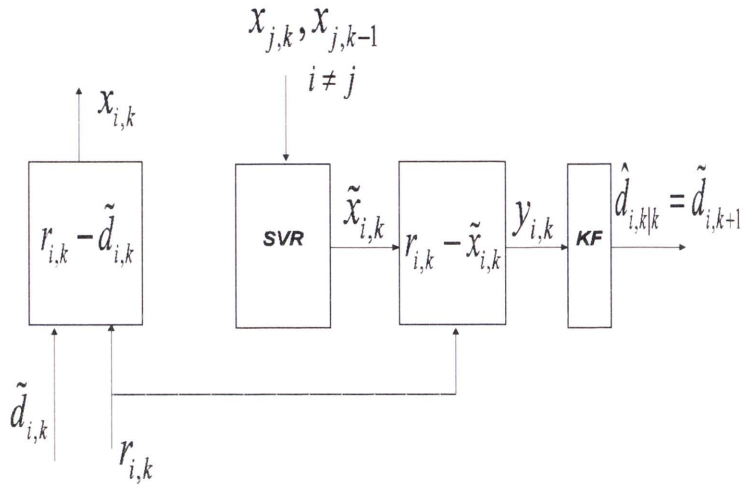


Figure 6.2: The SVR-KF drift estimation and measurement correction framework at node i.

6.3 Complexity analysis

The computational complexity and memory complexity of our online drift correction framework are mainly contributed by the SVR and KF computations.

The computational complexity for modelling data using SVR is mainly contributed by the training phase. Training the SVR involves a quadratic optimisation process. Polynomial time algorithms such as the interior point methods incur $O(m^3)$ arithmetic operations and have a complexity of $O(\sqrt{m}L_s)$ iterations [93]. m is the number of training vectors and L_s is the size of the optimisation problem, i.e., roughly the number of bits required to represent the problem. The overall complexity for the training phase is of $O(m^3)$. Alternatively, linear programming based support vector regressions can be used in place of the current quadratic programming based SVR [94, 95], which is advantageous in terms of computational complexity. There are several algorithms available for linear programming in the literature [93]. The simplex algorithm is extremely efficient in practice, although it has been shown to have worst case exponential complexity in the number of variables [96].

The training of SVR is usually done once or at low frequencies during the life time of the sensor network, unless the network is deployed in a highly dynamic environment. Hence they can be trained off-line or at a high performance node which is resourceful. Then the trained SVR can be uploaded to each sensor for use in the running phase. Note that only the support vectors are required to be uploaded to the sensors as opposed to the whole training data set. This way the communication and memory overhead in the sensor during the running phase can be reduced. Further, the SVR can be retrained if there is a significant change in the observed phenomenon. This can be done in an incremental fashion with less computational overhead. This is a topic of future research in this context.

The running phase of the SVR involves evaluating the output of the function $f(.)$ for the corrected measurements of the neighbourhood obtained at each time step. This involves a maximum computational complexity of $O(m_{sv}^2)$ in each sensor, where $m_{sv} << m$ is the number of support vectors. The memory complexity for the SVR training phase is $O(m + 2m_{sv} + n + 1)$ and for the running phase $O(2m_{sv} + 1)$. Each node has to communicate its corrected measurements among its closest neighbours in each time step. This involves a communication overhead of $O(n-1)$, where n is the number of nodes within the closest neighbourhood (cluster).

The KF estimates the drift at every time step using equations (6.17-6.19). This involves real number computations rather than matrix computations. Hence the computational complexity is only contributed by the number of multiplications and divisions of real numbers. Memory complexity is mainly contributed by storing the values of the estimated drift of the current and previous instances, current reading, and the predicted measurement from the SVR.

6.4 Evaluation

Our aim is to evaluate the ability of our proposed framework to correct the drift experienced in a sensor node using the information gathered from the nearest neighbouring nodes. The data in our evaluation are a set of real sensor measurements

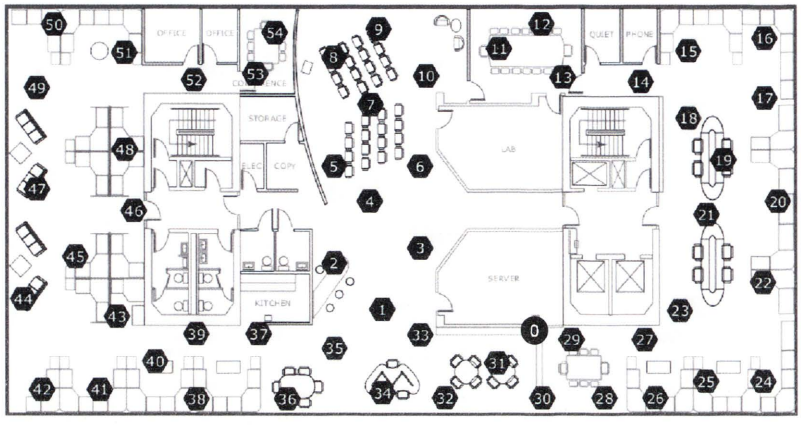


Figure 6.3: Sensor nodes in the IBRL deployment. Nodes are shown in black with their corresponding node-IDs. Node 0 is the gateway node [97].

gathered from a deployment of wireless sensors in the Intel Research Berkeley Laboratory (IBRL) [97].

In 2004, a set of wireless sensors with 55 sensor nodes (including a gateway node) were deployed in the IBRL lab for monitoring the lab environment (refer to Figure 6.3). They recorded temperature, humidity, light and voltage measurements at 30 seconds intervals during the period starting from 28th February 2004 to 5th April 2004.

We consider a network consisting of seven sensor nodes ($n = 7$) selected from the IBRL deployment. The node IDs considered here are 1, 2, 3, 4, 33, 34 and 35. These nodes come from a cluster of neighbouring sensors in the IBRL deployment (refer to Figure 6.3). Temperature measurements are used in our evaluations.

Data from the sensor nodes are re-sampled at seven minute intervals and the first 2000 samples are used for our evaluation purposes. This corresponds to the data collected during a ten day period from 28th February 2004 to 9th March 2004. We used the first 1000 samples (this corresponds to the first five days data) as the training set for use in the *training phase*. An exponential drift is introduced to the real data in each node, starting randomly after the first 1000 samples. The data after 1000 samples and up to 2000 samples are used in the *running phase* for testing our

algorithm for drift correction. These samples correspond to the next five days of the IBRL data.

Our algorithm is implemented in MatLab, utilising the SVM toolbox from [98]. For comparison purposes, we run the algorithm on two data sets. The first, is the data *without* the introduced drift (WOD), and the other, is the data *with* drift introduced (WD). Initially the SVR is trained on the first 1000 samples at each node. We fix the ϵ parameter to 0.05 in all our evaluation. Parameters C and σ are selected using grid search and cross-validation. In order to obtain this, we divide the training set (the first 1000 samples) into two parts. The first 60% of the training set is used for training the SVR and the other 40% is used for testing. Root mean square error $RMSE = \sqrt{\frac{\sum_{l=1}^{q}(y_l - f(x_l))^2}{q}}$, mean absolute error $MAE = \sum_{l=1}^{q} |y_l - f(x_l)|$ (where q is the number of training vectors) and number of support vectors are computed for each (C, γ_G) pair. The pair (C, γ_G) that yields lowest RMSE and MAE errors with a reasonable number of support vectors is selected and then used to obtain the completely trained SVR using the first 1000 samples for each node. The (C, γ_G) pairs satisfying these conditions are {(1600,40), (1600,40), (1600,80), (1600,40), (1600,80), (1600,160), (1600,40)} for the respective node IDs {1, 2, 3, 4, 33, 34 ,35}.

During the *running phase*, each sensor node i at time instant k uses the corrected measurements collected from neighbours $\{x_{j,k} : j = 1...n, j \neq i\}$ and the previous corrected measurements $\{x_{j,k-1} : j = 1...n, j \neq i\}$ as the input of the SVR. The SVR outputs the predicted measurement $\widetilde{x}_{i,k}$. $\widetilde{x}_{i,k}$ and the current reading $r_{i,k}$ are then fed to a KF to estimate the drift $\hat{d}_{i,k|k}$. Using the estimated drift $\hat{d}_{i,k|k}$ and the current measurement $r_{i,k}$, the corrected measurement $x_{i,k}$ is computed. This process is repeated at each time step for the rest of the 1000 samples. The KF parameters $Q_{i,k}$ and $\delta_{i,k}$ are tuned using trial and error. The values used in our evaluation are $Q_{i,k} = 0.001$ and $\delta_{i,k} = 2$. If $\delta_{i,k}$ is set to a high value, the estimated drift starts to follow the reading as it starts to trust the reading more than the SVR model. When $Q_{i,k}$ is high, the estimate becomes oscillatory and leads to an unstable state. Hence, a compromise has to be made in selecting these values in order to obtain good drift estimations.

We have conducted two simulations using two data sets. One data set has no drifts introduced. We denote this data set by *R-WOD*, which stands for *'Readings Without Drift'* and represents the sensor measurements that only suffer from noise. The other is the same data set with drifts introduced in seven different scenarios. We denote the readings of this data set by *R-WD*, which stands for *'Readings With Drift'* and represents the sensor measurements that suffer from both drift and noise. The seven drift scenarios considered in *R-WD* are as follows: scenario 1 being one node drifting, scenario 2 being two nodes drifting and so on until the last scenario (scenario 7) having all seven nodes drifting. The resulting corrected measurements obtained when the algorithm is run on the *R-WD* data set are denoted by *DCM-WD*, which stands for *'Drift Corrected Measurement for readings With Drift'*. Similarly, the corrected measurements obtained using data set *R-WOD* are denoted by *DCM-WOD*. It stands for *'Drift Corrected Measurement for readings Without Drift'*.

Figure 6.4 shows a graph obtained for the network having one of the nodes with drift introduced (namely, node ID 2). The plot shows curves for R-WOD, R-WD, DCM-WOD and DCM-WD during the sample period from 900 to 2000. From the graph, it can be observed that the DCM-WD curve is coinciding or very close to the DCM-WOD curve. This shows that the algorithm almost eliminates the introduced drift into the measurements with minimal error. We ideally expect the curve DCM-WOD (and DCM-WD curve) to coincide with the R-WOD curve. However, this is not the case as evident from the graph. This is because the SVR-KF framework introduces some system error (R-WOD - DCM-WOD) into the process during the operation. The reason behind that is that KF, by definition, assumes the linearity of the system and Gaussianity of the noise. However, $\psi_{i,k}$ may be here non Gaussian and the SVR is a highly nonlinear system. Therefore, the KF estimate will have some bias which will propagate among the neighbours and build overtime. A better candidate to solve the problem of system error is to use the Unscented Kalman filter (UKF) as will be seen next chapter.

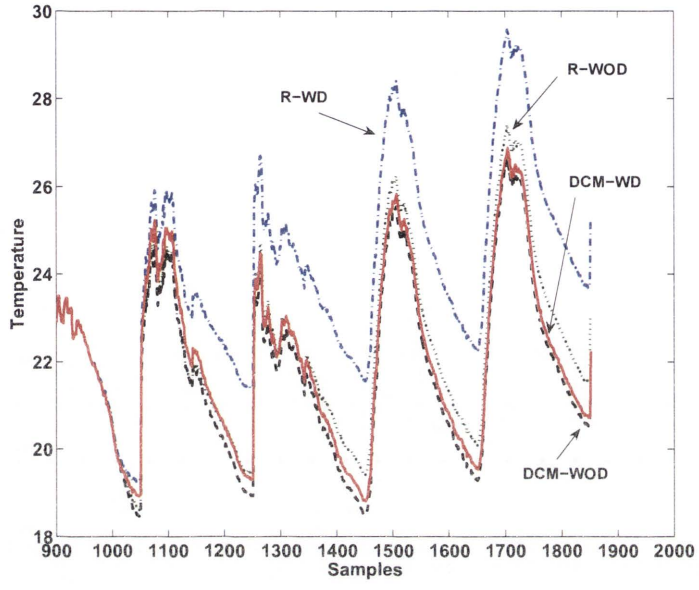


Figure 6.4: Results for node ID 2 when only this node experiences a drift. The curves shown are (i) R-WD (ii) R-WOD (iii) DCM-WD (iv) DCM-WOD

Looking back at the figure, it can be observed, as said earlier, that the gap between the DCM-WOD and R-WOD widens with time. This is because the errors introduced by the system accumulate as time progresses. If this trend is allowed to continue, the system becomes useless as the corrected measurements become overly erroneous. In practice, this accumulation of error can be limited by re-initialising the KF periodically during its operation.

Figure 6.5 shows the readings mean absolute error ($\overline{|R-WD-R-WOD|}$) for the whole network and for each of the 7 scenario (i.e, one sensor with drift, two sensors with drift and so on). This error is computed for each scenario as follows: For each node, and at each time instant, the absolute error between the observed readings R-WOD and R-WD is computed. Then the average error is computed by taking the average of the absolute errors of all the seven nodes. This graph, in essence, shows the mean absolute value of the drift introduced to the network in each of these scenarios, separately.

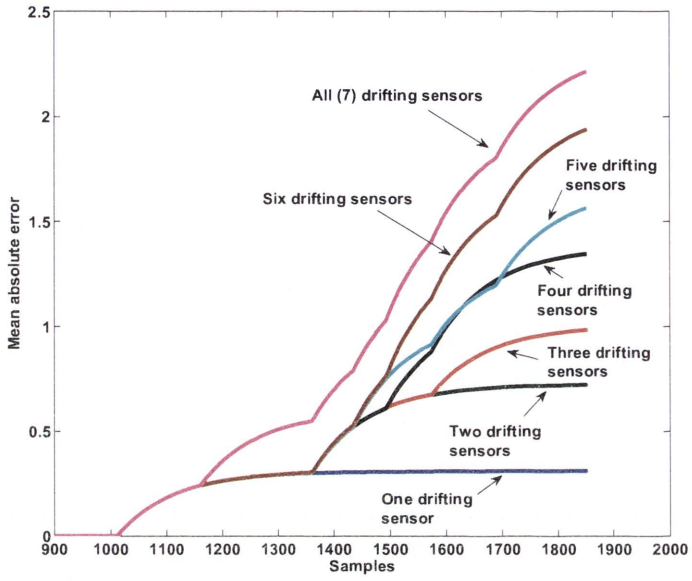


Figure 6.5: Mean absolute error of readings for each scenario.

As seen from figure 6.4, the DCM-WD curve incurs 2 error components. The first is the system error (R-WOD - DCM-WOD), the other, is the error in accounting for the drift (DCM-WOD - DCM-WD). Therefore, to judge the ability of the SVR-KF framework to eliminate the drift, we compare between the mean absolute error of the readings ($\overline{|R-WD-R-WOD|}$) and the mean absolute error of the corrected measurements ($\overline{|DCM-WD-DCM-WOD|}$).

Figure 6.6 shows the mean absolute error between the corrected measurements ($\overline{|DCM-WD-DCM-WOD|}$) for the whole network and for each of the 7 scenario. This figure reveals the effect of drift error in the network as a whole, as the number of sensors with drift increases. It is evident from this figure that when more than 50% of the nodes develop drift, the error starts to increase significantly and becomes comparable to the case when there is no correction applied (figure 6.5). For example, in the case of 3 sensors drifting, the maximum error with the error correction algorithm applied is approximately 0.4 as opposed to 0.85 when there is no correction. In contrast, the case when 4 sensors are drifting results in

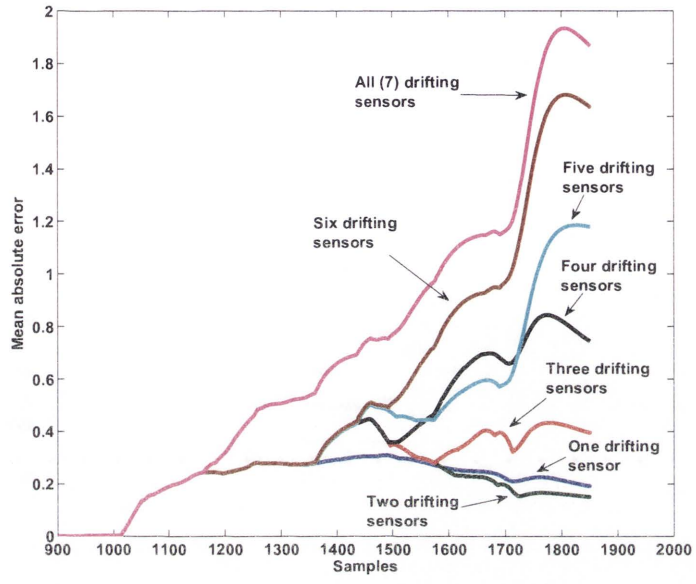


Figure 6.6: Mean absolute error of the corrected measurements for each scenario.

a maximum error of approximately 1.2 when the correction algorithm is applied, whereas it results in maximum error of 1.3 when no correction is applied.

Furthermore, comparing the two figures 6.5 and 6.6 for the scenario of 3 sensors drifting, it can be observed that the network with the error correction ability incurs a mean absolute error of 0.4 or less for the whole period of 2000 samples. In contrast, the network without the error correction ability, incurs a maximum mean absolute error of 0.4 up to the sample 1500. Beyond this the mean absolute error increases up to 1. This is an evidence that the network with error correction ability extends its useful life time compared to the one without the error correction ability.

6.5 Conclusion

In this chapter we have proposed a formal statistical procedure for estimating sensor errors in a non densely deployed WSN. The solution assumes that neighbouring sensors in a cluster have correlated measurements and that the instantiation of drift

in a sensor is uncorrelated with other sensors. We have used SVR to model sensor measurements by incorporating the spatio-temporal correlation among neighbouring sensors and then to predict future measurements. The prediction is used by a KF to estimate the drift in the sensor reading under consideration. The algorithm runs recursively and is fully decentralised. We have demonstrated using real data obtained from the IBRL that the algorithm successfully suppresses the drifts developed in sensors and thereby prolongs the effective life of the network. We have noticed that although the KF based algorithm estimates the drift and corrects the reading of a node, it introduces some system errors that accumulate with time. Therefore, in the next chapter, we incorporate instead, UKFs as they can deal with non Gaussian noises and nonlinear systems.

Chapter 7

Addressing Estimation Errors Caused by Nonlinearity of SVR

IN the previous chapter we have introduced an SVR-KF framework for detecting and correcting sensor measurement errors in a WSN. The solution did not require the sensors to be densely deployed. From the evaluations, we showed that the algorithm successfully eliminated the drift introduced into the measurements with minimal errors. However, we also noticed that the SVR-KF framework introduced some system errors (different from the sensor drift) into the process during the operation. The system error accumulated with time. If this were allowed to continue, The system corrected measurements would have become overly erroneous.

In this chapter we introduce a new algorithm that solves the above mentioned problem under the non dense deployment conditions. The solution may seem to be similar to the SVR-KF of the previous chapter, whereas it is substantially different. We use here UKFs instead of KFs. The use of UKF reduces the system error noticed in the SVR-KF evaluations since it is a better method for estimating the mean and the variance of a random variable propagating through a nonlinear system. The estimated variable here is the actual temperature whereas in the previous chapter it is the drift. The SVR is also used in a different fashion.

As we did in the previous chapter, we use the SVR as a form of statistical modelling rather than physical relations to model the spatio-temporal cross correlations among sensors. This, in principle, makes the framework presented in this chapter applicable to most sensing problems. The solution presented here does not assume linearity of the drifts and is fully decentralised. We demonstrate using real data

obtained from the IBRL that our algorithm successfully suppresses drifts and noise developed in sensors and thereby prolongs the effective lifetime of the network.

7.1 Modelling and predicting measurements using SVR

Similar to the previous chapter, we aim by using SVR to predict the actual sensor measurements $\widetilde{x}_{i,k}$ of a sensor node i at time instant k using the corrected measurements from neighbouring sensors. Our intention is to learn a model function $f(.)$ that can be used for predicting the subsequent actual sensor measurements through out the whole period of the experiment. SVR implements this in two phases, namely the *training phase* and the *running phase*. During the training phase, sensor measurements collected during the initial deployment period (training data set) are used to model the function $f(.)$. During the *running phase*, the trained model $f(.)$ is used to predict the subsequent actual sensor measurements $\widetilde{x}_{i,k}$.

We use the Gaussian kernel $K(u,v) = exp(-\frac{\|u-v\|^2}{2\gamma_G^2})$ for the SVR. There are three modelling parameters that need to be considered in using SVR to fix a model to the data, namely C,ϵ and γ_G. These were defined in the previous chapter. In practice, the parameter values are obtained by grid-search and cross-validation techniques [68, 91]. Further, the number of support vectors produced also depends on these parameter settings [88, 91], thereby providing flexibility in terms of obtaining the desired number of support vectors during training.

The sensor measurements collected during the initial periods of deployment of the network are used as our training data set. We assume that this data is void of any drift and can be used for training the SVR at each node. This is a reasonable assumption in practice, as the sensors are usually calibrated before deployment to ensure that they are working in order. Hence the training data set we consider at each node i is given by $X_s = (TrX, TrZ)$, where $TrX = \{x_{j,k-1} : j = 1...n, k = 1...m, j \neq i\}$, $TrZ = \{x_{i,k} : k = 1...m\}$ and m is number of training data vectors.

After the training phase, each node i has only to keep in memory the support vectors, the non-zero Lagrange multipliers $(\alpha_k - \alpha_k*)$ and the bias $\mathbf{b}$. This model

obtained via SVR training will then be used during the *running phase* for predicting the subsequent actual measurements $\widetilde{x}_{i,k}$ using (6.12). The output of SVR and the readings $r_{i,k}$ are used by the *UKF* to estimate the corrected measurement of that sensor. The SVR-UKF framework is explained in the next section.

7.2 Iterative measurement estimation and correction using an SVR-UKF framework

The solution to the smooth drift problem consists of the following iterative steps. At stage k, a reading $r_{i,k}$ is made by node i. The node also has a prediction for its corrected measurement (actual temperature at this sensor), $\widetilde{x}_{i,k}=f(\{x_{j,k-1}\}_{j=1,j\neq i}^{n})$, as a function of the corrected measurements of all neighbouring sensors in the cluster from the previous time step. Using this predicted value ($\widetilde{x}_{i,k}$) together with $r_{i,k}$, the corrected reading $x_{i,k}$ and the drift value $d_{i,k}$ are estimated. The node then sends the corrected sensor value $x_{i,k}$ to its neighbours. After that, each node collects the neighbourhood corrected measurements and computes $\widetilde{x}_{i,k}$ and so on. It is important here to emphasise that our main objective is to estimate $x_{i,k}$ the corrected reading which represents our estimate for the groundtruth value $T_{i,k}$ at node i. Assuming that $x_{i,k}$ and $d_{i,k}$ change slowly with time the dynamics of $x_{i,k}$ and $d_{i,k}$ are mathematically described by:

$$x_{i,k} = x_{i,k-1} + \eta_{i,k}^{(1)} \quad \eta_{i,k}^{(1)} \sim N(0, Q_{i,k}^{(1)}) \tag{7.1}$$

$$d_{i,k} = d_{i,k-1} + \eta_{i,k}^{(2)} \quad \eta_{i,k}^{(2)} \sim N(0, Q_{i,k}^{(2)}) \tag{7.2}$$

where $\eta_{i,k}^{(1)}$ and $\eta_{i,k}^{(2)}$ are the process noises. They are taken to be uncorrelated Gaussian noises with zero means and variances $Q_{i,k}^{(1)}$ and $Q_{i,k}^{(2)}$, respectively.

The value $x_{i,k}$ is never sensed or measured. What is really measured is $r_{i,k}$, the reading of the sensor. As we argued earlier, $r_{i,k}$ deviates from $x_{i,k}$ by both systematic and random errors. The random error is taken to be a Gaussian noise $w_{i,k} \sim N(0, R_{i,k})$ with zero mean and variance $R_{i,k}$ (measurement noise variance).

The systematic error is referred to as the drift $d_{i,k}$. This leads to (7.3).

$$y_{i,k}^{(1)} = r_{i,k} = x_{i,k} + d_{i,k} + w_{i,k} \quad w_{i,k} \sim N(0, R_{i,k}) \tag{7.3}$$

We also define $y_{i,k}^{(2)}$ as the difference between the measurement $r_{i,k}$ and the SVR modelled value $\widetilde{x}_{i,k}$ and refer to $y_{i,k}^{(2)}$ as the drift measurement of node i at time instant k.

$$\begin{aligned} y_{i,k}^{(2)} &= y_{i,k}^{(1)} - f(\{x_{j,k-1}\}_{j=1,j\neq i}^{n}) \\ &= x_{i,k} + d_{i,k} + w_{i,k} - f(\{x_{j,k-1}\}_{j=1,j\neq i}^{n}) \\ &= x_{i,k} + d_{i,k} + w_{i,k} - \widetilde{x}_{i,k} \quad w_{i,k} \sim N(0, R_{i,k}) \end{aligned} \tag{7.4}$$

The model is expressed in vector notation as follows:

$$X_{i,k} = \begin{bmatrix} x_{i,k} \\ d_{i,k} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{i,k-1} \\ d_{i,k-1} \end{bmatrix} + \begin{bmatrix} \eta_{i,k}^{(1)} \\ \eta_{i,k}^{(2)} \end{bmatrix} \tag{7.5}$$

$$Y_{i,k} = \begin{bmatrix} y_{i,k}^{(1)} \\ y_{i,k}^{(2)} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_{i,k} \\ d_{i,k} \end{bmatrix} + \begin{bmatrix} w_{i,k} \\ w_{i,k} \end{bmatrix} - \begin{bmatrix} 0 \\ \widetilde{x}_{i,k} \end{bmatrix} \tag{7.6}$$

The noise component associated with $X_{i,k}$ is Gaussian with mean vector $\mu_{X_{i,k}} = [0 \quad 0]^T$ and covariance matrix $Qx_{i,k} = \begin{bmatrix} Q_{i,k}^{(1)} & 0 \\ 0 & Q_{i,k}^{(2)} \end{bmatrix}$. The noise component associated with $Y_{i,k}$ has a mean vector $\mu_{Y_{i,k}} = [0 \quad 0]^T$ and covariance matrix $Ry_{i,k} = \begin{bmatrix} R_{i,k} & R_{i,k} \\ R_{i,k} & R_{i,k} \end{bmatrix}$ which indicates that it is not Gaussian. The system is clearly observable when $\widetilde{x}_{i,k} = x_{i,k}$, i.e. when $\widetilde{x}_{i,k}$ is a true, bias free, representation of $x_{i,k}$ and the difference between $x_{i,k}$ and $\widetilde{x}_{i,k}$ is zero.

Since the noise component associated with $Y_{i,k}$ is not Gaussian, the KF cannot be used [99] to estimate $x_{i,k}$ and $d_{i,k}$. Another filter that can be used for solving such a problem is the Particle Filter. Unfortunately, the high computational complexity of the Particle Filter makes it unsuitable for the use in WSNs, where the sensors

are limited in their energy and computational capabilities. A better alternative is to use the UKF. The Unscented Transformation (UT) was introduced by Julier et al. in [100] as an approximation method for propagating the mean and covariance of a random variable through a nonlinear transformation. This method was used to derive UKF in [101]. UKF can deal with versatile and complicated nonlinear sensor models and non-Gaussian noise that are not necessarily additive [75] with a comparable computational complexity to the Extended Kalman Filter (EKF) [102]. It also outperforms the EKF since it provides better estimation for the posterior mean and covariance to the third order Taylor series expansion when the input is Gaussian, whereas, the EKF, only achieves the first order Taylor series expansion [102]. Below, we explain the UKF algorithm in detail.

The UT as mentioned before is a method for finding the statistics of a random variable $Z = g(X)$ which undergoes nonlinear transformation. Let X of dimension L be the random variable that is propagated through the nonlinear function $Z = g(X)$. Assume that X has a mean $\hat{X}$ and a covariance P. According to [75], to find the statistics of Z using the scaled unscented transformation, which was introduced in [103], the following steps must be followed: First, $2L + 1$ (where L is the dimension of vector X) weighted samples or *sigma points* $\sigma_i = \{\mathcal{W}_i, \mathcal{X}_i\}$ are deterministically chosen to completely capture the true mean and covariance of the random variable X. Then, the sigma points are propagated through the function $g(X)$ to capture the statistics (mean and covariance) of Z. A selection scheme that satisfies the requirement is given below:

$$\begin{aligned}
\mathcal{X}_0 &= \hat{X}, \quad \mathcal{W}_0^m = \frac{\lambda}{\lambda + L} \\
\mathcal{W}_0^c &= \frac{\lambda}{\lambda + L} + (1 - \alpha^2 + \beta) \\
\mathcal{X}_i &= \hat{X} + (\sqrt{(L + \lambda)P})_i, \quad \mathcal{W}_i = \frac{1}{2(\lambda + L)} \\
\mathcal{X}_{L+i} &= \hat{X} - (\sqrt{(L + \lambda)P})_i, \quad \mathcal{W}_{L+i} = \frac{1}{2(\lambda + L)}
\end{aligned} \tag{7.7}$$

where $i = 1, ..., L$ and $\lambda = \alpha^2(L + \kappa) - L$ is a scaling parameter. α determines the

spread of the sigma points around the mean $\hat{X}$ and is usually set to a small positive value (*e.g.*, 0.001). κ is a secondary scaling parameter which is usually set to 0, and β is used to incorporate prior knowledge of the distribution of X. The optimal value of β for a Gaussian distribution is $\beta = 2$ as stated in [102]. The term $(\sqrt{(L+\lambda)P})_i$ is the ith row of the matrix square root of matrix $(L+\lambda)P$. In our work here α, κ and β are taken to be equal to 0.001, 0, 2, respectively. The UKF is used to estimate $X_{i,k}$ for sensor i at time step k. The dimension L of $X_{i,k}$ is equal to 2. This means that we only have five sigma points for each node i. The steps of the UKF algorithm are given below as in [75]:

Let $\hat{X}_{i,k-1|k-1}$ be the prior mean of the state variable and $P_{i,k-1|k-1}$ be the associated covariance for node i. To simplify the notation we write the prior mean of the state variable and the associated covariance as $\hat{X}_{k-1|k-1}$ and $P_{k-1|k-1}$ (without showing the sensor number i) keeping in mind that they refer to a certain sensor node i. This also applies for all the other parameters we use in describing the UKF algorithm.

The sigma points are calculated from (7.7) and then propagated through the state equation function $g(.)$. This results in $\mathcal{X}_{0,k|k-1}, \mathcal{X}_{1,k|k-1}, \mathcal{X}_{2,k|k-1}, \mathcal{X}_{3,k|k-1}$ and $\mathcal{X}_{4,k|k-1}$ as shown in (7.8).

$$\mathcal{X}_{k|k-1} = g(\mathcal{X}_{k-1}) = \mathcal{X}_{k-1} \tag{7.8}$$

The predicted mean and covariance of the state variable are given by (7.9) and (7.10), respectively.

$$\hat{X}_{k|k-1} = \mathcal{W}_0^m \mathcal{X}_{0,k|k-1} + \sum_{i=1}^{2L} \mathcal{W}_i \mathcal{X}_{i,k|k-1} \tag{7.9}$$

$$\begin{aligned} P_{k|k-1} &= \mathcal{W}_0^c (\mathcal{X}_{0,k|k-1} - \hat{X}_{k|k-1})(\mathcal{X}_{0,k|k-1} - \hat{X}_{k|k-1})^T \\ &+ \sum_{i=1}^{2L} \mathcal{W}_i (\mathcal{X}_{i,k|k-1} - \hat{X}_{k|k-1})(\mathcal{X}_{i,k|k-1} - \hat{X}_{k|k-1})^T + Qx_k \end{aligned} \tag{7.10}$$

The propagated sigma points are then passed through the measurement function $h(.)$ as shown in (7.11).

$$\mathcal{Y}_{k|k-1} = h(\mathcal{X}_{k|k-1}) \tag{7.11}$$

Then the predicted mean and covariance of each sensor measurement are given by (7.12) and (7.13), respectively.

$$\hat{Y}_{k|k-1} = \mathcal{W}_0^m \mathcal{Y}_{0,k|k-1} + \sum_{i=1}^{2L} \mathcal{W}_i \mathcal{Y}_{i,k|k-1} \tag{7.12}$$

$$\begin{aligned} P_{Y_k Y_k} &= \mathcal{W}_0^c (\mathcal{Y}_{0,k|k-1} - \hat{Y}_{k|k-1})(\mathcal{Y}_{0,k|k-1} - \hat{Y}_{k|k-1})^T \\ &+ \sum_{i=1}^{2L} \mathcal{W}_i (\mathcal{Y}_{i,k|k-1} - \hat{Y}_{k|k-1})(\mathcal{Y}_{i,k|k-1} - \hat{Y}_{k|k-1})^T + Ry_k \end{aligned} \tag{7.13}$$

The cross covariance of the predicted state and sensor measurement is found by (7.14).

$$\begin{aligned} P_{X_k Y_k} &= \mathcal{W}_0^c (\mathcal{X}_{0,k|k-1} - \hat{X}_{k|k-1})(\mathcal{Y}_{0,k|k-1} - \hat{Y}_{k|k-1})^T \\ &+ \sum_{i=1}^{2L} [\mathcal{W}_i (\mathcal{X}_{i,k|k-1} - \hat{X}_{k|k-1})(\mathcal{Y}_{i,k|k-1} - \hat{Y}_{k|k-1})^T] \end{aligned} \tag{7.14}$$

where

$$K_k = P_{X_k Y_k} P_{Y_k Y_k}^{-1} \tag{7.15}$$

The updated posterior mean and covariance of the state are then estimated by (7.16) and (7.17), respectively.

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k (Y_k - \hat{Y}_{k|k-1}) \tag{7.16}$$

$$P_{k|k} = P_{k|k-1} + K_k P_{Y_k Y_k} K_k^T \tag{7.17}$$

where $\hat{X}_{k|k}$ and $P_{k|k}$ are the mean and covariance of the state of node i at time step k.

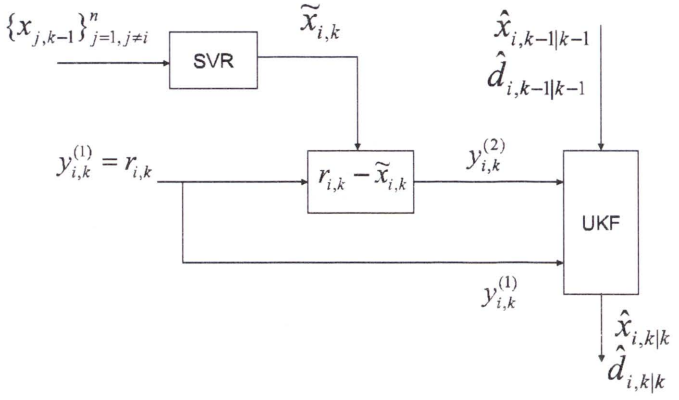


Figure 7.1: The SVR-UKF Measurement correction framework at node *i*.

Figure 7.1 shows a block diagram of our drift correction algorithm. It clearly summarises the stages of the SVR-UKF framework in one of the nodes in the cluster. The steps of the algorithm are stated below:

Decentralised error correction algorithm using the SVR-UKF framework

At time step k

- Each node i finds its predicted corrected measurement $\widetilde{x}_{i,k} = f(\{x_{j,k-1}\}_{j=1,j\neq i}^{n})$.
- Each node i obtains its reading $y_{i,k}^{(1)} = r_{i,k}$.
- Each node i calculates the drift measurement $y_{i,k}^{(2)}$.
- Each node i finds the sigma points $\sigma_i = \{\mathcal{W}_i, \mathcal{X}_i\}$ from $\hat{X}_{i,k-1|k-1} = [\hat{x}_{i,k-1|k-1}\ \hat{d}_{i,k-1|k-1}]^T$.
- For each node i, the sigma points are propagated through the state equation function $g(.)$.
- The UKF estimates the corrected measurement and the drift using (7.9)-(7.17) and then sends the result to the neighbouring nodes.
- The algorithm reiterates.

7.3 Complexity Analysis

The computational complexity and memory complexity of our online drift correction framework are mainly contributed by the SVR and the UKF computations.

The computational complexity for modelling data using SVR is mainly contributed by the training phase. Training the SVR involves a quadratic optimisation process. Polynomial time algorithms such as the interior point methods incur $O(m^3)$ arithmetic operations and have a complexity of $O(\sqrt{m}L_s)$ iterations [93]. m is the number of training vectors and L_s is the size of the optimisation problem, i.e., roughly the number of bits required to represent the problem. The overall complexity for the training phase is of $O(m^3)$. Alternatively, linear programming based support vector regressions can be used in place of the current quadratic programming based SVR [94, 95], which is advantageous in terms of computational complexity. There are several algorithms available for linear programming in the literature [93]. The simplex algorithm is extremely efficient in practice, although it has been shown to have the worst case exponential complexity in the number of variables [96].

The training of SVR is usually conducted once or at low frequencies during the life time of the sensor network, unless the network is deployed in a highly dynamic environment. Hence they can be trained off-line or at a high performance node with sufficient resources. The trained SVRs can then be uploaded to their respective sensors for use in the running phase. Note that only the support vectors are needed to be uploaded to the sensors as opposed to the whole training data set. This reduces the communication and memory overhead at each sensor. Further, the SVR can be retrained if a significant change is observed in the measured phenomenon. This can be done in an incremental fashion with less computational overhead. This is a direction for future research in this context.

The running phase of the SVR involves evaluating the output of the function $g(.)$ for the corrected measurements of the neighbourhood obtained at each time step. This involves a maximum computational complexity of $O(m_{sv}^2)$ in each sensor, where $m_{sv} << m$ is the number of support vectors. The memory complexity for

the SVR training phase is $O(m + 2m_{sv} + 1)$ and for the running phase is $O(2m_{sv} + 1)$. Each node has to communicate its corrected measurements among its closest neighbours in each time step. This involves a communication overhead of $O(n-1)$, where n is the number of nodes within the closest neighbourhood (cluster).

The standard UKF incurs a computational complexity of $O(\frac{13}{6}L^3 + 8Lp^2 + 2pL^2 + 8L^2 + p^3)$ with L as the dimension of the state vector and p as the dimension of the measurement vector [99]. In our case $L = 2$ and $p = 2$. The memory complexity is mainly contributed by keeping the previous time step estimates and the value predicted by the SVR.

The communication overhead in the network is contributed by the reporting of x among the neighbours. This overhead can be considerably reduced by applying adaptive sampling that is governed by the knowledge of the behaviour of the sensed phenomenon. If the measured phenomenon tends to change linearly, the frequency of reporting the values of the neighbours can be reduced. If the phenomenon tends to change in a highly nonlinear fashion, then the reporting frequency of the neighbour values should be increased. Hence, by incorporating an adaptive neighbour reporting mechanism with an adjustable frequency, guided by the level of nonlinearity present in the measurements, the communication overhead can be minimised in practice. This kind of adaptive strategy has been used previously for selective sampling and anomaly detection in sensor networks as in [104].

7.4 Evaluation

Our aim is to evaluate the ability of our proposed framework to correct the drift experienced in sensor nodes and to extend the functional life of the sensor network. The data in our evaluation are a set of real sensor measurements gathered from a deployment of wireless sensors in the IBRL [97].

Similar to chapter 6, the data from the sensor nodes are re-sampled at seven minute intervals and the first 2000 samples are used for our evaluation purposes.

This corresponds to the data collected during a ten day period from 28th February 2004 to 9th March 2004. We use the first 1000 samples (this corresponds to the first five days' data) as the training set for use in the *training phase*. An exponential drift is introduced to the real data in each node, starting randomly after the first 1000 samples. The data after 1000 samples and up to 2000 samples are used in the *running phase* for testing our algorithm for drift correction. These samples correspond to the next five days of the IBRL data. Temperature measurements are used in all our evaluations.

We formed two networks of sensors using nodes selected from the IBRL deployment. We call these as *case 1* and *case 2*. In case 1, a network is formed using seven sensor nodes selected from the IBRL deployment. The node IDs considered here are {1, 2, 3, 4, 33, 34 ,35}. These nodes form a cluster of neighbouring sensors in the IBRL deployment (refer to Figure 6.3). In this set up, each sensor communicates with all of its six neighbours in the network. In case 2, a network is formed using sixteen sensor nodes. Here, each sensor communicates only with its eight closest neighbours. The node IDs used and their respective neighbour list are shown in the first two columns of Table 7.1.

Our algorithm is implemented in MatLab, utilising the SVR toolbox from [98] and the UKF toolbox from [105]. For comparison purposes, we run the algorithm on two data sets. One, the data *without* the introduced drift (WOD), and the other, the data *with* drift introduced (WD). Initially, the SVR of each node is trained on the first 1000 samples of itself and its neighbours. We fix the ϵ parameter to 0.05 in all of our evaluations. Parameters C and γ are selected using grid search and cross-validation. In order to obtain this, we divide the training set (the first 1000 samples) into two parts. The first 60% of the training set is used to train the SVR and the other 40% is used for testing.

The root mean square error $RMSE = \sqrt{\frac{\sum_{l=1}^{q}(y_l - f(x_l))^2}{q}}$, the mean absolute error $MAE = \sum_{l=1}^{q} |y_l - f(x_l)|$ (where q is the number of training vectors) and the number of support vectors are computed for each (C, γ_G) pair. The pair (C, γ_G) that yields lowest RMSE and MAE errors with a reasonable number of support vec-

Table 7.1: Sensor nodes IDs, their assigned neighbours and the SVR parameters (C and γ_G) for Case 2 with 2 sampling rates.

Node ID	Neighbour node IDs	C(2001 *samples*)	γ_G(2001 *samples*)	C(4001 *samples*)	γ_G(4001 *samples*)
1	2, 3, 4, 6, 32, 33, 34, 35	3200	80	1600	40
2	1, 3, 4, 6, 33, 34, 35, 36	3200	80	1600	320
3	1, 2, 4, 6, 32, 33, 34, 35	3200	160	1600	320
4	1, 2, 3, 6, 7, 10, 33, 35	3200	80	1600	160
6	1, 2 ,3 , 4, 7, 8, 9, 10	3200	80	1600	160
7	1, 2 ,3 , 4, 6, 8, 9, 10	3200	160	1600	320
8	1, 2 ,3 , 4, 6, 7, 9, 10	3200	40	1600	160
9	1, 2 ,3 , 4, 6, 7, 8, 10	3200	40	1600	160
10	1, 2 ,3 , 4, 6, 7, 8, 9	3200	80	1600	160
31	1, 2, 3, 32, 33, 34, 35, 36	3200	40	1600	40
32	1, 3, 31, 33, 34, 35, 36, 37	3200	40	1600	160
33	1, 2, 3, 31, 32, 34, 35, 36	3200	320	1600	320
34	1, 2, 31, 32, 33, 35, 36, 37	3200	40	1600	80
35	1, 2, 31, 32, 33, 34, 36, 37	3200	80	3200	80
36	1, 2, 31, 32, 33, 34, 35, 37	3200	320	1600	320
37	1, 2, 31, 32, 33, 34, 35, 36	3200	80	1600	320

tors is selected and then used to obtain the completely trained SVR of each node. For case 1, the (C, γ_G) pairs are {(1600,40), (1600,40), (1600,80), (1600,40), (1600,80), (1600,160), (1600,40)} for the respective node IDs {1, 2, 3, 4, 33, 34 ,35}. For case 2, the (C, γ_G) pairs are shown in the 3rd and 4th columns of Table 7.1 against their node IDs. The 5th and 6th columns show the (C, γ) pairs for case 2 when doubling the sampling rate (4001 samples).

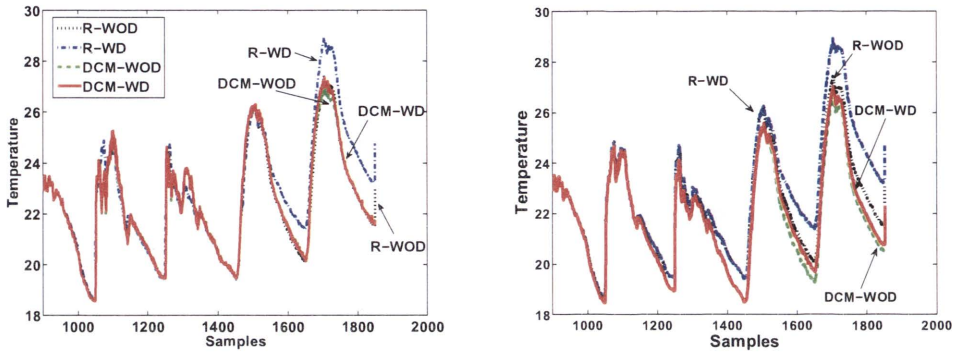
The UKF parameters μ, κ and β are set to the default values as explained in Section 7.2. Through out our evaluations, we take $Q_{i,k}^{(1)} = Q_{i,k}^{(2)} = Q_{i,k}$. $Q_{i,k}$ and $R_{i,k}$ are tuned using trial and error for both cases. The values used in our evaluation are $Q_{i,k} = 0.001$, $R_{i,k} = 0.01$ for case 1, and $Q_{i,k} = 0.001$, $R_{i,k} = 0.02$ for case 2. If $R_{i,k}$ is set to a high value, the estimated temperature will follow the reading (which may have drift) whereas if $R_{i,k}$ is set to a small value, the estimated temperature will not be able to follow the real temperature. Thus, it will not totally correct the drift. On the other hand, a high value for $Q_{i,k}$ will result in oscillatory estimates and lead to an unstable state. Hence, a trade off has to be considered in selecting the values for

$Q_{i,k}$ and $R_{i,k}$ to obtain the best results.

For each case (case 1 and case 2), we have conducted two simulations using two data sets. One data set has no drifts introduced. We denote this data set by *R-WOD*, which stands for *'Readings WithOut Drift'* and represents the sensor measurements that only suffer from noise. The other is the same data set with drifts introduced in several scenarios. We denote the readings of this data set by *R-WD*, which stands for *'Readings With Drift'* and represents the sensor measurements that suffer from both drift and noise. The drift scenarios considered in *R-WD* are as follows: scenario 1 (SCN 1) being one node drifting, scenario 2 (SCN 2) being two nodes drifting and so on until the last scenario (SCN 7 for case 1 and SCN 16 for case 2) having all nodes drifting. The resulting corrected measurements obtained when the algorithm is run on the *R-WD* data sets are denoted by *DCM-WD*, which stands for *'Drift Corrected Measurement for readings With Drift'*. Similarly, the corrected measurements obtained using data set *R-WOD* are denoted by *DCM-WOD*, which stands for *'Drift Corrected Measurement for readings WithOut Drift'*.

Figure 7.2(a) shows a graph obtained for the network of case 1 having one of the nodes with drift introduced (namely node ID 2). The plot shows curves for R-WOD, R-WD, DCM-WOD and DCM-WD during the sample period [900-2000] samples. From the graph, we can observe that the DCM-WD curve is coinciding or close to the DCM-WOD curve. This shows that the algorithm successfully eliminates the introduced drift into the measurements with minimal error. Further, the DCM-WD curve also coincides with the R-WOD curve. This also emphasises that the SVR-UKF framework successfully corrects the drifted measurements.

Similarly, Figure 7.2(b) shows a graph obtained for the network under the same drift scenario (drift introduced to node ID 2 only). The error correction algorithm that is applied for this figure uses the SVR-KF framework which was proposed in chapter 6. From the graph, we can observe that the DCM-WD curve is close to the DCM-WOD curve. Further, it can be seen that there is a gap between the DCM-WOD and R-WOD that widens with time. This means that error in the case of the SVR-KF framework accumulates with time. As discussed in chapter 6, the reason



(a) SVR and Unscented kalman filter framework

(b) SVR and Kalman filter framework

Figure 7.2: Results for node ID 2 when only this node experiences a drift. The curves shown are (i) R-WD (ii) R-WOD (iii) DCM-WD (iv) DCM-WOD.

for such a poor response of the SVR-KF framework is that the KF assumes linearity of the system while it is, in reality, nonlinear (especially the regression function $f(.)$). Also it assumes the Gaussianity of the drift measurement noise while it may be non Gaussian. In contrast, for the SVR-UKF framework (Figure 7.2(a)) the DCM-WD, DCM-WOD and R-WD curves nearly coincide. This clearly shows that the SVR-UKF framework outperforms the SVR-KF framework. The use of UKF gives better results as it can deal with non Gaussian noises and is specially designed for nonlinear systems.

To evaluate the performance of our algorithm from the network's point of view, we compare the average absolute error of all the sensors of the network with and without implementing our drift correction algorithm.

Figure 7.3 shows the mean absolute error between the true temperatures (R-WOD) and the values reported by the sensors (R-WD) for the whole network (case 2), for five different scenarios. The mean absolute error of the network is computed for each scenario as follows: for each node, at each instant of time, the absolute error between the true temperature (R-WOD) and the value reported by the sensors (R-WD) is computed. The average for all these nodes' absolute errors is then found. This gives the mean absolute error of the network. Similarly, the mean absolute error between the true temperatures (R-WOD) and the drift corrected measurements

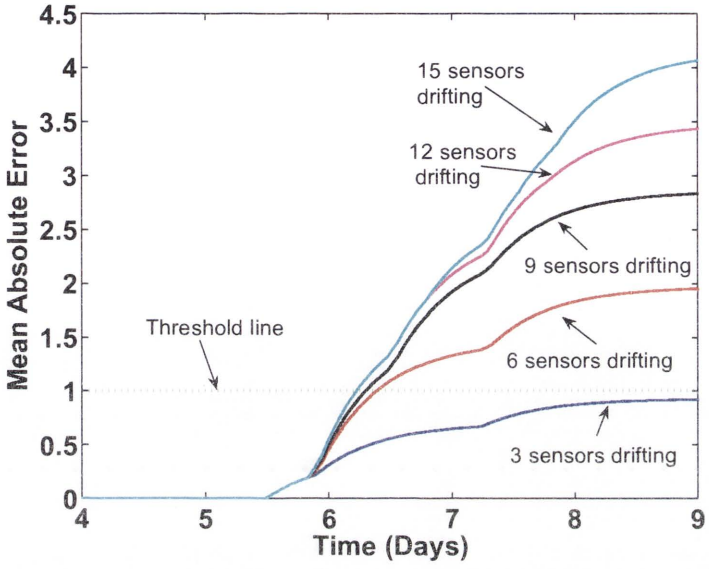


Figure 7.3: Mean Absolute Error for the network without correction.

(DCM-WD) is calculated at each instant of time and plotted in Figure 7.4. By comparing Figures 7.3 and 7.4 it is evident that applying the drift correction algorithm results in less measurements error for all of the scenarios. For our evaluation purposes we assumed that the maximum mean absolute error that can be tolerated in the network is 1^oC. If the mean absolute error of the network exceeds that limit, the network is deemed to be useless or has broken down. This maximum limit is shown by a horizontal threshold line in Figures 7.3 and 7.4. The choice of the threshold is dependent on the error tolerance allowed by the application.

In Figure 7.3, it is evident that the curves for scenarios 6, 9, 12 and 15 cross the threshold line after the 6^{th} day of the experiment. In contrast, in Figure 7.4, the curves for scenarios 6 and 9 do not cross the threshold line at all for the whole period of the experiment, while the curves of scenarios 12 and 15 cross the threshold line on the 8^{th} day and the 7^{th} day, respectively. This demonstrates that our algorithm extends the operational life of the network for all of the scenarios.

In another simulation we repeated case 2 after doubling the sampling rate. This resulted in 4001 samples for the 10 days experiment. Figure 7.5 shows the mean ab-

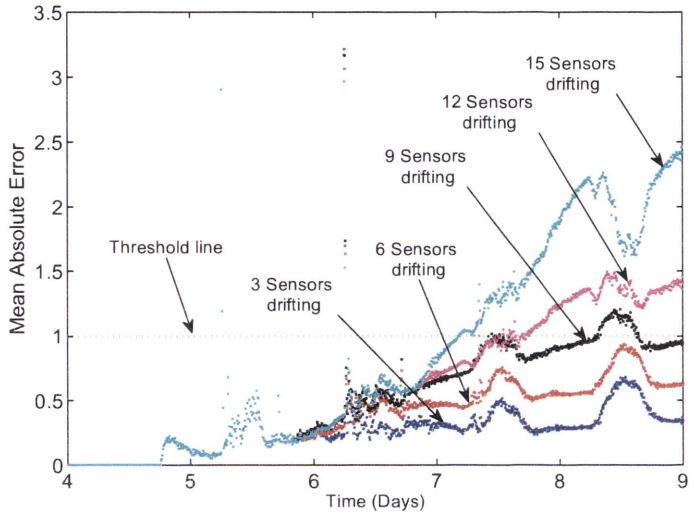


Figure 7.4: Mean Absolute Error for the network with correction for 2001 samples in 10 days

solute error between the true temperatures (R-WOD) and the drift corrected measurements (DCM-WD) for the whole network (case 2 for 4001 samples) for five different scenarios. The error is computed in a similar way to the method used for the Figures 7.3 and 7.4. By comparing Figures 7.3 and 7.5, it is evident that the application of the drift correction algorithm results in less measurements errors for all of the scenarios.

Looking at Figures 7.4 and 7.5, we can notice that the performance when using 4001 samples is better for scenario 9 since the absolute error curve does not cross the $1^{o}C$ threshold line as it does in the 2001 samples case. This means that the operational lifetime has been extended from around 8 days in the case of 2001 samples, to more than 9 days for the case of 4001 samples. Moreover, we notice in Figure 7.5 that the curves for each scenario are smoother than the corresponding curves in Figure 7.4 and that the observed occasional jumps and peaks are smaller. The jumps in the curves are caused by the fast changes in the readings or the ambient temperature at some instants of time. An effective way of reducing the size of the jumps is to increase the sampling rate as we noticed in Figure 7.5. However, that would be at a cost of the increased communication overhead due to the increased data trans-

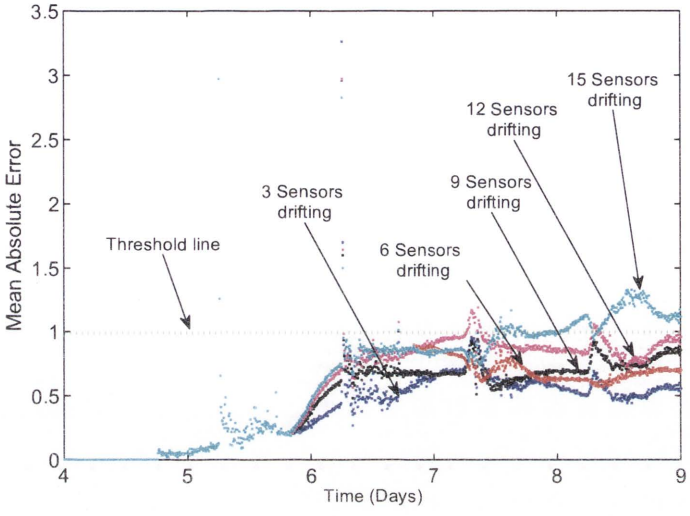


Figure 7.5: Mean Absolute Error for the network with correction for 4001 samples in 10 days

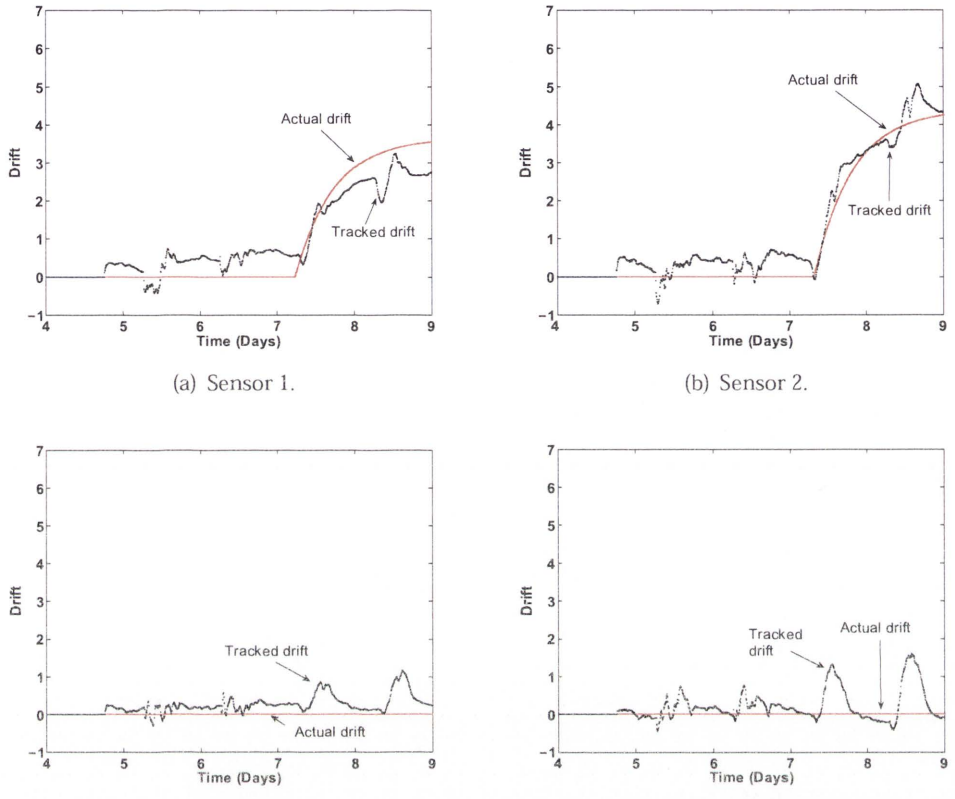
missions among the sensors. This means that a trade off between the smoothness of the curves and the communication overhead has to be made. A possible solution to reduce the size of the jumps without the need to increase the sampling rate is to use the IMM algorithm with the SVR-UKF framework. The IMM algorithm is used in target tracking to deal with manoeuvring objects that show sudden changes in their dynamics [76, 84]. Next chapter, we will use IMM with the SVR-UKF framework to enhance our drift correction algorithm as in [72]. Another important thing to note in both Figures 7.4 and 7.5 is that the mean absolute error of the network's estimated temperatures is proportional to the number of sensors developing drift.

The SVR-UKF framework efficiently reduces the drift generated in the sensors and extends the operational life of the network. Nevertheless, it induces some error in the estimates of the sensors that do not actually suffer from drifts. However, the introduced error is very small in comparison with the generated drift. This is demonstrated in figures 7.6 and 7.7, which show the real and estimated drifts for the case of 2001 samples and 4001 samples, respectively. Both figures are plotted for SCN 6 of case 2 showing only the graphs of 4 nodes (2 drifting nodes and 2 non drifting nodes) out of the 16 nodes present in the network. We notice from the

figures that the estimated drifts in both nodes 7 and 36 have some error induced by our algorithm, whereas it accurately tracks with some fluctuations the actual drifts of nodes 1 and 2. Moreover, the induced error as well as the fluctuations in the case of 2001 samples has higher peaks than those for the case of 4001 samples. It is clear that most of the time the induced error is negligible compared with the actual drift and our 1^oC threshold, except for some peaks especially in the 2001 samples case. This leads to the conclusion that a higher sampling rate results in less induced error and better estimation of the drift. This agrees with our previous observations for the mean absolute error of the network plotted in figures 7.4 and 7.5. The reason why increasing the sampling rate results in less error, is that our algorithm is derived for smooth changes in the observed phenomenon. However, our data set has some relatively fast changes in temperature. Hence, more samples will result in smoother variations in the measurements and thereby resulting in less error. As stated before, a possible alternative to deal with these fast changes is to integrate the IMM with our SVR-UKF framework.

The choice of $R_{i,k}$ and $Q_{i,k}$ is crucial. It affects the accuracy of estimating the temperature and the induced error. In general, we can say that increasing $R_{i,k}$ improves the tracking of drift in the drifting sensors. However, it also increases both the induced error in drift estimation in the non drifting sensors and the fluctuations in the drifting sensors. Since, the error caused by the fast changes in temperature in the case of 4001 samples is less than that for the case of 2001 samples (as explained previously), $R_{i,k}$ is taken to be 0.05 for the case of 4001 and 0.02 for the 2001 samples case. This way, the drift tracking is improved for the 4001 samples case keeping error levels comparable to the case of 2001 samples. On the other hand, increasing $Q_{i,k}$ increases the fluctuations in the estimated drift in both drifting and non-drifting sensors and causes the response to become less stable. The $Q_{i,k}$ used in all our simulations is equal to 0.001.

The drift correction performance for a sensor node in a cluster is dependent on the correlation of the actual temperature at the sensor under consideration with the actual temperatures at the neighbours. The SVR at a sensor predicts the ac-



(a) Sensor 1.

(b) Sensor 2.

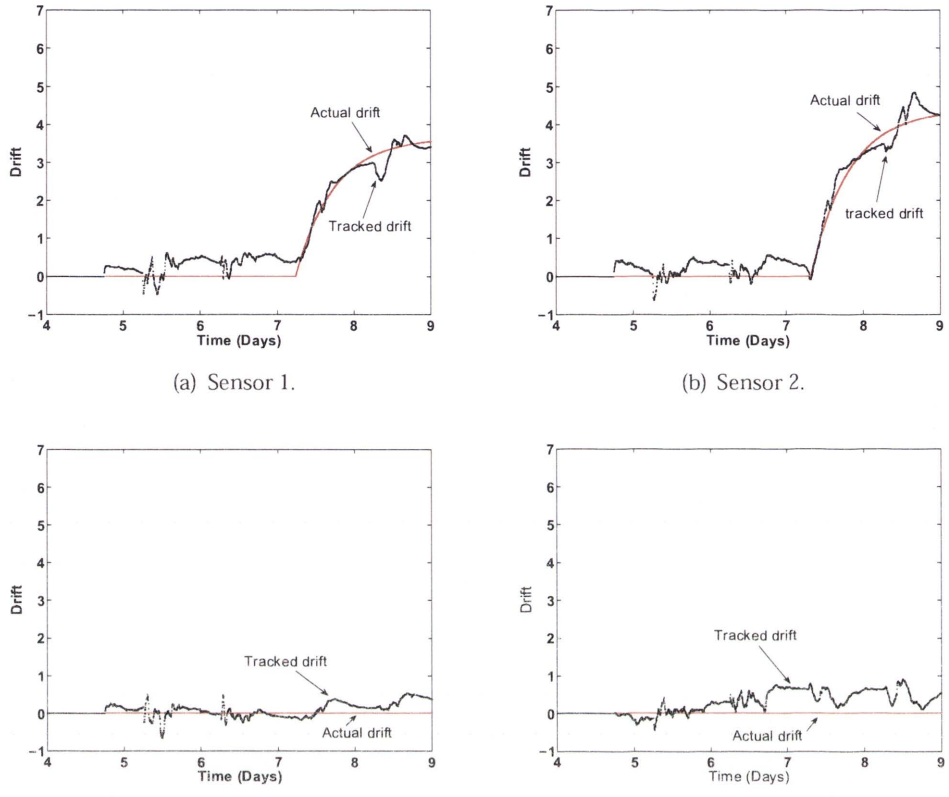
(c) Sensor 7.

(d) Sensor 36.

Figure 7.6: Estimated Drift in sensors with and without drift when the sampling rate is 2001 samples in 10 days.

tual temperature at the sensor $\widetilde{x}_{i,k}$ using previous estimates of the neighbourhood $\{\hat{x}_{j,k-1|k-1}\}_{j=1,j\neq i}^{n}$. Therefore, low correlation will lead to a poor prediction, and result in poor estimate of the actual temperature at the sensor under consideration. In practice, the correlation among the nodes may change depending on their spatial proximity within the cluster and with the change in the observed phenomenon along time.

It can be observed in the IBRL sensor deployment that not all the sensors were subject to the same conditions. This is because of their physical locations. Some of the nodes were closer to air conditioning. Some were closer to windows and hence were affected by the sun. Some were closer to the kitchen and thus affected by



(a) Sensor 1.

(b) Sensor 2.

(c) Sensor 7.

(d) Sensor 36.

Figure 7.7: Estimated Drift in sensors with and without drift when the sampling rate is 4001 samples in 10 days.

the heat and humidity coming from there. Furthermore, the patterns followed by sensor measurements changed seasonally. As an example, during a week period, the pattern followed in week days was different than that followed in weekends. That was because the air conditioning was reduced or turned off in the laboratory on the weekends. This caused the interrelationship among the sensors to vary not only with their spatial locations, but also with time.

We have chosen a node (Node ID 32) in the IBRL deployment to demonstrate how the correlations with its neighbours changes with time (the data considered here are void of any introduced drift). Table 7.2 shows the correlation coefficients ρ between the readings of node 32 and the readings of each one of its neighbours

for two time intervals: (1) the training phase (the first five days of the experiment), and (2) the running phase (the last five days). It can be observed that the correlation values, in general, vary between the training phase and the running phase. This change in correlation would affect the drift correction performance as the SVR model obtained in the training phase would not necessarily accurately predict the readings in the running phase.

A solution to overcome such a problem is to choose the neighbour sensors of each node so that they are physically close and subject to similar conditions. An alternative solution is to upgrade the model to become incremental with time to account for phenomenal changes. This can be achieved using *incremental learning* of the SVR. The learning process can then be performed at each time step (incrementally) or at predefined short intervals, depending on how severe the change is. Incremental SVR learning algorithms [106,107] can be utilised with the UKF to perform adaptive drift correction in the network. Devising an adaptive drift correction framework by incorporating incremental learning is a direction for future work.

Table 7.2: Correlation Coefficients of Node ID 32 with it's neighbours at the training phase ρ_t and running phase ρ_r.

Neighbours' ID	1	3	31	33	34	35	36	37
ρ_t	0.969	0.939	0.998	0.990	0.804	0.846	0.775	0.877
ρ_r	0.948	0.913	0.998	0.974	0.851	0.804	0.845	0.845

7.5 Conclusion

In this chapter we have proposed a formal statistical procedure for detecting and correcting sensor errors in a non densely deployed WSN based on the assumption that neighbouring sensors have correlated measurements and that the instantiation of drift in a sensor is uncorrelated with other sensors. We have used SVR to model the interrelationships of sensor measurements in a neighbourhood. This enables us to incorporate the spatio-temporal correlation of neighbouring sensors, in order to predict future measurements. The prediction is used by a UKF to estimate the actual value of the measured variable (here temperature) at the sensor

under consideration. The algorithm runs recursively and is fully decentralised. No assumptions regarding the linearity of drift or the density (closeness) of sensor deployment are made. We use statistical modelling (our SVR-UKF framework) rather than physical modelling to model the spatio-temporal correlation among sensors. This makes the framework presented in this chapter applicable to most sensing problems. We have demonstrated using real data obtained from the IBRL that the algorithm successfully suppresses the errors developed in sensor measurements and thereby prolongs the effective life of the network.

A major source of error in our measurement estimation and correction method (though small) is the sudden steep change in the measured variable. This is dealt with in this chapter by increasing the sampling rate of the data in a trade-off with the communication overhead. Next chapter, we will use the idea of the IMM with our SVR-UKF framework to minimise that problem. In future, we intend to implement an incremental SVR framework to periodically re-train the SVR, in order to adapt to any phenomenal changes that may occur in the network. Moreover, we intend to apply this solution to a WSN deployed in an outdoor environment, since the interrelationships among sensors in such an environment are expected to be different.

Chapter 8

Coping with Unsmooth Measurements and Under Sampled Data

A major source of error in our measurement estimation and correction method (though small) presented in the previous chapter is the sudden steep change in the measured variable. This was dealt with in that chapter by increasing the sampling rate of the data in trade off with the communication overhead and energy consumed in communication. In this chapter, we use the idea of IMM with the SVR-UKF framework to minimise that problem. The advantage of using the IMM algorithm in this context is two-fold: It can follow (track) data that suffer from sharp changes and sudden jumps. Secondly, it can deal with jumps in the readings caused by lowering the sampling rate. This means that using IMM will allow reducing the communication between sensors to maintain the calibration. This is expected to reduce the energy consumed from the batteries.

The proposed algorithm is designed for non-densely deployed sensor networks. It is tested on real data obtained from the IBRL sensor deployment. The results show that the IMM-SVR-UKF framework performs better than SVR-UKF framework in terms of smoothness and stability of the networks' mean absolute error curve for considerably lower sampling rate. A direct result for that is reducing communication among the sensors and eventually saving the energy consumed in communication.

8.1 Iterative measurement estimation and correction using SVR with UKF based IMM algorithm

The algorithm presented in the previous chapter was designed taking into consideration that sensors readings and drifts change smoothly. However, the measurements taken by the sensors and the drifts as well may sometimes have relatively fast changes and jumps, either due to the actual fast change in the measured phenomenon, or due to reducing the sampling rate for the sake of reducing the communication overhead. In this case, we use IMM for better tracking of the fast changes in the readings and the drifts as was done for unsmooth drifts in chapter 5.

To account for the possible jumps, the corrected reading with abrupt changes and also the drift with abrupt changes are modelled as a jump markovian system. Mathematically, the state dynamics of jump markovian systems are assumed to belong to the set of models defined by (8.1):

$$\{X_{i,k} = X_{i,k-1} + u_i^\theta + \eta_{i,k}^\theta\}_{\theta=1}^{M} \quad \eta_{i,k}^\theta \sim N(0, Qx_{i,k}^\theta) \tag{8.1}$$

where $X_{i,k} = [x_{i,k} \quad d_{i,k}]^T$ is the state vector with the corrected measurement $x_{i,k}$ and the drift $d_{i,k}$ as its states, $\theta = 1, 2, \ldots M$ is the model number, $u_i^\theta = [ux_i^\theta \quad ud_i^\theta]^T$ is the input vector or jump vector corresponding to θ_{th} model for the i_{th} sensor and $\eta_{i,k}^\theta$ is the process noise vector for each model. $\eta_{i,k}^\theta$ is taken to be Gaussian with mean vector $\mu_{X_{i,k}} = [0 \quad 0]^T$ and covariance matrix $Qx_{i,k}^\theta = \begin{bmatrix} Q_{i,k}^{\theta(1)} & 0 \\ 0 & Q_{i,k}^{\theta(2)} \end{bmatrix}$.

Equation (8.1) represents an M number of possible models for each node. Each model differs from the others in the size of the jumps u_i^θ. The resultant state vector estimate for node i at time instant k, $\hat{X}_{i,k|k}$, would be a weighted combination of the estimates of each model $\hat{X}_{i,k|k}^\theta$. The resultant state vector estimate for each node $\hat{X}_{i,k|k}$ is found as was shown in chapter 5 by:

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

where $\mu^{\theta}_{i,k|k}$ is the model probability matrix. It is the probability that the estimated state vector $\hat{X}_{i,k|k}$ follows the model vector $\hat{X}^{\theta}_{i,k|k}$ given the measured values until the time step k.

The value of the corrected measurement $x_{i,k}$ is never sensed or measured. What is really measured is $r_{i,k}$, i.e., the reading of the sensor. As we argued earlier, $r_{i,k}$ deviates from $x_{i,k}$ by both systematic and random errors. The random error is taken to be a Gaussian noise $w_{i,k} \sim N(0, R_{i,k})$ with zero mean and variance $R_{i,k}$ (measurement noise variance). The systematic error is referred to as the drift $d_{i,k}$. This leads to (8.2).

$$y^{(1)}_{i,k} = r_{i,k} = x_{i,k} + d_{i,k} + w_{i,k} \quad w_{i,k} \sim N(0, R_{i,k}) \tag{8.2}$$

We also define $y^{(2)}_{i,k}$ as the difference between the measurement $r_{i,k}$ and the SVR modelled value $\widetilde{x}_{i,k}$ and refere to $y^{(2)}_{i,k}$ as the drift measurement of node i at time instant k.

$$\begin{aligned} y^{(2)}_{i,k} &= y^{(1)}_{i,k} - f(\{x_{j,k-1}\}^{n}_{j=1,j\neq i}) \\ &= x_{i,k} + d_{i,k} + w_{i,k} - f(\{x_{j,k-1}\}^{n}_{j=1,j\neq i}) \\ &= x_{i,k} + d_{i,k} + w_{i,k} - \widetilde{x}_{i,k} \quad w_{i,k} \sim N(0, R_{i,k}) \end{aligned} \tag{8.3}$$

This can be expressed in vector notation by:

$$Y_{i,k} = \begin{bmatrix} y^{(1)}_{i,k} \\ y^{(2)}_{i,k} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_{i,k} \\ d_{i,k} \end{bmatrix} + \begin{bmatrix} w_{i,k} \\ w_{i,k} \end{bmatrix} - \begin{bmatrix} 0 \\ \widetilde{x}_{i,k} \end{bmatrix} \tag{8.4}$$

The noise component associated with $Y_{i,k}$ has a mean vector $\mu_{Y_{i,k}} = [0 \quad 0]^T$ and covariance matrix $Ry_{i,k} = \begin{bmatrix} R_{i,k} & R_{i,k} \\ R_{i,k} & R_{i,k} \end{bmatrix}$ which indicates that it is not Gaussian. The system is clearly observable when $\widetilde{x}_{i,k} = x_{i,k}$, i.e. when $\widetilde{x}_{i,k}$ is a true, bias free, representation of $x_{i,k}$ and the differnce between $x_{i,k}$ and $\widetilde{x}_{i,k}$ is zero.

Since the state equation and the measurement equation are both linear, the IMM model discussed in chapter 5 can be used to estimate $X_{i,k}$. However, KFs are replaced by UKFs to deal with the non Gaussian noise associated with $Y_{i,k}$.

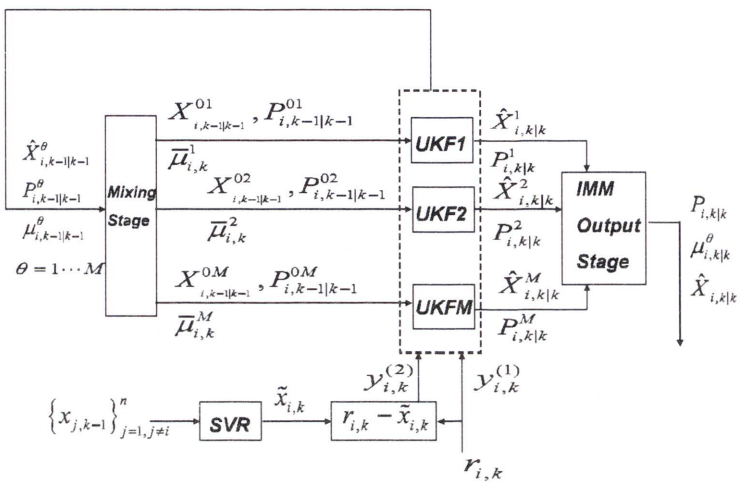


Figure 8.1: Measurement correction framework at node i for fast changing readings, $\overline{\mu}^{\theta}_{i,k} = \mu^{\alpha|\theta}_{i,k-1|k}$.

Figure(8.1) shows a block diagram of our drift correction algorithm. It clearly summarises the stages of the IMM-SVR-UKF framework in one of the nodes in the cluster. The steps of the algorithm are stated below:

Decentralised measurement correction algorithm using the IMM-SVR-UKF framework

At step k

- Each node i finds its predicted corrected measurement $\tilde{x}_{i,k} = f(\{x_{j,k-1}\}^{n}_{j=1,j\neq i})$.
- The prior model probabilities $\mu^{\theta}_{i,k-1|k-1}$ are available.
- Each node i obtains its reading $r_{i,k}$
- Mixing stage

$$\mu^{\alpha|\theta}_{i,k-1|k} = \frac{\Gamma_{\alpha\theta}\,\mu_{i,k-1|k-1}(\alpha)}{\sum_{\beta=1}^{M}\Gamma_{\beta\theta}\,\mu_{i,k-1|k-1}(\beta)}$$

$$\hat{X}^{0\theta}_{i,k-1|k-1} = \sum_{\alpha=1}^{M}\hat{X}^{\alpha}_{i,k-1|k-1}\,\mu^{\alpha|\theta}_{i,k-1|k}$$

$$P^{0\theta}_{i,k-1|k-1} = \sum_{\alpha=1}^{M}\mu^{\alpha|\theta}_{i,k-1|k}\{P^{\alpha}_{i,k-1|k-1} + [\hat{X}^{\alpha}_{i,k-1|k-1} - \hat{X}^{0\theta}_{i,k-1|k-1}][\hat{X}^{\alpha}_{i,k-1|k-1} - \hat{X}^{0\theta}_{i,k-1|k-1}]^{T}\}$$

- Unscented Kalman Filter update stage

1. for each model θ the sigma points $\sigma_i = \{\mathcal{W}_i, \mathcal{X}_i\}$ are found from $\hat{X}^{0\theta}_{i,k-1|k-1}$ using equations (7.7).
2. For each model θ, the sigma points are propagated through the function $g(.)$ using equation (7.8).
3. The UKF finds model estimates $\hat{X}^{\theta}_{i,k|k}$ and $P^{\theta}_{i,k|k}$ using equations (7.9)-(7.17).

- IMM output stage

  The Model probabilities are updated as follows:

$$\mu^{\theta}_{i,k|k} = \frac{\lambda_{i,k}(\theta) \sum_{\alpha=1}^{M} \Gamma_{\alpha\theta}\, \mu_{i,k-1|k-1}(\alpha)}{\sum_{\beta=1}^{M} \lambda_{i,k}(\beta) \sum_{\phi=1}^{M} \Gamma_{\phi\beta}\, \mu_{i,k-1|k-1}(\phi)}$$

  where $\lambda_{i,k}(\theta) = N(H\hat{X}^{\theta}_{i,k|k-1}, HP^{\theta}_{i,k|k-1}H^T + Ry_{i,k})$.

  The state estimate and its associated covariance are updated as follows:

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

$$P_{i,k|k} = \sum_{\theta=1}^{M} \mu^{\theta}_{i,k|k} \{P^{\theta}_{i,k|k} + [\hat{X}^{\theta}_{i,k|k} - \hat{X}_{i,k|k}][\hat{X}^{\theta}_{i,k|k} - \hat{X}_{i,k|k}]^T\}$$

- The estimated corrected measurement $x_{i,k|k}$ is sent to the neighbouring nodes.
- The algorithm reiterates.

8.2 Evaluation

Our aim is to evaluate the ability of our proposed framework to correct the drift experienced in sensor nodes and to extend the functional life of the sensor network. The data in our evaluation are a set of real sensor measurements gathered from a deployment of wireless sensors in the IBRL [97].

In our evaluations, we use the same network (case2), the same training set, the same SVR parameters (case2) and the same UKF parameters as the ones used in the previous chapter. Having used the same data, we compare the performance of the IMM-SVR-UKF framework with the SVR-UKF framework in terms of the

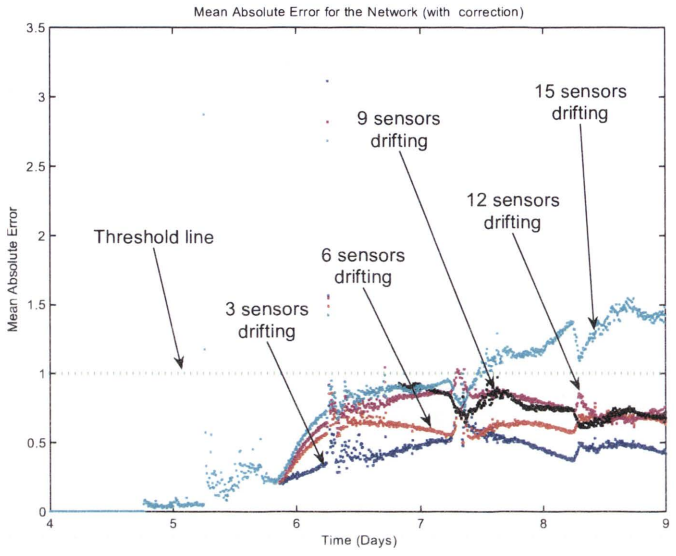


Figure 8.2: Mean Absolute Error for the network with correction for 2001 samples in 10 days using 11 levels IMM.

network's mean absolute error for several drift scenarios. In addition to that, we show the effect of the number of the IMM modes on the overall performance of the algorithm.

Figure 8.2 shows the mean absolute error for the network when using an 11 mode ($M = 11$) IMM with the SVR-UKF framework for 2001 samples in 10 days. We refer to the resulting combination as IMM-SVR-UKF framework.

By comparing figure 8.2 with figure 7.4 (both of them are for 2001 samples), it can be noticed that the IMM-SVR-UKF framework (see figure 8.2), as opposed to the SVR-UKF (see figure 7.4), results in smoother curves and stable mean absolute errors for the network after the 6^{th} day, for all of the 5 scenarios. Similarly, the results of using the IMM-SVR-UKF in figure 8.2 are relatively smoother and more stable than the results of applying the plain SVR-UKF framework on 4001 samples (see figure 7.5). It is obvious that for half the sampling rate, the IMM-SVR-UKF framework gives us comparable results. This means that the communication overhead is reduced by half and so is the energy consumed in communication among the sensors. However, the improved performance when using IMM is at the cost of

Table 8.1: Processing times required by SVR-UKF based and IMM-SVR-UKF based error correction algorithms.

Algorithm	Processing time/iteration (PT)	$Ratio = \frac{PT(Any)}{PT(UKF)}$
SVR-UKF	2.82 ms	1
IMM-SVR-UKF ($M = 3$)	17.95 ms	6.36
IMM-SVR-UKF ($M = 7$)	40.36 ms	14.31
IMM-SVR-UKF ($M = 11$)	62.74 ms	22.24

the increased computational complexity. We use the processing time required by each algorithm as an indication of its computational complexity. Table 8.1 shows the average processing time required by the SVR-UKF framework and the IMM-SVR-UKF one (for different number of models) as reported by our MatLab simulations. The *Ratio* column clearly shows that the IMM-SVR-UKF framework requires approximately $2M$ the time required by the SVR-UKF framework. Obviously, the computational complexity can be reduced by reducing the number of models M used in the IMM algorithm.

The results of the mean absolute error for the network, when using the IMM-SVR-UKF framework for $M = 7, 5, 3$, are shown in figures 8.3, 8.4 and 8.5, respectively. From these figures, It is clear, that using more models for IMM results in a relatively smoother response. Nevertheless; the differences are very small and still for $M = 3$ the system performs better than the plain SVR-UKF framework in terms of smoothness and stability of the absolute error curves for both cases of 2001 samples (figure 7.4), and of 4001 samples (figure 7.5).

As mentioned earlier, the sampling rate of the IBRL data was 2 samples /min. However, some data were missing. The average number of samples in the group of sensors we are dealing with is 17500 samples in the 10 days period. By using IMM, we reduced the sampling rate in the 10 days period from 17500 to 2001 and got very stable response with acceptable error level for our application (within the $1C^o$ threshold). Comparing the SVR-UKF framework with the IMM-SVR-UKF framework, we notice that the latter gives better results with half the sampling rate i.e. half the communication energy is saved. This results in longer battery life. Even that the complexity of the IMM-SVR-UKF is higher, which means higher energy

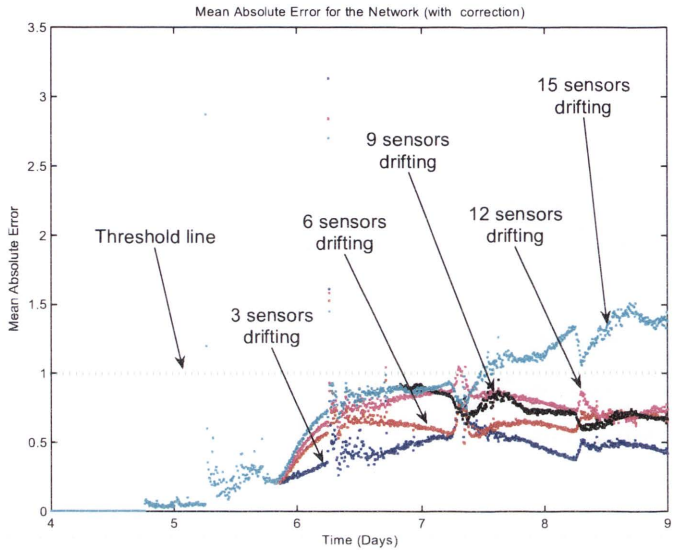


Figure 8.3: Mean Absolute Error for the network with correction for 2001 samples in 10 days using 7 levels IMM.

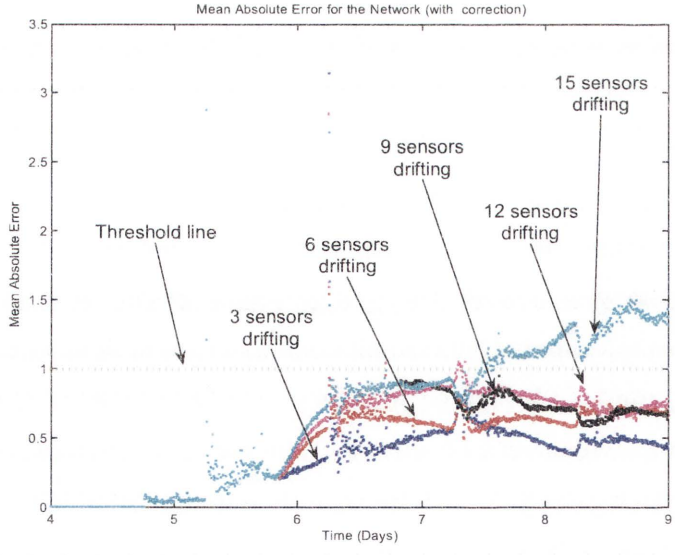


Figure 8.4: Mean Absolute Error for the network with correction for 2001 samples in 10 days using 5 levels IMM.

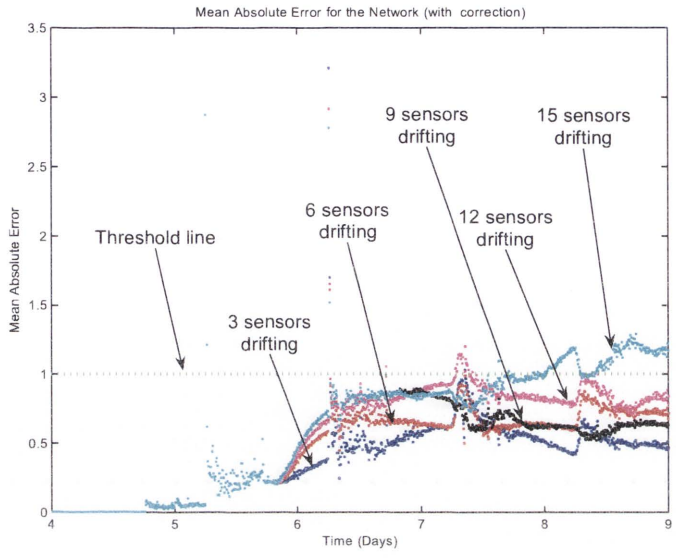


Figure 8.5: Mean Absolute Error for the network with correction for 2001 samples in 10 days using 3 levels IMM.

consumed in sensors calculations per iteration, the power consumed in a sensor for signal processing is usually small compared to the power consumed in communication as seen in table 8.2 quoted from [108]. Another example from WSN literature supporting that is given in [109]. It is shown there, that the ratio between communication and computation energy consumption ranges from 10^3 to 10^4 for sensors like Sensoria sensors and Berkeley motes. In conclusion, it can be said that using IMM is expected to preserve energy, especially, when dealing with low M as in figure 8.5.

Table 8.2: Energy consumed for each sensor action, based on measurements of the Mica2 sensor node qouted from [108].

Action	Energy consumed
sample (single sensor)	1.637×10^{-6} J
send (single message)	1.653×10^{-3} J
listen (for 1 sec)	23.88×10^{-3} J
sleep (for 1 sec)	90×10^{-6} J
aggregate (compute max of array)	1.637×10^{-6} J

8.3 Conclusion

In this chapter we have proposed the use of the IMM algorithm with the SVR-UKF framework presented in the previous chapter, to overcome the error in the measurement estimation caused by the sudden steep changes in the measured data. We refer to the resulting framework as IMM-SVR-UKF framework. The IMM-SVR-UKF framework serves as an alternative for increasing the sampling rate of the data in order to reduce the effect of the jumps on the accuracy of the estimated readings.

Besides its ability to follow (track) data that suffer from sharp changes and sudden jumps, the IMM-SVR-UKF framework can deal with jumps in the readings caused by lowering the sampling rate. Consequently, this allows reducing the communication overhead among sensors to maintain the calibration, and eventually reduces the energy consumed from the batteries.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

IN this thesis, we have addressed the problem of random errors and systematic errors (drifts and bias) in sensor measurements and explained their bad effects on the inferences made by the sensor networks. We have argued that the harsh conditions of deployment of the sensors in the network impose finding automatic procedures for continuously calibrating these sensors as manual calibration is infeasible. We have surveyed several solutions for error detection / error detection and correction from WSN literature. Most of these solutions are application specific and assume the linearity of the faults to simplify the problem. They do not explicitly address the problems of random and systematic errors in sensor measurements. Alternatively, we use statistical modelling rather than physical modelling to model the spatio-temporal cross correlations among sensors' measurements. This makes the frameworks presented in this thesis applicable to most sensing problems with minor changes. In addition to that, the solutions presented here account for both the systematic and the random errors in sensor measurements and do not assume linearity of the faults.

The solutions presented here rely on the fact that a physical phenomenon in a certain area follows some spatio-temporal correlation. According to this, we assume that the sensors readings in that area are correlated. We also assume that measurement errors due to faulty equipment are likely to be uncorrelated. Based on these assumptions, and inspired by the resemblance of the registration problem

in radar target tracking with the bias error problem in WSN, we follow a Bayesian framework to estimate the correct sensor measurements, together with the drifts and biases they suffer from. We present several methods for solving the drift problem in WSNs. These methods can be categorised into 2 categories according to the density of deployment of sensors in the WSN.

In the case of densely deployed WSN, the sensors in the neighbourhood are assumed to be close to each other that they observe the same phenomenon and ideally read similar measurements. Hence, the average of their corrected readings is taken as a basis for each sensor to self-assess its measurement and to estimate its drift using a KF. The KF output is then used to correct the sensor reading, and as a feedback to itself to estimate the next time step drift. The solution is computationally simple using an averaged sensing value and a single state KF iterative procedure, allowing its implementation in a WSN. The above mentioned solution is designed to detect and correct smooth drifts. In the case of drifts that have sudden jumps or surges, the KF is replaced by an IMM based filter since KF does not respond efficiently to changes in the dynamics as the drift changes abruptly at some points. The IMM based algorithm performs better (in our context) for both the smooth and unsmooth drift cases, however, at the cost of the increased computational complexity. Nevertheless, the IMM solution is still considered in this context, as computationally simple since it deals with one element state and measurement vectors and uses the average of the neighbourhood. Both solutions are completely decentralised and also scalable. Any new node joining the neighbourhood needs only to obtain information from its neighbouring sensors and then find the average and apply the KF or the IMM iterative procedures. Similarly, any sensor excluded from the neighbourhood will not affect the others as they use the average.

On the other hand, when the sensors are not densely deployed, the average cannot be used by the sensors to self-assess their readings. In this case each sensor is required to find a way to express it's reading in terms of its neighbour's corrected measurements. This is done using SVR. We use SVR to model the interrelationships of sensor measurements in a neighbourhood. This enables us to incorporate

the spatio-temporal correlation of neighbouring sensors, in order to predict future measurements. In Chapter 6, the SVR predicted value is used by a KF to estimate the actual drift in the measured value (here temperature) at the sensor under consideration, so that it can be used to correct the reading of the sensor. We have noticed that although the KF algorithm estimates the drift and corrects the reading of a node, it introduces some system errors that accumulate with time. We refer that error to the use of KF filter with the SVR nonlinear systems. A solution for this problem is given in chapter 7.

In chapter 7, the KFs are replaced by UKFs since they are specially designed to deal with nonlinear systems. Particle filters have not been used due to their high computational complexity which is undesirable in WSNs applications. The use of UKF reduces the system error noticed in the SVR-KF evaluations, since it is a better method for estimating the mean and the variance of a random variable propagating through nonlinear systems (here SVRs). The estimated variable here is the corrected measurement (temperature) as opposed to drift in chapter 6. A major source of error in the SVR-UKF temperature estimation framework (though small) is the sudden steep change in the measured variable. This is dealt with in chapter 7 by increasing the sampling rate of the data in a trade-off with the communication overhead. An alternative solution to increasing the sampling rate is given in chapter 8.

In chapter 8, we have proposed the use of IMM with the SVR-UKF framework to overcome the error in our temperature estimation caused by the sudden steep changes in the measured data. Besides its ability to follow (track) data that suffer from sharp changes and sudden jumps, the IMM-SVR-UKF framework can deal with jumps in the readings caused by lowering the sampling rate. Consequently, this results in less communication overhead among sensors to maintain the calibration, and eventually reduces the energy consumed from the batteries.

The solutions presented for the non-densely deployed sensor networks are designed for smooth drifts. Similar to the solutions presented in the dense deployment case, the solutions are recursive and totally decentralised. The use of UKF

and IMM considerably increases the computational complexity of the measurement correcting system. Although using IMM in chapter 8 increases the computational complexity per iteration of the SVR-UKF algorithm and therefore increases the energy per iteration consumed by each sensor for computation, it reduces the communication overhead and therefore reduces the communication energy. The communication energy is the main component of the total energy consumed by the sensor as was discussed in chapter 8. The computational complexity for modelling data using SVR is mainly contributed by the training phase. The training of SVR is usually done once or at low frequencies during the life time of the sensor network, unless the network is deployed in a highly dynamic environment. In the case of IBRL data, the environment is controlled. Hence, the SVR can be trained off-line or at a high performance node which is resourceful. The trained SVR can then be uploaded to each sensor for use in the running phase. It is important to note that only the support vectors are required to be uploaded to the sensors as opposed to the whole training data set. This way, the memory overhead in the sensors during the running phase is reduced.

Extensive evaluations of the presented algorithms for both simulated and real data proved that they are effective in detecting and correcting sensor errors. A comparison between the performance of the sensor network with and without applying the error detection and correction algorithms clearly showed that implementing these algorithms extends the useful life time of the network. A study of the limits of proposed techniques as a function of number of sensors deployed, the amount of computing and communication required was made. The processing time required by each algorithm was used as an indicator of its computational complexity. The evaluation results under dense deployment conditions using simulated data (chapters 4, 5) showed that the average error in sensor readings was reduced by 95% when 20% of the sensor nodes were drifting as opposed to 81% when 70% of the sensors were drifting indicating that performance of the error correcting algorithms degrades as the number of sensors with biases increases. It was also shown in chapter 3 that the performance of the error correcting algorithms is dependent

on the number of sensors in the neighbourhood and their ability to communicate with each other.

The algorithms addressing the practical application of sparsely deployed WSN in an office area (chapters 6, 7, 8) were evaluated on real data obtained from a sensor network deployed in IBRL. The evaluations clearly showed that the use of SVR to model the spatio-temporal correlations among neighbour sensor measurements and predict the future sensor measurements, led to correcting biased sensor measurements and extended functional life time of the network. The functional life time of the network was extended from six days when no drift correction algorithms were implemented to more than nine days with the error detection and correction algorithms implemented. The effect of changing the data sampling rate was studied showing better error correction results and more stable response for higher sampling rates at the cost of increased communication energy expenditure. The performance under low sampling rate was improved using IMM to result in lower communication energy consumption and extended battery life.

9.2 Future Research Directions

In future, we intend to implement an incremental SVR framework to periodically re-train the SVR, in order to adapt to any phenomenal changes that may occur in the network. This way, the WSN will continue to able to automatically calibrate its sensors, even if the environment they are working in has changed from the environment they were trained in. So if the heating and cooling pattern changes in the IBRL, the sensors will still continue to detect and correct their drifts. Such phenomenal changes are expected to be encountered more often in an uncontrollable environment such as outdoor environment. Therefore, a future step will be implementing and testing this solution in a WSN deployed in an outdoor environment.

We also intend to implement and test the validity of a hybrid model that uses both the average and the SVR for detecting and correcting drifts and biases in the measurements of the nodes. A scenario where the hybrid model is expected to be

applicable is in a sensor network comprising large number of nodes and deployed in an area that shows spatial variation in the phenomenon under consideration. Sensors that are close to each other form clusters. The sensors in each cluster run an algorithm such as the ones in chapters 4, 5 for detecting and correcting drifts using the average. In order to make use of the spatio-temporal correlations among the clusters, each cluster head runs an SVR based drift detection and correction algorithm. This way, the network will automatically detect that a cluster, with too many drifting sensors, is reporting erroneous data. The operator of the network can then deploy new sensors in the area covered by that cluster, and the algorithm will work without the need for re-initialisation. In event that a cluster head develops a considerable drift to the extent that it should be removed from the network, it can send its support vectors to one of its properly operating neighbours or to a newly deployed sensor to become the new cluster head and continue running the SVR based drift correction algorithm with the other cluster heads. This makes the hybrid model scalable.

We are also planning to come up with a combined model addressing both the drift problem and the trust management problem in WSNs. The trust in the data reported by each sensor will depend on the reliability of the link with that sensor (communication trust) and on how correct the sensor's reported data are (data trust). The data are considered trusted as long as the drift correction is working properly and the drift does not exceed a certain value. However, the data trust value will also depend on the number of the sensors that are drifting in the neighbourhood, since as showed in this thesis; the number of drifting sensor affects the performance of the drift correction process. The combined trust, which depends on both the data and communication trust, will determine whether to keep the sensor or label it as untrustworthy and remove it from the network. Finally, we intend to implement our designed models in a live sensor network and, if possible, commercialisation of the models.

Bibliography

[1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," *Int. Conference on Acoustics, Speech, and Signal Processing)*, May 2001.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Comp. Networks*, vol. 38, pp. 393–422, 2002.

[3] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak, "A collaborative approach to in-place sensor calibration," *Int. Workshop on Information Processing in Sensor Networks*, pp. 301–316, 2003.

[4] M. Takruri and S. Challa, "Drift aware wireless sensor networks," in *Proc. of the 10th intl. conference on information fusion*, Quebec City, Canada, July 2007.

[5] B. Krishnamachari and S. Iyengar, "Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Tran. Computers*, vol. 53, no. 3, pp. 241–250, 2004.

[6] A. Bharathidasan and V. Ponduru, "Sensor networks: An overview," Tech. Rep., 2002.

[7] M. Tubaishat and S. Madria, "Sensor networks: an overview," *Potentials, IEEE*, vol. 22, no. 2, pp. 20–23, 2003.

[8] M. Momani, "Bayesian methods for modelling and management of trust in wireless sensor networks," Ph.D. dissertation, University of Technology, Sydney, Sydney, Australia, July 2008.

[9] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 102–114, 2002.

[10] S. Iyengar and R. Brooks, *Distributed Sensor Networks.* CRC Press, 2005.

[11] J. Rabaey, M. Ammer, J. da Silva Jr., D. Patel, and S. Roundy, "Picoradio supports ad hoc ultra-low power wireless networking," *IEEE Computer Magazine*, vol. 33, no. 7, pp. 42–48, 2000.

[12] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, pp. 18–25, 2006.

[13] G. Werner-Allen, J. Johnson, M. Ruiz, and J. L. M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network," in *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN05)*, 2005.

[14] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *IEEE Computer Journal*, vol. 37, pp. 41–49, 2004.

[15] S. D. Glaser, "Some real-world applications of wireless sensor nodes," in *Proceedings of the SPIE Symposium on Smart Structures and Materials NDE 2004*, 2004.

[16] J. Paek, O. Gnawali, K.-Y. Jang, D. Nishimura, R. Govindan, J. Caffrey, M. Wahbeh, and S. Masri, "A programmable wireless sensing system for structural monitoring," in *Proceedings The 4th World Conference on Structural Control and Monitoring (4WCSCM)*, 2006.

[17] T. Gao, D. Greenspan, M. Welsh, R. Juang, and A. Alm, "Vital signs monitoring and patient tracking over a wireless network," in *27th Annual International Conference of the IEEE EMBS05*, Shanghai, September 2005, pp. 102–105.

[18] E. Petriu, N. Georganas, D. Petriu, D. Makrakis, and V. Groza, "Sensor-based information appliances," *IEEE Instrumentation and Measurement Magazine*, vol. 3, pp. 31–35, 2000.

[19] J. Liu, J. Reich, and F. Zhao, "Collaborative in-network processing for target tracking," *Journal on Applied Signal Processing*, vol. 4, pp. 378–391, 2003.

[20] G. Pottie and W. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[21] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*. ACM, 2001, pp. 272–287.

[22] H. Zhang and A. Arora, "Gs3: Scalable self-configuration and self-healing in wireless sensor networks," *Computer Networks*, vol. 43, no. 4, pp. 459–480, 2003.

[23] V. B. Jr and J. Mallett, "Collaborative knowledge building by smart sensors," *BT Technology Journal*, vol. 22, no. 4, pp. 45–51, 2004.

[24] C. Chiasserini and R. Rao, "On the concept of distributed digital signal processing in wireless sensor networks," in *in Proceedings of MILCOM2000*, Oct. 2002.

[25] S. Nath, Y. Ke, P. Gibbons, B. Karp, and S. Seshan, "A distributed filtering architecture for multimedia sensors," in *In First Workshop on Broadband Advanced Sensor Networks (BaseNets)*, 2004.

[26] H. Qi, S. Iyengar, and K. Chakrabarty, "Distributed sensor networks - a review of recent research," *Journal of the Franklin Institute*, vol. 338, pp. 655–668, 2001.

[27] A. Chandrakasan, R. Amirtharajah, S. Cho, J. Goodman, G. Konduri, J. Kulik, and W. Rabiner, "Design considerations for distributed micro-sensor systems," in *Proceedings of the IEEE 1999 Custom Integrated Circuits Conference*, 1999, pp. 279–286.

[28] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 1999, pp. 263–270.

[29] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," in *Proc. Int'l J. High Performance Computing Applications*, 2002.

[30] P. Bonnet, J. Gehrke, and P. Seshadri, "Querying the physical world," *IEEE Personal Communications*, vol. 7, no. 5, pp. 10–15, 2000.

[31] Y. Weng, S. Kallakuri, X. Liang, A. Doboli, S. Hong, T. Robertazzi, and S. Doboli, "Dynamic architecture adaptation to improve scalability of sensor networks: A case study for a smart sensor for face recognition," in *Proceedings of the Real-Time Systems Symposium (RTSS'2004)*, Lisbon, Portugal, 2004.

[32] P. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan, "Irisnet: An architecture for a world-wide sensor web," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 22–33, 2003.

[33] S. Nath, Y. Ke, P. Gibbons, B. Karp, and S. Seshan, "Irisnet: An architecture for enabling sensor-enriched internet services," Tech. Rep., June 2003.

[34] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, "On-line fault detection of sensor measurements," *IEEE Sensors*, pp. 974–980, 2003.

[35] L. Balzano, "Addressing fault and calibration in wireless sensor networks," Master's thesis, University of California, Los Angeles, California, 2007.

[36] T. Islam and H. Saha, "Study of long-term drift of a porous silicon humidity sensor and its compensation using ann technique," *Sensors and Actuators A: Physical*, vol. 133, no. 2, pp. 472–479, 2007.

[37] S. Marco, A. Ortega, A. Pardo, and J. Samitier, "Gas identification with tin oxide sensor array and self-organizing maps: Adaptive correction of sensor drifts," *IEEE Transactions on Instrumentation and Measurement*, vol. 47, no. 1, pp. 316–321, 1998.

[38] S. Brahim-Belhouari, A. Bermak, M. Shi, and P. Chan, "Fast and robust gas identification system using an integrated gas sensor technology and gaussian mixture models," *IEEE Sensors Journal*, vol. 5, no. 6, pp. 1433–1444, 2005.

[39] X. Wang and M. Ye, "Hysteresis and nonlinearity compensation of relative humidity sensor using support vector machines," *Sensors and Actuators B: Chemical*, vol. 129, no. 1, pp. 274–284, 2008.

[40] S. Chessa and P. Santi, "Crash faults identification in wireless sensor networks," *Computer Communications*, vol. 25, no. 14, pp. 1273–1282, 2002.

[41] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Anomaly detection in wireless sensor networks," *IEEE Wireless Communication*, vol. 15, no. 4, pp. 34–40, 2008.

[42] C. Loo, M. Ng, C. Leckie, and M. Palaniswami, "Anomaly detection in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2, no. 4, pp. 313–332, 2006.

[43] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek, "Distributed anomaly detection in wireless sensor networks," in *Proceedings of the 10th IEEE International Conference Communication Systems*, Singapore, October 2006.

[44] ——, "Quarter sphere based distributed anomaly detection in wireless sensor networks," in *Proceedings of the IEEE International Conference Communications (IEEE ICC '07)*, UK, June 2007.

[45] D. Liu, P. Ning, and W. Du, "Detecting malicious beacon nodes for secure location discovery in wireless sensor networks," in *in The 25th IEEE International Conference on Distributed Computing Systems (ICDCS '05)*, 2005.

[46] A. Srinivasan, J. Teitelbaum, and J. Wu, "Drbts: Distributed reputation based beacon trust system," in *in the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC '06)*, 2006.

[47] M. Momani and S. Challa, "Gtrssn: Gaussian trust and reputation system for sensor networks," in *in International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE'07)*, University of Bridgeport, 2007.

[48] M. Momani, S. Challa, and R. Alhmouz, "Can we trust trusted nodes in wireless sensor networks?" in *in the International Conference on Computer and Communication Engineering (ICCCE '08)*, Kuala Lumpur,Malaysia, 2008.

[49] N. Okello and G. Pulford, "Simultaneous registration and tracking for multiple radars with cluttered measurements," *IEEE Signal Processing Workshop on Statistical Signal and Array Processing*, pp. 60–63, June 1996.

[50] N. Okello and S. Challa, "Simultaneous registration and track fusion for networked trackers," in *conference on information fusion*, Montral, Canada, August 2003.

[51] L. Brown, "A survey of image registration techniques," *ACM Computing Surveys*, vol. 24, no. 4, pp. 326–376, December 1992.

[52] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 1, pp. 977–1000, June 2003.

[53] B. Hoadley, "A baysian look at inverse linear regression," *J. of the American Stats. Association*, vol. 65, no. 329, pp. 356–369, March 1970.

[54] L. Balzano and R. Nowak, "Blind calibration of sensor networks," *Information Processing in Sensor Networks*, April 2007.

[55] V. Bychkovskiy, "Distributed in-place calibration in sensor networks," Master's thesis, University of California, Los Angeles, California, 2003.

[56] L. Balzano and R. Nowak, "Blind calibration of networks of sensors: Theory and algorithms," in *Networked Sensing Information and Control*. Springer US, 2008, pp. 9–37.

[57] J. Feng, S. Megerian, and M. Potkonjak, "Model-based calibration for sensor networks," *Sensors*, pp. 737 - 742, October 2003.

[58] K. Whitehouse and D. Culler, "Calibration as parameter estimation in sensor networks," in *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, 2002.

[59] ——, "Macro-calibration in sensor/actuator networks," *Mobile Networks and Applications Journal (MONET), Special Issue on Wireless Sensor Networks*, June 2003.

[60] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, "Simultaneous localization, calibration, and tracking in an ad hoc sensor network," in *In Proceedings of 5th International Conference on Information Processing in Sensor Networks (IPSN06)*, 2006, pp. 27–33.

[61] A. Ihler, J. Fisher, R. Moses, and A. Willsky, "Nonparametric belief propagation for self-calibration in sensor networks," in *In Proceedings of the Third international Symposium on Information Processing in Sensor Networks*, 2004.

[62] E. Elnahrawy and B. Nath, "Cleaning and querying noisy sensors," in *in Proceedings of ACM WSNA03*, 2003.

[63] B. Sallans, D. Bruckner, and G. Russ, "Statistical model-based sensor diagnostic for automation systems," in *Fieldbus systems and their applications*, M. L. Chavez, Ed. Elsevier, 2005, pp. 239–246.

[64] X. Xu, J. W. Hines, and R. E. Uhrig, "On-line sensor calibration monitoring and fault detection for chemical processes," in *Maintenance and Reliability Conference (MARCON 98)*, 1998, pp. 12–14.

[65] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: an efficient framework for modeling sensor network data," in *In IPSN04*, 2004, pp. 1–10.

[66] R. Nowak and U. Mitra, "Boundary estimation in sensor networks: Theory and methods," in *In IPSN*, 2003, pp. 80–95.

[67] Y. M. Wang, R. T. Schultz, R. T. Constable, and L. H. Staib1, "Nonlinear estimation and modeling of fmri data using spatio-temporal support vector regression," *Information Processing in Medical Imaging*, vol. 2732, pp. 647–659, 2003.

[68] M. K. Gill, T. Asefa, M. W. Kemblowski, and M. McKee, "Soil moisture prediction using support vector machines1," *J. of the American Water Resources Association*, vol. 42, no. 4, pp. 1033–1046, 2006. [Online]. Available: http://www.blackwell-synergy.com/doi/abs/10.1111/j.1752-1688.2006.tb04512.x

[69] M. K. Gill, M. W. Kemblowski, and M. McKee, "Soil moisture data assimilation using support vector machines and ensemble kalman filter," *J. of the American Water Resources Association*, vol. 43, no. 4, pp. 1004–1015, 2007.

[70] M. Takruri, K. Aboura, and S. Challa, "Distributed recursive algorithm for auto calibration in drift aware wireless sensor networks," in *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering*, K. Elleithy, Ed. Springer, 2008, pp. 21–25.

[71] M. Takruri, S. Challa, and R. Chacravorty, "Recursive bayesian approaches for auto calibration in drift aware wireless sensor networks," *to appear in the Journal of Networks*, 2010.

[72] M. Takruri, S. Challa, and R. Chakravorty, "Auto calibration in drift aware wireless sensor networks using the interacting multiple model algorithm," in *Mosharaka International Conference on Communications, Computers and Applications (MIC-CCA 2008)*, Amman, Jordan, August 2008.

[73] M. Takruri, S. Rajasegarar, S. Challa, C. Leckie, and M. Palaniswami, "Online drift correction in wireless sensor networks using spatio-temporal modeling," in *Intl. conference on information fusion*, Cologne, Germany, July 2008.

[74] ——, "Spatio-temporal modelling based drift aware wireless sensor networks," *To appear in the International Journal of Distributed Sensor Networks*, 2010.

[75] S. Challa, R. Evans, M. Morelande, and D. Musicki, *Fundamentals of Object Tracking*. Cambridge University Press, 2008.

[76] Y. Bar-Shalom, *Estimation and Tracking : Principles, Techniques and Software*. Boston Artech House, 1993.

[77] D. Koks and S. Challa, "An introduction to bayesian and dempster-shafer data fusion," Tech. Rep., 2005.

[78] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Tran. ASME J. Basic Eng.*, pp. 35–45, March 1960.

[79] G. Welch and G. Bishop, "An introduction to the kalman filter," 1995.

[80] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: a survey," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 34, no. 1, pp. 103–123, 1998.

[81] L. Johnston and V. Krishnamurthy, "An improvement to the interacting multiple model (imm) algorithm," *IEEE Tranactions on Signal Processing*, vol. 49, no. 12, pp. 2909–2923, 2001.

[82] H. Blom, "An efficient filter for abruptly changing systems," *Proceedings of the 23rd IEEE Conference on Decision and Control*, pp. 656–658, Dec. 1984.

[83] Y. Bar-Shalom, S. Challa, and H. Blom, "Imm estimator versus optimal estimator for hybrid systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 3, pp. 986–991, 2005.

[84] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *IEEE Tran. on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.

[85] X. Wang, S. Challa, R. Evans, and X. R. Li, "Minimal submodel-set algorithm for maneuvering target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1218–1231, 2003.

[86] X. R. Li, "Engineer's guide to variable-structure multiple-model estimation for tracking," in *Multitarget-Multisensor Tracking: Applications and Advances*, Y. Bar-Shalom and W. Blair, Eds. Boston, MA: Artech House, 2000, vol. 3, ch. 10, pp. 499–567.

[87] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[88] S. M. Clarke, J. H. Griebsch, and T. W. Simpson, "Analysis of support vector regression for approximation of complex engineering analyses," *J. of Mechanical Design*, vol. 127, no. 6, pp. 1077–1087, 2005.

[89] V. N. Vapnik, *Statistical Learning Theory*. John Wiley & Sons, 1998.

[90] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[91] B. Scholkopf and A. Smola, *Learning with Kernels*. MIT Press, 2002.

[92] K.-R. Muller, A. J. Smola, G. Rätsch, B. Schökopf, J. Kohlmorgen, and V. Vapnik, "Using support vector machines for time series prediction," pp. 243–253, 1999.

[93] S. G. Nash and A. Sofer, *Linear and nonlinear programming*. McGraw-Hill, 1996.

[94] O. L. Mangasarian and D. R. Musicant, "Large scale kernel regression via linear programming," *Mach. Learn.*, vol. 46, no. 1-3, pp. 255–269, 2002.

[95] A. Smola, B. Scholkopf, and G. Ratsch, "Linear programs for automatic accuracy control in regression," in *Proc. of Int. Conf. on Artificial Neural Networks*, 1999.

[96] M. Epelman and R. M. Freund, "A new condition measure, preconditioners, and relations between different measures of conditioning for conic linear systems," *SIAM J. on Optimisation.*, vol. 12, no. 3, pp. 627–655, 2002.

[97] 2006, "http://db.lcs.mit.edu/labdata/labdata.html," in *[online]*, Accessed on 07/09/2006. [Online]. Available: http://db.lcs.mit.edu/labdata/labdata.html

[98] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy, "Svm and kernel methods matlab toolbox," Perception Systmes et Information, INSA de Rouen, Rouen, France, 2005. [Online]. Available: http://asi.insa-rouen.fr/enseignants/~arakotom/toolbox/index.html

[99] S. Lu, L. Cai, D. Lu, and J. Chen, "Two efficient implementation forms of unscented kalman filter," *IEEE Int. Conference on Control and Automation*, pp. 761 – 764, 2007.

[100] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," *American control Conference*, pp. 1628 – 1632, June 1995.

[101] S. Julier and J. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, 1997.

[102] E. Wan and R. van der Merwe, "The unscented kalman filter for nonlinear estimation," *IEEE Symposium 2000 (AS-SPCC)*, Oct. 2000.

[103] S. Julier, "The scaled unscented transformation," *American Control Conference*, vol. 6, pp. 4555–4559, 2002.

[104] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Online outlier detection in sensor data using non-parametric models," in *VLDB'06*, 2006, pp. 187–198.

[105] S. Srkk and J. Hartikainen, "Ekf/ukf toolbox for matlab v1.2," Centre of Excellence in Computational Complex Systems Research, Helsinki University of Technology (HUT), Finland, 2007, http://www.lce.hut.fi/research/mm/ekfukf/.

[106] A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi, "Incremental training of support vector machines," *IEEE Tran. on Neural Networks*, vol. 16, no. 1, pp. 114–131, Jan. 2005.

[107] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," *Technical Report 98-14, Microsoft Research, Redmond, Washington.*, April 1998. [Online]. Available: citeseer.ist.psu.edu/platt98sequential.html

[108] G. Mainl, L. Kang, S. Lahaie, D. C. Parkes, and M. Welsh, "Using virtual markets to program global behavior in sensor networks," in *In Proceedings of the 11th ACM SIGOPS European Workshop*, 2004.

[109] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich, "Collaborative signal and information processing: An information directed approach," in *Proceedings of the IEEE*, vol. 91, no. 8, 2003, pp. 1199–1209.