5-2010

Anomaly detection in unknown environments using wireless sensor networks

YuanYuan Li
University of Tennessee - Knoxville, yli15@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by YuanYuan Li entitled "Anomaly detection in unknown environments using wireless sensor networks." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Lynne E. Parker, Major Professor

We have read this dissertation and recommend its acceptance:

Michael Berry, Michael Thomason, Hairong Qi, Wesley Hines

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by YuanYuan Li entitled "Anomaly Detection in Unknown Environments Using Wireless Sensor Networks." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science

Lynne Parker , Major Professor

We have read this dissertation
and recommend its acceptance:

Michael Berry

Michael Thomason

Hairong Qi

Wesley Hines

Accepted for the Council:

Carolyn R. Hodges
Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Anomaly detection in unknown environments using wireless sensor networks

YuanYuan Li
University of Tennessee - Knoxville, yli15@utk.edu

Anomaly detection in unknown environments using wireless sensor networks

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

YuanYuan Li

May 2010

Dedication

This dissertation is dedicated to my parents, Li Wen and Xu Libo, who love me unconditionally. They have taught me the importance of learning, hard work and perseverance. Without their inspiration and sacrifice I would never have achieved this lifelong dream. I also want to dedicate this dissertation to my husband, Dwi Sianto Mansjur, who gave me tremendous encouragement and support in my work for all these years. Finally, I want to dedicated this dissertation to my grandparents, Li Xianzhou, Yu Ying, Xu Gonglan, and Xu Xiuzhen.

Acknowledgments

This dissertation could not have been completed without the inspiration and support from many people. First and foremost, I would like to express my deepest thanks to my advisor, Dr. Lynne Parker. Her willingness to support my work and her inspiration and guidance throughout my studies have helped me to step into the interesting world of robotics. I thank her for giving me the instructions and suggestions that have been indispensable to the completion of this dissertation. I would also like to thank the rest of my thesis committee, Dr. Michael Berry, Dr. Michael Thomason, Dr. Hairong Qi and Dr. Wesley Hines, for their invaluable advice that helped to improve this work. I greatly appreciate their time and input to this dissertation. I sincerely thank Dr. Matt Welsh, of Harvard University, who shared the Reventador data from Volcano Tingurahua with me.

Within the Distributed Intelligence Laboratory (DILab), I really enjoyed working with robots, Blitz and Cappy, and the Mica2 wireless sensor motes. I would like to give my thanks to my fellow graduate students, Xingyan Li, Chris Reardon, Balajee Kannan, Michael Bailey, Richard Edwards, Tony Zhang, John Hoare, Shaddi Hasan, Hao Zhang, Nick Overfield, Yifan Tang, Rasko Pjesivac, Mike Franklin, Sudarshan Srinivasan, and Robert Lowe, for their constant encouragement and helpful suggestions on my work.

I also would like to thank my friends Tabitha Samuel and Kerry O'Donnell, who helped me to proofread my dissertation.

Finally, I must express my appreciation to the many friends who have accompanied me while being thousands of miles away from home. I sincerely thank Charles and Betty Danchertse who took care of me all these years ever since I first came to U.S.. I enjoyed the friendly environment of our Computer Science Department and the friendship with Siyuan, Shaotao, Chang, Chen, Peng, Teng, Ling, Xia, Wei, Jun, Yun, Huadong, Yihua, Vlado, and many others. Your encouragement through both good times and hard times have been a source of strength for me.

Abstract

This dissertation addresses the problem of distributed anomaly detection in Wireless Sensor Networks (WSN). A challenge of designing such systems is that the sensor nodes are battery powered, often have different capabilities and generally operate in dynamic environments. Programming such sensor nodes at a large scale can be a tedious job if the system is not carefully designed. Data modeling in distributed systems is important for determining the normal operation mode of the system. Being able to model the expected sensor signatures for typical operations greatly simplifies the human designer's job by enabling the system to autonomously characterize the expected sensor data streams. This, in turn, allows the system to perform autonomous anomaly detection to recognize when unexpected sensor signals are detected. This type of distributed sensor modeling can be used in a wide variety of sensor networks, such as detecting the presence of intruders, detecting sensor failures, and so forth. The advantage of this approach is that the human designer does not have to characterize the anomalous signatures in advance.

The contributions of this approach include: (1) providing a way for a WSN to autonomously model sensor data with no prior knowledge of the environment; (2) enabling a distributed system to detect anomalies in both sensor signals and temporal events online; (3) providing a way to automatically extract semantic labels from temporal sequences; (4) providing a way for WSNs to save communication power by transmitting compressed temporal sequences; (5) enabling the system to detect time-related anomalies without prior knowledge of abnormal events; and, (6) providing a novel missing data estimation method that utilizes temporal and spatial information to replace missing values. The algorithms have been designed, developed, evaluated, and validated experimentally in synthesized data, and in real-world sensor network applications.

Contents

List of Figures

Chapter 1

Introduction

An emerging class of Wireless Sensor Network (WSN) applications involves the acquisition of large amounts of sensory data from battery-powered, low computation and low memory wireless sensor nodes. Examples of such applications include volcano monitoring [Werner-Allen et al., 2006], animal habitat monitoring [Naumowicz et al., 2008], structural monitoring [Xu et al., 2004], etc. These systems share the same goal of detecting "interesting" events in an unknown environment over a period of time. The systems must acquire data at a constant rate and transfer high-fidelity data across a network. Moreover, the systems are also subject to unpredictable constraints on radio bandwidth, computational power, and energy usage over long periods of time. Given these constraints, it is typically not feasible to send all collected sensory data to a central location for processing and decision making. As a result, the sensor nodes should strive to process the raw sensor signal locally and perform local decision making to determine the most "interesting" signals/events, such as detecting anomalous events. Local processing and decision making avoids wasting resources on "uninteresting" data, such as avoiding sending normal raw sensor readings to a human operator for interpretation. Currently, most research in the WSN area has focused on hardware design, self-organization, various routing algorithms, or energy saving patterns [Naumowicz et al., 2008]. Practical distributed decision making algorithms for anomaly detection in a natural environment are still lacking.

In anomaly detection applications, a wireless sensor node in the network can monitor its local region and communicate through a wireless channel with other nodes to collaboratively produce a high-level representation of the environment. Using such a network, a large area can be monitored at a relatively low cost. The fundamental challenge is how to use the limited resources of wireless sensor nodes to deliver the most valuable information.

The goal of this research was to develop a practical, scalable, autonomous and robust anomaly detection system that is able to detect time-related anomalies and impute missing data in an unknown environment using a WSN and mobile robots. Although, much research exists that addresses some aspects of this problem, there is no single system that addresses all of the properties of this system. In order to understand the properties better, the basic characteristics of the WSN are first discussed in the following subsection.

1.1 Wireless Sensor Network characteristics

The term *Wireless Sensor Network* refers to a network of small, low-cost, low-power devices that can sense, actuate, and communicate information about their environment. Below is a summarized list of some fundamental characteristics of a WSN [Akyildiz et al., 2002], [Marrón et al., 2006], [Barrenetxea et al., 2008], and [Karl and Willig, 2005]:

1. Application specific: WSNs are conceivable for various applications with different spatial deployments that range from being very sparse to very dense. In sparse deployment with non-overlapping sensing ranges, the sink (sensor node to which information should be delivered) needs to collect sensory data from all sensor nodes in order to monitor the entire environment. On the other hand, if the sensor deployment is dense with overlapping sensing ranges, approximations of neighboring nodes' sensory data can be used to save communication costs.

2. Responsive to the environment: WSNs generally have to respond to the environment; their traffic characteristics can be expected to be very different from traditional wired networks, mobile ad hoc networks, etc. WSNs are likely to have long periods (e.g., months) of inactivity that can alternate with short periods (e.g., seconds or minutes) of very high activity when an event of interest occurs.

3. Wireless ad-hoc in nature: WSNs are often required to self-configure into connected networks. One basic scenario includes a fixed topology of sensor nodes, together

with a limited number of more powerful base stations, in which no maintenance or recharging is allowed after deployment. Therefore, cost minimization and autonomous behavior are desirable.

4. Physically distributed: WSNs are composed of a large number of nodes, each of which has an autonomous computational unit, and communicates with its neighbors via data packets. Data is also distributed throughout the nodes of the network and can be gathered at a central station only with high communication costs. Consequently, algorithms requiring global information from the entire network become very expensive.

5. Scable to large numbers of nodes: WSNs have to scale to large numbers (hundreds or perhaps thousands) of nodes in order to cover a large geographical area. This requires scalable learning and communication structures.

6. Energy restricted: WSNs are usually powered by batteries; therefore, energy is a scarce resource. When there are no “interesting” events, it is better to reduce the activities of the sensor nodes.

7. Simple and inexpensive: WSNs are inexpensive because the nodes have a small processing unit (typically, 8 MHz), and the memory of the sensor node is small (typically, 512 KB). The operating and networking software must be kept orders of magnitude simpler compared to modern desktop computers.

8. Low dependability: WSNs transmit data at low reliability mainly due to the unstable wireless channel. The use of a machine learning approach with local processing and decision making capabilities will lessen the impact of unreliable wireless communication.

9. Static: WSNs usually are composed of non-mobile (static) sensor nodes. However, in some applications, the sensor node can be mobile or partially mobile. There are two additional aspects of mobility to be considered in WSNs. First, the sensor network can be used to detect and observe a physical phenomenon that moves, e.g., to track chemical clouds. Second, the sinks of information in the sensor network can

be mobile as well. This mobility adds a dynamic element to the applications and can cause some difficulties for communications, which otherwise operate efficiently in fully static scenarios.

Because of these fundamental characteristics, WSNs hold the promise of revolutionizing sensing in a wide range of application domains. This dissertation research has been developed with these fundamental characteristics in mind while designing the anomaly detection system.

1.2 Problem statement and challenges

This dissertation developed the design of a practical, scalable, autonomous, and robust anomaly detection system that is able to detect time-related anomalies and estimate missing data in a previously unknown environment using a WSN and at times mobile robots. The term “unknown environment” means that there is no possibility of pre-programming the state of the environment and the types of anomalies before the system deployment. Both the state of the environment and the types of anomalies must be learned by the system autonomously over an initial period of time.

Aside from being able to detect time-related anomalies and impute missing values, a set of desired characteristics of the system have been identified. Specifically, the desired characteristics of the system are as follows:

1. *Able to detect anomalies in an unknown environment with minimum human supervision.* The system should able to learn to detect anomalies “online” and “unsupervised” (minimum human supervision). This characteristic makes supervised, offline learning algorithms, such as Bayesian networks [Janakiram et al., 2006], unsuitable for the WSN.

2. *Able to easily scale to large numbers of nodes.* Since a WSN typically has a large number of sensor nodes, tuning the parameters of learning algorithms can be a long and tedious process. Therefore, the learning algorithm should have as few parameters to adjust as possible.

3. *Able to support a hierarchical structure.* Heinzelman, et al., show in [Heinzelman et al., 2000] that a hierarchical structure in a WSN is able to decrease communication requirements by reducing the size of the data transmitted; this in turn saves energy.

4. *Able to continuously monitor the environment.* In the developed approach, human intervention is not required to reset the system after an anomaly has been detected. The system can reset itself.

5. *Able to perform sensor fusion.* Motes generally contain multiple sensors. The learning system should be able to detect anomalies from abnormal combinations of sensory data, instead of making decisions based only on readings from individual sensors.

6. *Computationally inexpensive.* The sensor nodes generally have limited computational resources and limited power. Typically, machine learning techniques like Expectation Maximization (EM)-based and gradient-based algorithms are computationally expensive, and thus are not appropriate for WSNs.

7. *Memory efficient.* The learning algorithm has to be small enough to be implemented and installed on sensor nodes with limited memory. Thus, learning algorithms that require large amounts of memory during the learning process, such as those that use particle filters, might not be a good choice.

8. *Modular.* The system should be designed to serve as many applications as possible without reprogramming. Each component can be removed if its capability is not required. For example, if time-related anomalies are not of interest, then one should be able to turn off the time model; the system would then simply detect anomalies in the sensor signatures.

9. *Decentralized.* The sensor nodes should be able to process data locally and support local decision making.

1.3 The approach

To address the problem and challenges, a novel anomaly detection system is designed and developed for WSNs. The WSN consists of a set of static sensor nodes that are pre-deployed into the environment. The sensor nodes first learn an initial model of the environment using a classifier. In addition to the classifier, a time analysis model is built to represent the time state of the environment. Together these models define the *normal* model of the environment. After a period of deployment (the training phase), all deviations from the *normal* model are treated as anomalies. Upon detection of an anomaly, an alert is generated. At this point, for some applications, an autonomous mobile robot responds to the anomaly alert by traveling to the location where the anomaly has occurred. In future work, the mobile robot may use additional sensors (e.g., microphones, cameras) to verify that there is indeed an anomaly in the area, in order to make the system more autonomous and robust. If there was indeed an anomaly in the environment, the robot could notify its upper level supervisor, i.e., a human operator. Moreover, the mobile robot could move to different locations to provide more coverage and perform various tasks that static sensors cannot, such as locating power sources, seeking out repair, or requesting the dispatchment of other nodes. However, these additional mobile robot tasks are beyond the scope of this dissertation.

One of the challenges in this research is to determine a systematic procedure to train the sensor nodes so that they are sensitive only to "unique" events that are of interest in specific applications — for example, the use of a microphone sensor for the purpose of detecting volcanic activities. In addition, there is generally more than one type of sensor mounted on a sensor board, which means that fusion of sensor data is needed. To address this challenge, a machine learning technique has been incorporated into the WSN so that the networks can automatically learn to recognize *normal* and *abnormal* modes of operation. The approach makes use of a classifier to categorize the environmental states and perform sensor fusion, specifically, the Fuzzy Art Resonance Theory (ART) [Carpenter and Grossberg, 1991] neural network. The Fuzzy ART system is an unsupervised Artificial Neural Network (ANN) that can perform dimensionality reduction and pattern classification simultaneously. There is no off-line training phase required by the Fuzzy ART neural network. Moreover, the algorithm is simple enough to be implemented on the small platform of the Crossbow motes [Crossbow, 2008], while maintaining a good performance level. Fuzzy ART was first implemented in the WSN area by Kulakov and Davcev in [Kulakov and Davcev, 2005]. The developed approaches build upon this Fuzzy ART model, and improve anomaly detection performance by incorporating temporal and spatial information.

One shortcoming of the baseline Fuzzy ART neural network presented by Kulakov and Dacev is that it cannot detect time-related changes. Therefore, to detect time-related anomalies in WSNs, the developed approach makes use of a three-step detection scheme. First, the system extracts semantics out of the temporal sequence using a symbol compression technique called Lempel-Ziv-Welch (LZW) [Welch, 1984]. Then, it uses a Variable Memory Length Markov Model (VMM), i.e., a Probabilistic Suffix Tree (PST) [Ron et al., 1996], to model the compressed temporal sequence. Finally, the system uses a Universal Background Model (UBM) [Reynolds et al., 2000] likelihood-ratio detector to detect anomalies in the time patterns. The symbol compression module makes the system more flexible when deciding the type of time related anomalies to detect in different applications. For example, if it is important to only model the sequences of events rather than the time duration of each event, symbol compression can be used to map the same consecutive categories into a new symbol for the PST to model. The PST models contain information on the state transitions as well as the time duration in a state using a variable number of states. The UBM detector is a likelihood-ratio detector to match the current observation sequences against the normal PST model. If the probability of the observation sequence is below a threshold, an abnormal alert is raised.

Due to unreliable wireless communication, missing data can be a major problem preventing a classifier from performing accurately. Missing data may also be due to synchronization problems, sensor faults, communication malfunctions, or malicious attacks. This issue is indeed unavoidable in a system based on WSNs. This problem was not addressed by Kulakov and Davcev in their implementation [Kulakov and Davcev, 2005]. In this study, around 40% of sensor data are found missing on average; it is not uncommon to lose 85% of the readings in a multi-hop sensor network [Manjhi et al., 2005]. Therefore, it is hard for a classifier to learn and categorize the state of the environment with so many missing values. With the high rate of missing values, an imputation algorithm[1] to estimate the missing values is preferred instead of ignoring them. The imputation technique is needed in sparse WSNs where every sensor reading is important and/or with limited amount of training data and the system cannot afford to lose any information. Additionally, when dense

[1]To substitute for missing data is called *imputation* in statistics.

WSNs can become sparse as sensor nodes wear out over time, such imputation technique is also envisioned to make the learning algorithm more robust. Different ways to impute the missing values are studied in this research. It is commonly known that environments are correlated in time and space [D'Costa et al., 2004]. This study has shown that utilizing spatial and temporal information to impute missing values yields good performances. The developed approach makes use of Nearest Neighbor (NN) imputation technique that uses a kd-tree [Friedman et al., 1977] data structure to automatically learn space and time correlations. A kd-tree is a space-partitioning data structure for organizing points in a K-dimensional space. Each space data is represented by a vector of K dimensions; the nearest neighbors to a specified data vector can be searched according a distance measure. This research designed a weighted Euclidean distances and variances to make the kd-tree search more suitable for WSNs with missing data. The goal of using a kd-tree to model the space correlations is to reduce the amount of data examined while searching for the closest match.

1.4 Contributions

The contribution of this thesis is the development of an anomaly detection system — a novel general approach that autonomously detects anomalies in an unknown environment using sensor data that is collected by a Wireless Sensor Network in a distributed fashion. This research makes several important contributions to Wireless Sensor Network research, including:

1. *Makes PST practical for time modeling in WSNs by extracting semantics out of temporal sequences.* This is the first work that makes use of symbol compression techniques to compress temporal signals in WSNs.

2. *Enables the system to save communication costs by compressing temporal sequences in WSNs.* Transmitting decisions at a lower frequency can potentially reduce communication costs in WSNs.

3. *Enables the system to detect anomalies without prior knowledge of the types of anomalies in the environment.* Typical formulations for detecting time-related anomalies

require models of anomalies. An UBM likelihood-ratio detector enables the system to detect time-related anomalies without prior knowledge of different types of anomalies.

4. *Enables the system to organize time and space correlated data into a kd-tree data structure.* This is the first work that makes use of a kd-tree to autonomously represent temporal and spatial relationships among the sensor nodes in WSNs.

5. *Provides a new spatial-temporal missing data imputation technique that is practical for WSNs.* The system takes advantage of spatial and temporal correlated data in WSNs to estimate missing data. Additionally, this is a non-parametric approach which does not assume any distributions of the data.

6. *Presents weighted Euclidean distance and variance metric for kd-tree construction that takes into account missing data in WSNs.* This is the first work that has developed the weighted metric for kd-trees, which makes the search tree more suitable for WSNs when handling missing data.

7. *Compares and contrasts different missing data imputation techniques for WSNs.* Although there are some existing works on missing data estimation for WSNs, some developed algorithms are not practical for implementation on sensor nodes. Other works only state the way they handle missing data instead of formally analyzing the results, explaining the approach's validity, and comparing it with other techniques.

8. *Uses an anomaly detection system that is modular and flexible in design.* The developed anomaly detection system is designed to serve as many applications as possible without rewriting the learning program. The modular design allows us to separate different kinds of detection as much as possible. For example, consider a system with three major components: a missing data estimator, a classifier, and a time model. If detecting time-related anomalies is not of interest for some applications, the operator can simply turn off the time model learning. If in some environment wireless communication is very stable, the missing data estimator can be turned off to save energy.

The organization of this dissertation is as follows: Chapter 2 reviews the related work in WSNs including its applications, anomaly detection systems, time-series analysis, and missing data. Chapter 3 presents the novel distributed anomaly detection approach developed in this dissertation. Chapter 4 presents novel approaches to modeling temporal sequences and detecting time-related changes. Chapter 5 presents novel missing data imputation techniques developed for WSNs. Chapter 6 presents an intruder detection application using a WSN and autonomous mobile robot. Chapter 7 summarizes the main contributions of this dissertation and describes potential extensions of the anomaly detection approach in WSNs.

Chapter 2

Literature review

The objective of this research is to design a practical, efficient, scalable, and robust anomaly detection system using Wireless Sensor Networks (WSN) in an unknown environment. As mentioned in the introduction, the system should be able to detect anomalies in an unknown environment with minimal human supervision. To achieve this objective, research and results from several fields of study are required. In this chapter, the research and results that are relevant to the research objective are reviewed. Although there are several aspects of anomaly detection systems addressed in the existing WSN literature, no system has addressed all the objectives of the system developed in this dissertation.

First, applications of WSNs used for environment monitoring and intrusion detection are reviewed in Sections 2.1 and 2.2. Then, Section 2.3 reviews machine learning techniques used in WSNs. Finally, time-related analysis and missing data techniques used in WSNs are reviewed in Sections 2.4 and 2.5.

2.1 Application of WSNs for environment monitoring

A wide range of applications make use of Wireless Sensor Networks for environment monitoring, such as monitoring animals, monitoring building structures, monitoring volcanic activities, flood or leak detection, etc. Many of the systems are deployed for data collection with no learning involved. This section discusses key examples of this work.

Mainwaring, et al., developed a system for habitat monitoring [Mainwaring et al., 2002].

They described design requirements that cover the hardware design of the nodes, the design of the sensor network, and the capabilities for remote data access and management. A system architecture is presented to address the requirements for habitat monitoring in general, and an instance of the architecture for monitoring seabird nesting environment and behavior is presented. The deployed network consists of 32 nodes on a small island off the coast of Maine streaming live data to the web. The system collects the temperature data from the environment to a central server. There is no learning in the data gathering process. Some of the same authors provided an in-depth study of applying WSNs to real-world habitat monitoring in [Polastre et al., 2004]. They analyzed the environmental and node health data to evaluate system performance. They showed that the sensor data is also useful for predicting system operation and network failures. Based on over one million data readings, they analyzed the node and network design and developed network reliability profiles and failure models. In [Naumowicz et al., 2008], the authors describe a design and deployment of a WSN that delivered high resolution sensor data while monitoring seabirds on a UK National Nature Reserve. However, unlike the system presented in this dissertation, these systems collect sensory data at a central location on a continuous basis over a long period of time. Furthermore, there is no learning involved during the data collection or transfer of sensory data. It generally takes a significant amount of power to transfer multi-dimensional data across a large network over a long period of time.

In addition to animal habitat morning, WSNs have also been applied in structural monitoring applications. For example, Wisden [Xu et al., 2004] has presented the first wireless structural monitoring system. The Wisden system continuously collects structural response data from a multi-hop network of sensor nodes, and then displays and stores the data at a sink for processing. However, this is a centralized system; a distributed system where each node is able to monitor and process information by itself is more desirable. The works of [Stoianov et al., 2007] and [Schulz et al., 2008] addressed a water leaking monitoring system. Challenges of this system include sampling at high data rates, maintaining aggressive duty cycles, and ensuring tightly time-synchronized data collection. Santini, et al., have showed the feasibility of WSNs for assessment of noise pollution in urban environments in [Santini et al., 2008]. They presented a prototype for the collection and logging of noise pollution data based on the Tmote platform, which they used to perform indoor and outdoor noise pollution measurements. Furthermore, they presented tinyLAB, a Matlab-based tool developed in the context of this work, which enables real-time acquisition, processing and visualization of data collected in WSNs. Li and Liu [Li and Liu, 2007] have presented a coal mine monitoring system that is named Structure-Aware Self-Adaptive system (SARA). SARA employs a hole-detection algorithm to monitor the inner surface of tunnels by utilizing radio signals among sensor nodes to model the structure of the sensor network. Experiments have shown that with proper deployment, the system is able to rapidly detect structural variations caused by underground collapses. These systems all operate in a centralized fashion. They collect sensor data to a sink node and let the node post-process collected data. In addition, the systems have no learning involved when detecting structural changes. Thus, these systems are not general or flexible enough to apply in other places or to other application areas.

In the works of [Werner-Allen et al., 2005] and [Werner-Allen et al., 2006], the authors described their experiences using a WSN to monitor volcanic eruptions with low-frequency acoustic sensors. They developed a wireless sensor array and deployed it at Volcan Tungurahua, an active volcano in central Ecuador. The network collected low-frequency acoustic signals at 102 Hz, transmitting data over a 9km wireless link to a remote base station. In addition to continuous sampling, they developed a distributed event detector that automatically triggers data transmission when a well-correlated signal is received by multiple nodes. They used linear regression to predict missing data. They used a threshold-based detector and an Exponentially Weighted Moving Average (EWMA)-based detector. The EWMA detector calculates two moving averages with different gain parameters, representing short-term and long-term averages of the signal. The missing data estimator component makes the system robust. Additionally, intelligence is added to the local data collectors. However, the system is a data collection system rather than a distributed anomaly detection system. The authors made the volcano dataset available for this research. Chapters 4 and 5 make use of this data to test the developed anomaly detection system and missing data estimator.

2.2 Intrusion and outlier detection in WSNs

Many intrusion and outlier detection systems that are implemented in the area of WSNs focus on detecting network intrusion instead of detecting intruders in the physical environment, e.g., [Techateerawat and Jennings, 2006], [Hai et al., 2007], [Loo et al., 2006], [Eskin et al., 2002], [Onat and Miri, 2005], [Banerjee et al., 2005]. Existing detection systems either use a statistical based detection technique or a swarm intelligence-based technique.

The works of [Techateerawat and Jennings, 2006] and [Hai et al., 2007] presented Intrusion Detection Systems (IDS) for a sensor network that is based on the network activities (e.g., number of success and failure of authentications). The system compares event data with signature records to find harmful attacks from an intruder. Additionally, the authors of [Hai et al., 2007] applied the detection system in a cluster-based sensor network very much like the developed system in this dissertation. This type of detection system can only identify the anomalies that it has seen before. However, this research is interested in detecting anomalies in unknown environments, in which there are no abnormal prototypes available for the system to learn.

The authors of [Onat and Miri, 2005], [Loo et al., 2006], and [Eskin et al., 2002], presented intrusion detection schemes that build a model of normal traffic behavior, and then use this model of normal traffic to detect abnormal traffic patterns. Their approaches are able to detect attacks that have not been previously seen. The detection system also has a feature selection phase where the features are specific to network traffic activities. The anomaly detection system presented in this dissertation takes a similar approach by building a normal model of the environment; sensor signals that do not match the normal models are considered as anomalies. However, this prior research does not address the issue of detecting time-related anomalies.

In [Banerjee et al., 2005], an ant colony based intrusion detection mechanism that could keep track of the intruder trails is presented. This technique can work in conjunction with conventional machine learning based intrusion detection techniques to secure sensor networks. This work tracks the paths of intrusion after anomalies are detected. Tracking the path of an intruder is one of the future directions of this research. However, it is not included in the scope of this dissertation.

There is limited existing research that detects anomalous sensor signals in wireless sensor networks. An outlier detection algorithm based on Bayesian Belief Networks (BBN) is presented in [Janakiram et al., 2006]. The system is also able to estimate missing values in the sensor data. The BBNs are able to capture the relationship between the attributes of sensor nodes as well as spatial temporal correlations that exist among the sensor nodes. However, the technique requires offline training and human knowledge of the environment. [Branch et al., 2006] addressed the problem of unsupervised outlier detection in wireless sensor networks. The authors have used a Nearest Neighbor algorithm. Both centralized and distributed algorithms were implemented and examined using a simulator and real sensor data streams. However, none of the systems mentioned above detect time-related anomalies in the environment or deal with the missing data problem.

2.3 Machine learning techniques in WSNs

Various machine learning techniques are used in WSNs. However, many of them have focused on improving the communication protocols of the WSNs. In general, they usually use one of the following three techniques: swarm intelligence, reinforcement learning or statistical-based learning.

The most widely used routing protocol that uses ant colony optimization is AntNet [Di Caro and Dorigo, 1998], which is an online Monte Carlo technique. There are many variations of this work, such as AntHocNet [Caro et al., 2005], ARS [Oida and Sekido, 1999], MANSI [Shen and Jaikaeo, 2005], UniformAnts [Subramanian et al., 1997], etc. All of these algorithms use the notion of stigmergy, which is indirect communication that takes place among individuals through modifications induced in their environment. Moreover, attempts to solve classification or detection problems have been carried out using this optimization technique [Holden and Freitas, 2004]. However, the classifiers based on ant-colony optimization generally extract a set of class rules from the data. This process requires knowledge of the environment, and needs to be carried out offline.

Q-Routing [Boyan and Littman, 1994] is a type of reinforcement learning, and is one of the earliest works in packet routing. A simple Q-learning approach used in multicast protocol is called Feedback Routing for Optimizing Multiple Sinks (FROMS) [Egorova-Förster and Murphy, 2007]. Other variations of the Q-learning technique used for routing protocols include [Sun et al., 2002], DRQ-Routing [Kumar and Miikkulainen, 1997], SAMPLE [Dowling et al., 2005], TPOT-RL [Stone, 2000], etc. The drawback of using Q-learning is that it is only tractable for small environments with relatively few states and actions. However, reinforcement learning in real-world problems requires coping with large (possibly infinite) states, observations, and action spaces [Langford and Zadrozny, 2005]. Therefore, the reinforcement learning technique is not suitable for the environment that is considered in this research.

Other distributed classification algorithms in sensor networks are based on statistical techniques, such as [Loo et al., 2006], [Eskin et al., 2002], [Rabbat and Nowak, 2004], [Guestrin et al., 2004], [Duarte and Hu, 2004], [Kulakov and Davcev, 2005], etc. Rabbat, et al., presented a Distributed Expectation-Maximization (DEM) algorithm for density estimation and clustering for a wireless sensor network. However, the training phase is computationally expensive, and requires prior knowledge of the number of classes [Rabbat and Nowak, 2004]. Guestrin, et al., presented a general framework in which the nodes in the sensor network collaborate to fit a global function to each of their local measurements [Guestrin et al., 2004]. The algorithm is based upon a kernel linear regression model, where the model takes the form of a weighted sum of local basis functions. Regression models generally suffer from problems such as model uncertainty and bias-variance trade-off. There are methods based on fix-width clustering, such as K-Nearest Neighbor, Gaussian and Support Vector Machine (SVM) [Loo et al., 2006], [Eskin et al., 2002], [Duarte and Hu, 2004]. The fixed-width clustering algorithm has been shown to be highly effective for anomaly detection in IP networks. However, the clusters in sensor signals may come with different widths. Techniques like K-Nearest Neighbor and C-means generally require a priori knowledge of the number of classes or an estimation of K. A Gaussian classifier based on the Maximum-Likelihood approach can be sensitive to the choice of starting values. Additionally, the likelihood equations need to be specifically worked out for a given distribution and estimation problem. This is often non-trivial. A problem with SVM is that the technique is designed to solve a 2-class problem. Extending SVM classifiers to solve multi-class problems remains as an open research question. The research in this dissertation is addressing the issue of anomaly detection, which can be formulated as 2-class problem, i.e., normal and abnormal. These approaches need prior knowledge of abnormal events. In addition, the ability to model the unknown environment into natural states helps the human understand and analyze the events that occurred. Kulakov and Davcev implemented ART and Fuzzy ART neural network algorithms in a WSN to detect unusual events [Kulakov and Davcev, 2005]. The neural networks are unsupervised learning methods for categorization of sensory inputs. The presented neural networks classifiers have distributed short-term and long-term memory of the sensory inputs. Since it is small enough to be implemented on the sensor nodes, and supports a hierarchical learning structure, the developed anomaly detection system makes use of this Fuzzy ART neural network as the baseline classification algorithm to detect anomalies.

Heinzelman, et al., presented a well-known energy-efficient data routing protocol in a WSN called LEACH (Low Energy Adaptive Clustering Hierarchy) [Heinzelman et al., 2000]. LEACH divides the nodes into clusters, and clusterheads aggregate data from their cluster members. Moreover, the membership of a cluster is adaptive in LEACH. With this hierarchical structure, the system is able to decrease communication requirements by reducing the size of the data transmitted; this in turn saves energy. The research in this dissertation also uses a hierarchical learning and communication structure. However, adaptive cluster membership selection is not implemented at the current time.

Hierarchical Cluster Based-Routing (HCR) [Matin and Hussain, 2006] is an extension of the LEACH clustering algorithm, which uses a genetic algorithm to form the clusters. It is assumed that the base station has complete knowledge of the network (e.g., topology and battery status of all nodes). It then uses a genetic algorithm to compute the best clusters for this network and sends a broadcast with the full cluster information to all nodes in the network (wide-range one-hop broadcast). The algorithm has several severe drawbacks. First, it drains the batteries of clusterheads quickly, which is inefficient. Second, it assumes global knowledge about the network topology and cannot handle topology changes or asymmetric links. Using a genetic algorithm is also unsuitable, since it is time- and computation-intensive. The learning system should function in a fully distributed fashion. Furthermore, the system should be able to respond in real time. Lastly, it should be practical to be implemented on the sensor nodes. Therefore, HCR is not considered for this research.

2.4 Time-related analysis in WSNs

Various regression models have been presented for time-related analysis WSNs, such as autoregressive models [Tulone and Madden, 2006], Least-Square-Error based Linear forecast method [Lim and Shin, 2005], and the Non-seasonal Holt-Winters Linear forecast method [Lim and Shin, 2005]. Most of these systems are linear regression models; these time series models have been widely used outside the wireless sensor network domain as a way to approximate and summarize time series with applications in finance, communication, weather prediction, and a variety of other areas. However, as mentioned earlier, using regression models to detect time-related anomalies normally involve a complex parameter estimation process. They may suffer from model mismatch and bias-variance trade-off problems.

There has been some work on the use of probabilistic time-series models in WSNs, e.g., Kalman Filter-based models. These systems rely on a combination of local and global probabilistic models, which are kept in synchronized to reduce communication between sensor nodes and the network sink [Zarchan and Musoff, 2000], [Jain et al., 2004], [Chu et al., 2006]. Moreover, Kalman Filter-based models are sophisticated and require significant computation.

The fixed length Markov model is another commonly used technique for time series analysis [Kedem and Fokianos, 2002]. Examples of fixed order Markov models include the Markov Chain, Hidden Markov Model, etc. Due to the limited resources in WSNs, building high and fixed order Markov models is not feasible. Mazeroff et al., [Mazeroff et al., 2008] implemented Variable Memory Markov model (VMM) in the forms of Probabilistic Suffix Tree (PST) models and Probabilistic Suffix Automata (PSAs) to build models of benign application behavior with the goal of detecting malicious applications in Windows XP. Note that in practice, the VMM is usually implemented in the form of either a PST or a PSA model. The two models are proven to be equivalent [Ron et al., 1996] and a PSA model can be inferred directly from a PST model by using the algorithm described in [Mazeroff et al., 2008]. PST models depend on a fixed number of random variables; in PST models this number of conditioning random variables may vary based on the specific observed realization. The PST model is a data-driven technique, which can be easily applied to

WSNs. This dissertation research makes use of the PST to model time sequence data and to detect time-related anomalies. The PST models have been applied in WSNs for object tracking in [Peng et al., 2006] and [Tsai et al., 2007]. PST models can be expensive in both space and time if not implemented carefully. Many researchers developed algorithms to build PST models in linear time and space [Kurtz, 1999], [Apostolico and Bejerano, 2000], [Sun and Deogun, 2004]. Lin et al., have developed an online PST-based time-series visualization tool to aid aerospace analysts [Lin et al., 2004]. In addition, the method is automatic, and can be applied without assuming any preliminary information. The PST model has been shown in applications involving prediction in [Begleiter and Yona, 2004] as well. This dissertation does not use prediction; however, it is a nice feature to have for sensor networks, since by predicting the values, the system can use the predicted values instead of constantly querying the network.

2.5 Missing data techniques in WSNs

Some initial experiments of this dissertation showed that on average about 40% of the sensor readings were missing. Several researchers working with WSNs (e.g., [Zhao et al., 2003], [Fletcher et al., 2004], [Werner-Allen et al., 2005]) have also encountered missing sensor reading problems. Several solutions have been suggested to tolerate this error at the communication level, such as link quality profiling [Zhao et al., 2003] or reliable data transportation protocols. This type of solution usually requires retransmitting the lost data, which costs additional transmission power. Moreover, the retransmission process causes delays in the anomaly detection decision process. There are higher-level algorithms developed to estimate missing sensor data, such as Fletcher, et al., [Fletcher et al., 2004]. They have estimated missing sensor data using a jump linear system and Kalman filtering. However, these regression models usually require extensive computation and an offline training process, in addition to large storage capabilities. Also, the presented approach was not tested on sensor nodes. Werner-Allen, et al., [Werner-Allen et al., 2005] built a simple linear Autoregressive model to estimate missing values based on collected historical sensory readings. Granger et al., [Granger et al., 2000] have suggested three different ways to modify the Fuzzy ART neural network to compensate for missing input data. Their developed solutions were not tested on sensor network data, which tends to have high correlations both in time and space. In this dissertation, this approach was compared with the developed spatial-temporal imputation technique; the result was presented in [Li and Parker, 2008]. Although missing data is not a well-studied area in WSNs, missing data in general is a well-studied subject in statistics. Little and Rubin [Little and Rubin, 1986] wrote an excellent introduction on statistical missing data techniques, such as Least Squares Estimates, Bartlett's ANCOVA, and likelihood-based approaches. The imputation technique is needed in sparse WSNs where every sensor reading is important and/or with limited amount of training data and the system cannot afford to lose any information.

This dissertation also developed a Nearest Neighbor (NN) imputation technique. It overcomes some limitations posed by the previously developed spatial-temporal imputation technique. NN classifiers were first introduced by Fix and Hodges [Fix and Hodges, 1951]. The NN methods gained popularity in machine learning through the work of Aha in [Aha, 1992], who showed that instance-based learning can be combined with attribute-weighting and the pruning of noisy instances. The resulting methods perform well in comparison with other learning methods. The use of the kd-tree to improve nearest-neighbor classification time was introduced by Friedman et al., [Friedman et al., 1977]. The approach used in this dissertation was directly modified from the algorithm given by Andrew Moore [Moore, 1990] and Friedman et al., [Friedman et al., 1977]. Moore along with Omohundro [Omohundro, 1987], pioneered kd-tree usage in machine learning.

Chapter 3

The overall architecture

This chapter gives an overview of the WSN anomaly detection system developed in this dissertation. First, the overall learning and communication architecture is introduced in Section 3.1. Then, the basic Fuzzy ART neural network is briefly described in Section 3.2. An overview of the approach to time analysis is presented in Section 3.3. Finally, Section 3.4 gives a brief description of the missing data imputation approaches developed in this research.

3.1 Overall architecture

In the developed learning system, sensor nodes are arranged in a hierarchical structure, as shown in Figure 3.1. In a static WSN, sensor nodes are divided into clusters. Each cluster has a clusterhead and multiple cluster members. Each cluster covers a geometric region and is responsible for detecting the environmental changes in that region. Both cluster members and clusterheads run an identical detection system — a missing data estimator, a classifier and a time-series model. Cluster members read in raw sensor readings s_i (e.g., light and sound) from the environment as input, and then classify data into categories c_i by using a Fuzzy ART neural network. Note that if the data dimensions of different sensors are different, then the Fuzzy ART neural network is applied to individual sensors first, then the resulting data is fused together to produce one class label. After the classification process, cluster members send their category labels to their clusterheads. Due to unstable

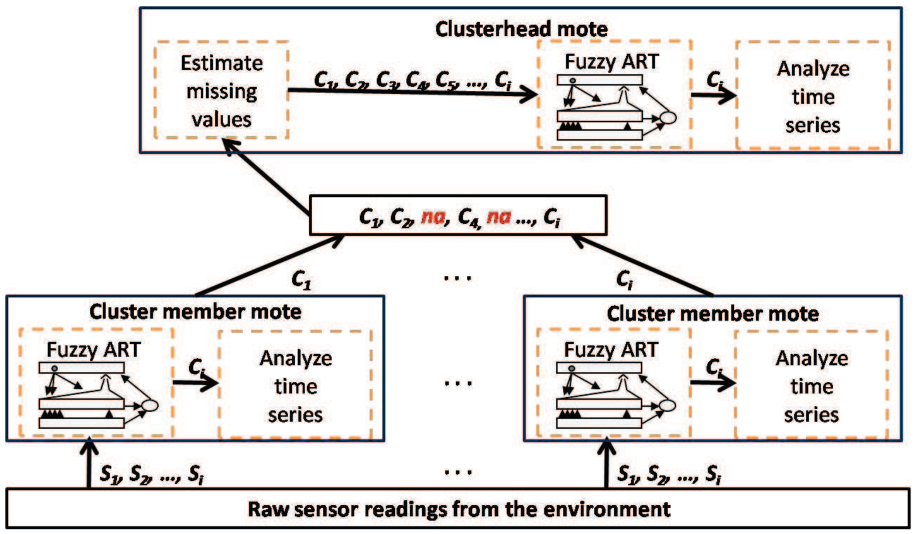


Figure 3.1: Overall communication and learning architecture, extended to estimate missing data and perform time analysis.

wireless communications, clusterheads often cannot receive category labels from all of their cluster members. Thus, the clusterheads first pre-process the collected category labels by estimating and replacing the missing values by using a spatial-temporal technique. Then, the processed categorizations are used as input to their classifiers and are fused together to reduce the size of the raw sensory data. The output of the Fuzzy ART neural network is a category label c_i. If the category label does not match one of the existing prototypes, a change is detected by the system. After the classification process is finished, the system performs further checks for time-related changes. The learning system has a hierarchical structure — clusterheads may have higher level clusterheads which classify their output class labels. Finally, the root node obtains the final model of the environment. With this architecture, the WSN can be easily scaled to large numbers of sensors. At the same time, this hierarchical approach reduces communication, which in turn saves energy in the WSN.

The Fuzzy ART neural network alone cannot detect time-related abnormal events. For example, if people turning on the lights during the day and turning off the lights when they leave work is considered as a normal event, then an intruder turning on the lights only briefly in the evening should trigger an alarm. As another example, if people turning on the lights then making noise inside of the room is normal, then the opposite sequence of making noise before turning on the lights should also trigger an alarm. The time analysis model detects these types of time-related changes by taking a stream of output class labels c from the Fuzzy ART neural network and comparing it against a learned time model M. If the querying temporal sequence is unlikely to occur, an alarm is raised. To achieve this, two temporal analysis models have been studied. The first time analysis approach is a heuristic Finite State Automaton (FSA) that models the time duration in each environmental state as a normal distribution. In addition, the FSA also models the state transition probabilities. The second temporal modeling approach is a significant enhancement of the first approach. It uses a symbol compression technique to extract the semantic symbols from the temporal sequences. It then uses a Probabilistic Suffix Tree to model the temporal semantic events. Both approaches analyze time using a discrete representation. Discrete modeling and processing have several advantages over continuous measurements and models, including: 1) sensor data is often discrete (e.g., certain radar systems [Wang and Krishnamurthy, 2008]); 2) environments that are modeled with discrete states have clear physical interpretations and are therefore natural and easy for humans to interpret (e.g., eruption or no eruption vs. vibration measurements); and 3) data compression techniques, which the system uses to reduce the size of the observations, typically require discrete state representations. The details of both temporal modeling techniques are presented in Chapter 4.

Missing sensor values add noise to the data, which degrades the performance of the classifier if not handled properly. To make the learning process more robust to this type of noise, the system pre-processes the data by estimating the missing values. A novel spatial-temporal imputation technique has been developed to solve this problem. This research has compared and contrasted different missing data imputation techniques with the developed technique. The system is able to achieve high accuracy by taking advantages of temporal and spatial correlations in WSNs. To overcome some limitations that are posed by the spatial-temporal imputation technique, another Nearest Neighbor (NN) imputation technique has been developed. The NN imputation technique utilizes a kd-tree to organize temporal and spatial correlated data and search the tree efficiently to fill in missing values. A detailed study is given in Chapter 5.

With this communication and learning architecture, the system is able to detect both abnormal environmental changes and time-related changes. The developed design is flexible because it allows the operator to turn off the time module if time anomaly detection is not of interest.

3.2 Fuzzy ART neural network

Adaptive Resonance Theory (ART) is a neural network architecture developed by Grossberg and Carpenter [Carpenter and Grossberg, 1991]. The Fuzzy ART neural network was first implemented in a WSN by Kulakov and Davcev in [Kulakov and Davcev, 2005]. The baseline Fuzzy ART neural network used in this dissertation is implemented in the same way. The motivation for choosing this Fuzzy ART model lies in its simplicity of computation. The Fuzzy ART neural network system is an unsupervised artificial neural network that can perform dimensionality reduction and pattern classification. The Fuzzy ART neural network is able to take analog input. It is able to discover both regularities and irregularities in the redundant input data by an iterative process of adjusting weights of interconnections between large numbers of simple artificial neurons. The Fuzzy ART neural network considers signal readings from all sensors simultaneously, and is able to learn continually. The neural network creates a new category when the input vector is significantly different from existing categories. The sensitivity of the categorization process can be controlled by a vigilance parameter ρ; higher vigilance produces highly detailed memories (many fine-grained categories), while lower vigilance results in more general memories (fewer, more general categories). When the vigilance value is equal to one, all the data will be “remembered” and assigned into different categories. In practical scenarios, the vigilance parameter is set to a certain threshold to capture only the representative features/signatures of the environment. Figure 3.2 gives a representation of their Fuzzy ART neural network. A typical Fuzzy ART neural network has three layers: an input layer (F0), a comparison layer (F1) and a category layer (F2). The learning process of the Fuzzy ART neural network is presented in detail as follows.

Each input vector I ($I_j \in [0,1]$; and $j = \{1, 2, ..., N\}$) is compared with each category (node) in the F2 layer to classify it with its best match. The choice function A_j is defined

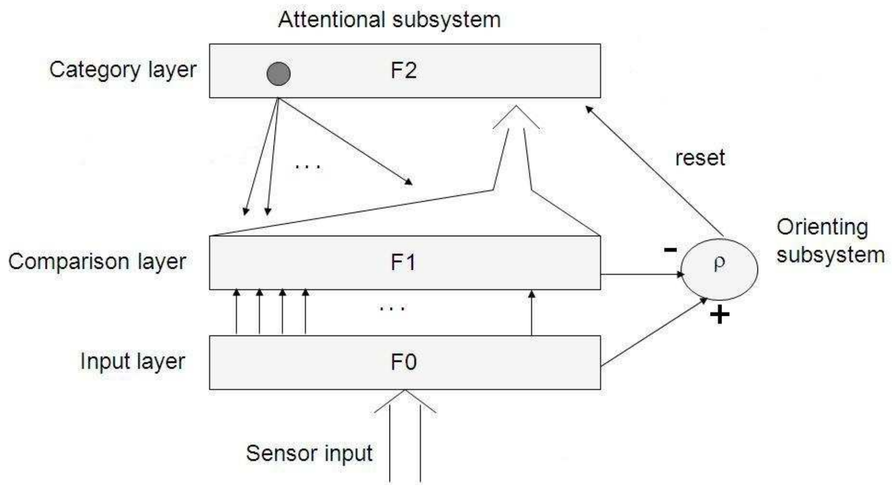


Figure 3.2: A typical Fuzzy ART architecture (from [Kulakov and Davcev, 2005]).

as:

$$A_j(I) = \frac{|I \wedge w_j|}{\epsilon + |w_j|} \tag{3.1}$$

where parameter w_j is the binary weight vector of category j, and parameter $\epsilon \in [0, 1]$. The ϵ parameter is generally used to make the denominator not equal to zero. The fuzzy AND operator $\wedge$ is defined by $I \wedge w_j = min(I, w_j)$ and the operator $| \cdot |$ is defined by $|x| \equiv \sum_{i=1}^{M} x_i$, where x is an arbitrary variable. Then, the approach selects the node J in the F2 layer that has the highest match $(A_J = max\{A_j | j = 1, ..., N\})$. The weight vector of the winning node (w_J) is compared to the current input at the comparison layer. The training process starts if the match function of the chosen category exceeds the vigilance; that is:

$$\frac{|I \wedge w_J|}{|I|} \geq \rho \tag{3.2}$$

where parameter $\rho \in [0, 1]$ represents the vigilance level. The number of developed categories can be controlled by the vigilance level. The higher the vigilance level is, the more sensitive the network is to changes in the environment. This can result in a larger number of finer categories. When $\rho = 1$, the network creates a new category for every unique input pattern.

The node J in the F2 layer captures the current input, and the network learns by modifying the weight vector w_J according to:

$$w_J^{new} = \gamma(I \wedge w_J^{old}) + (1-\gamma)w_J^{old} \tag{3.3}$$

where parameter $\gamma \in [0,1]$ is the learning rate. Fast learning occurs when $\gamma = 1$. If the stored prototype w_J does not match the input sufficiently (i.e., Equation (3.2) is not satisfied), the winning node in the F2 layer will be inhibited until a new input vector is applied. Then, another node in the F2 layer is selected with the highest A_j value, and will be matched against the input. This process is repeated until the network either finds a stored node whose value matches the input, or selects the uncommitted F2 layer node if all nodes result in mismatches. In this case, learning a new category is initiated according to Equation (3.3).

The Fuzzy ART system has a category proliferation problem. The category proliferation problem refers to a situation in which the number of classes increases too quickly in a neural network. Complement coding can be used to overcome this problem [Sapojnikova, 2003]. The complement of input I can be achieved by preprocessing each incoming vector a by $a^c \equiv 1-a$. After the complement coding process, input I becomes a $2M$-dimensional vector, $I = (a, a^c) \equiv (a_1, ..., a_M, a_1^c, ...a_M^c)$. Note that normalization of input I can be achieved by preprocessing each incoming vector a, $I = a/|a|$. The input vectors are automatically normalized after being preprocessed into complement coding form.

Labeling the sensory data during the testing phase occurs as follows. Given N distinct category prototypes denoted by $W = \{w_1, \cdots, w_N\}$ learned during the training phase, and a series of T observations $O = \{o_t : t \in T\}$ made over time, where t denotes time, the system finds the closest match of the current observation o_t to a category prototype in w_i. Formally, the formulation is defined as follows,

$$o_t \in w_i \text{ if } i = \arg\max_k P(w_k|o_t) \tag{3.4}$$

where $P(w_k|o_t) < \rho$, which means that if the difference between o_t and the closest match $P(w_k|o_t)$ is less than a vigilance parameter ρ, then the observation o_t is labeled as i. Note that i denotes the index of the category label. If the difference is larger than ρ, then an abnormal alert will be generated. Let c_t denote the category label learned from observations o_t at the time t. There is a one-to-one mapping from observations to category labels. Thus, a sequence of observations, O becomes a sequence of category labels, $C = \{c_t : c_t \in W, t \in T\}$.

3.3 Time models

The baseline Fuzzy ART neural network cannot detect time-related changes. Two time modeling methods have been studied to detect time-related changes. The first model is a heuristic Finite State Automaton (FSA). The environment can naturally be described by environmental states (i.e., categorized by the Fuzzy ART neural networks). The developed FSA model assumes the time duration that the system remains in a state is a Gaussian distribution. The FSA model uses an empirical rule (i.e., three-sigma rule) to identify abnormal time duration in a state. Whenever a transition in or out of a state occurs, the FSA model further checks whether the transition is probable. However, the FSA model poses some limitations. Such as, it is difficult to restart the detection process after an anomaly is detected. In addition, the FSA model assumes that the time duration in each state follows Gaussian distribution. To enhance the FSA model, a likelihood-ratio detector scheme is studied. The system uses a two-step temporal modeling technique to model temporal sequences. It first compresses the temporal sequence into shorter semantic temporal events. Then, the system uses a Probabilistic Suffix Tree (PST) [Mazeroff et al., 2008] to model the temporal semantics. Finally, a likelihood-ratio detector is used to detect time-related anomalies. The likelihood-ratio detection approach is online, fully autonomous, distributed, and data-driven. A detailed description in given in Chapter 4.

3.4 Missing data imputation

It is common to have missing data problems in WSNs. Different ways of estimating missing data have been contrasted and compared in this study. The studies show that utilizing spatial and temporal correlations to impute missing values yields high accuracies. A novel spatial-temporal imputation technique is first developed. The spatial-temporal imputation technique replaces a sensor node's missing values using the most common neighbors' sensor reading or its past sensor reading. However, the spatial-temporal imputation technique poses some limitations. To overcome these limitations, a novel Nearest Neighbor (NN) imputation technique is developed for estimating missing values in WSNs. The NN imputation method organizes the temporal and spatial correlated data into a kd-tree data structure [Hughey and Berry, 2000]. When estimating missing sensor values, the NN imputation technique uses the nearest neighbors found from the kd-tree traversal. One of the traditional kd-tree construction and search methods uses a Euclidean distance metric. Due to the missing values, a weighted Euclidean metric is developed for kd-tree construction and search; the new weighted metric takes into account the missing data percentage of each sensor, which is more suitable for WSNs. A detailed description of the missing data study is presented in Chapter 5.

Chapter 4

Semantic temporal modeling

Wireless Sensor Networks (WSNs) are widely used in environment monitoring applications. For example, a WSN can be used to monitor volcanic activities, thus saving researchers from the long trips and frequent returns to the deployment site. Each sensor in the network can monitor its local region and communicate wirelessly with other sensor nodes to collaboratively produce a high-level representation of the environmental states. With limited power resources, it is important to reduce the amount of data to be transferred through the wireless network. In Chapter 3, a hierarchical, distributed, and modular anomaly detection system is introduced that does not require centralized decision making. The system uses a neural network to perform sensor fusion from multiple sensors. This chapter investigates two time analysis models that are able to detect time-related anomalies. The first time model uses a heuristic Finite State Automata (FSA) to capture some aspects of time modeling. The second one extends the time model work by significantly enhancing the discrete state machine through the use of a more robust, high performance and memory efficient anomaly detection method. This method consists of symbol compression technique that extracts the semantic meaning from the temporal sequence data, along with a Probabilistic Suffix Tree that is a more efficient method for anomaly detection.

4.1 Initial time analysis using Finite State Automata

As the first step, a Finite State Automata (FSA) time model is developed. For each state, it records the average time and the variance of the time spent in a state during the initial training period. This model can detect abnormal transitions by using the transition probabilities; it can detect abnormal time durations spent in a state by using the recorded average time. In this model, the states of the FSA correspond to the class labels c_i that are output from the classifier. At each time increment, the system may either remain in the same state, or move to a new state. Note that in this implementation, the probability of the system remaining in the current state is not maintained; instead, the system retains the average time spent in the state.

In a WSN setting, the FSA is built autonomously during the training phase using Algorithm 1. Sensor motes periodically sense the environment at a fixed rate, and feed the normalized sensor readings to the classifier to build classes/states of the environment. For each state (c_i) in the FSA, an average time and the variance of the time the system remains in that particular state are recorded. Additionally, the state transition probabilities, p_{ij}, from one state to the next set of states are also calculated. Therefore, an alarm will trigger if the amount of time in a state is either too short or too long. In a similar fashion, if a state transition is not probable, then this may also trigger an abnormal alarm. Thus, an anomaly from state transitions and from state occupancy time can be captured.

The algorithm is flexible because the FSA captures both the transition and time duration in a systematic way. If one is only interested in capturing the abnormal state transitions, the algorithm can simply turn off the functionality of calculating the average time spent in a state. It assumes the time duration in each state follows a normal distribution and uses the three-sigma rule to detect anomalies (outliers).

There are some limitations of using this FSA-based model. First, classes are assumed to be independent of each other. In practice, statistical tests have to be performed in order to verify the independence between classes for a given set of learning samples. These tests generally require human involvement, and are not feasible for online applications. Second, it is hard to find an automatic way to determine the restarting state for the FSA model once an anomaly is detected. Because of this limitation, the system is not able to detect

Algorithm 1 Building the FSA

```
1: for each time step do
2:     if the current state is the same as the last time step then
3:         Record the time spent in this state.
4:     else
5:         Record the state transition.
6:     end if
7: end for
8: for each state $i$ do
9:     Find the mean $\mu_i$ and standard deviation $\sigma_i$ of the time the system remains in state
       $i$.
10:    Find the transition probability $p_{ij}$ for each possible state $j$.
11: end for
```

multiple time-related anomalies in the environment during the monitoring period. Finally, this approach assumes the time duration in each state follows a Gaussian distribution. Normality testing is generally required for this approach, or the system may risk having a model mismatch problem.

4.2 The enhanced semantic temporal modeling in WSN

To address some of the issues posed by the FSA model, an enhanced time modelling approach is developed to solve time-related anomalies. In order to model the temporal events with different lengths, a system should be able to model the time duration in each state. Additionally, the system should be able to automate the detection process when multiple time-related anomalies occur. In general, a high order fixed length Markov model is able to achieve this. However, given the constraints of WSNs, it is infeasible to use a traditional high order Markov chain, since an L-th order Markov model requires $|\Sigma|^L$ states, where $|\Sigma|$ denotes the number of alphabet symbols and L is the length of past history/memory being modeled. In addition, Markov models look at all states simultaneously and use a well-known backward/forward algorithm to find the most likely sequence and its probability. Therefore, the problem of finding a restarting point after an anomaly is detected is naturally solved.

For a large variety of time-related sequential data, statistical correlations decrease rapidly with the distance between symbols in the sequence. If the statistical correlations are indeed decreasing, then there exists a "memory" length M such that the empirical probability changes very little if conditioned on subsequences longer than M. Ron et al. [Ron et al., 1996] presented an elegant solution to this problem. The underlying observation in their work is that in many natural sequences, memory length depends on the context and is therefore not fixed. Therefore, as in Ron et al. [Ron et al., 1996], the system developed in this dissertation uses a Variable Memory Markov (VMM) model to preserve the minimal subsequences (of variable lengths) that are necessary for precise modeling of the given statistical source. This results in a more flexible and efficient sequence representation. It is particularly attractive in cases where the system needs to capture higher-order temporal dependencies in some parts of the behavior and lower-order dependencies elsewhere. The VMM model can be implemented in the form of a Probabilistic Suffix Tree (PST) model. A PST is a tree whose nodes are organized such that the root node gives the probability of each symbol of the alphabet while nodes at subsequent levels give next-symbol probabilities conditioned on a combination of one or more symbols having been seen first.

The constructed PST model is essentially a symbolic predictive model: the underlying continuous time signals are first abstracted to a discrete space, analogous to a set of finite categories. In some cases, this has the advantage of being more immune to the problems of noise while still preserving the essential underlying patterns or dependencies that govern behavior in the observed domain. Arguably, it also produces a more understandable model since its components are higher level abstractions. All these advantages of the PST model make it suitable to model temporal sequences in WSNs. One of the challenges of using a PST model on resource constrained sensor nodes is that the model complexity may grow exponentially with the memory, depending on the data. For example, the top part of Figure 4.1 shows the raw seismic sensor readings recorded over a 24-hour time period at Volcano Reventador, collected by Werner-Allen et al., [Werner-Allen et al., 2006]. The Fuzzy ART neural network (described in Chapter 3.2) is used to classify the sensor signals into 13 distinct classes/categories (refer the bottom part of Figure 4.1). The categories are assumed to carry the higher semantic meaning of the data, such as different levels of volcanic activities. Then the system can analyze the sequence of activities to understand their temporal semantic meanings. Observe that in the data of Figure 4.1 there are more than 15 hours of no activity followed by an eruption which lasts less than a minute. An active volcano may have days or months of inactivity followed by an eruption lasting only minutes. A WSN that is designed to model time sequences should be able to model this process and be able to detect eruptions. In many environmental monitoring applications, it is common to have a period of inactivity that extends over several days, weeks, or months. Using PST models directly to model the sensor data is computationally prohibitive. Thus, the system uses a data-driven approach to automatically infer discrete and abstract representations (symbols) of primitive object interactions. These symbols are then used as an alphabet to infer the high level structures of typical interactive behaviors using PST models. The approach used in this research is called the Lempel-Ziv-Welch (LZW) [Welch, 1984] symbol compression method. After compressing the temporal sequences, the system then uses a PST to model the semantic class sequence. The PST model now represents high level semantic notions. These environment feature descriptors are invariant of the sensor classes and time dependencies. They constitute the input to a statistical learning framework where discrete representations of interactive behaviors can be learned by modeling the probability distribution of the feature vectors within the interaction feature space.

Symbol compression methods can extract semantics from temporal sequences. Therefore, the WSNs could save power by transmitting and receiving the compressed temporal sequences. By adding a symbol compression module, the system addressed in Chapter 3 requires some modifications. Like the system architecture presented earlier, it uses a hierarchical learning/communication structure for the WSN. The sensor nodes in the WSN are divided into clusters, as shown in Figure 4.2. Each cluster has a clusterhead and multiple cluster members. Each cluster covers a geometric region and is responsible for detecting the environmental changes in that region. Both cluster members and clusterheads run an identical detection system — a classifier, a symbol compressor, a time analysis model, and a missing data estimator. Cluster members read in a raw sensor signal, o_i, (e.g., light and sound) from the environment as input, and then perform sensor fusion and classify the data into a category c_i. If an anomaly is detected by the classifier, an abnormal alert

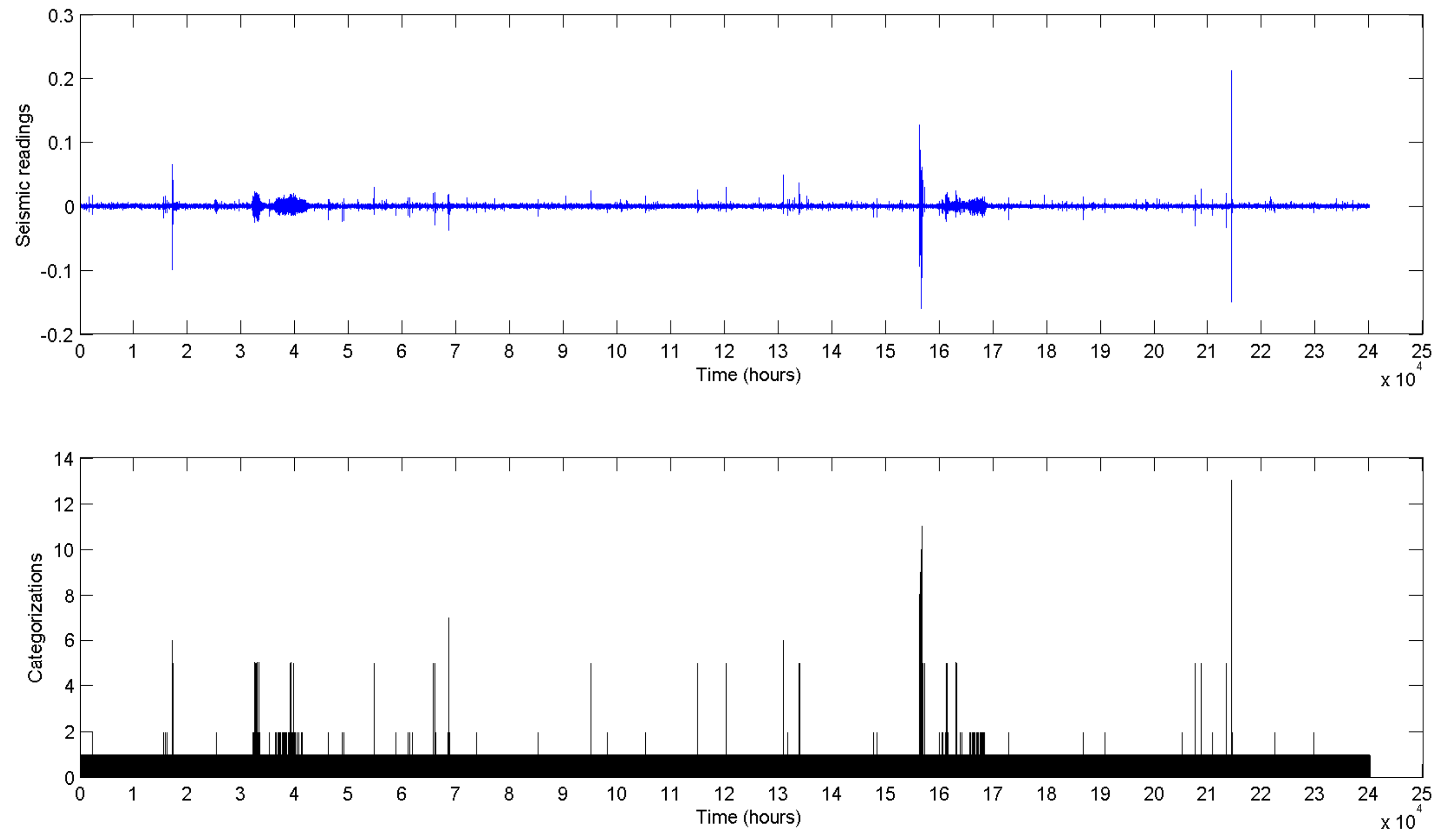


Figure 4.1: Top: raw seismic readings over a time period of 24 hours. Bottom: the Fuzzy ART categorizations/classifications based on the raw seismic readings.

is raised. After the classification process, cluster members compress their category labels and send the compressed category labels s_i (i.e., higher-level semantic symbols in temporal sequences) to their clusterheads. Note that in the intruder detection system discussed in Chapter 6, the cluster member would send the class labels after each time step. By compressing the class label sequence into a shorter temporal sequence, the network is able to save communication costs by removing the redundancies in the temporal sequence. The amount of redundancy removed depends on the compression ratio of the original temporal sequence. Then, the clusterhead analyzes the compressed sequences at a higher level. If an anomaly is detected by the time analysis module, an abnormal alert is raised. Clusterheads often cannot receive complete labels from all of their cluster members, due to unstable wireless communications. Ignoring missing data would result in too many false positives (as justified later in Chapter 5). Thus, the clusterheads first pre-process the collected category labels by identifying and estimating the missing values (described in Chapter 5). The learning system has a hierarchical structure — clusterheads may have higher level clusterheads which classify their output (compressed) class labels. Finally, the root node obtains the final model of the environment. With this learning architecture, the system is able to detect both abnormal environmental changes and time-related changes.

The system uses modular design to give the human operators the flexibility of turning off the modules that are not needed. For example, the operator can turn off the time analysis module if analyzing time is not of interest. In addition, the WSN can be easily scaled to a large numbers of sensors. At the same time, this hierarchical approach reduces communication, which in turn saves energy in the WSN. Specifically, to reduce the amount of data transmitted through the sensor network, the system can transmit the semantic representation of the sensor data instead of the raw sensor signals. In addition, transmitting the semantics from temporal events can further reduce communication costs.

The flow of the sensor signals of an individual sensor node in the new enhanced time model is demonstrated in Figure 4.3. This is a significant extension over the previous FSA design for temporal modeling. Each sensor node takes in multi-dimensional raw sensor vectors from the environment as input, and then performs sensor fusion and classifies the raw sensor signals into categories using a classifier. If the new category does not match one of the existing normal categories, a sensor level anomaly is detected by the

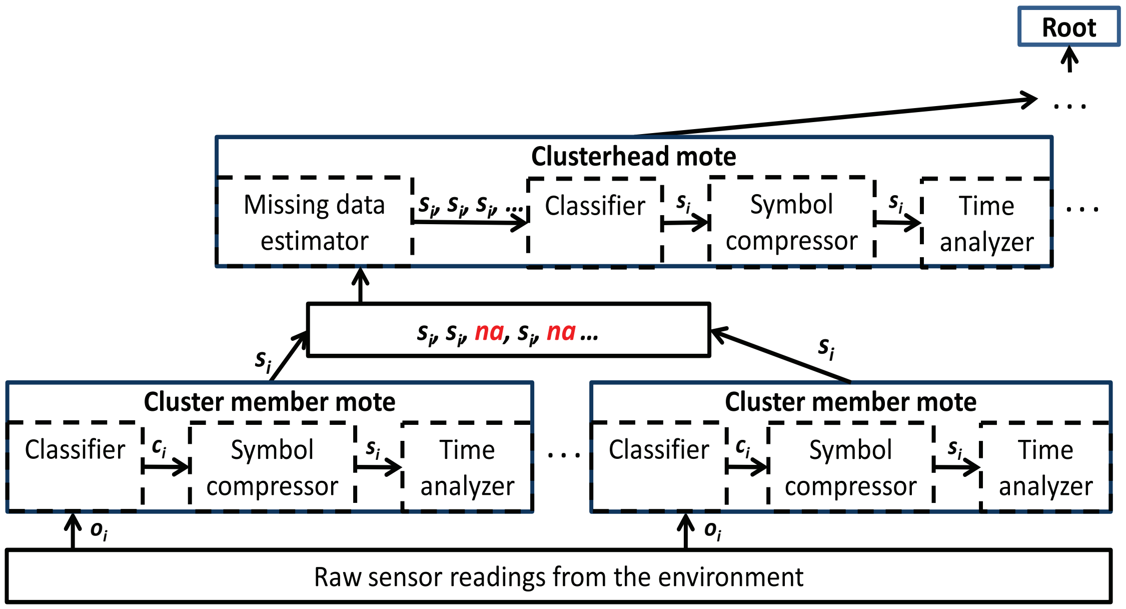


Figure 4.2: The overall sensor network architecture for learning semantic information from temporal sequences.

system. As previously described, the classifier that the system uses is the Fuzzy ART neural network [Carpenter and Grossberg, 1991]. The raw sensor signals are processed through the Fuzzy ART neural network to produce a sequence of meaningful environmental states/classes. After the sensor fusion and categorization process is completed, the system further checks whether there are time-related changes. The temporal sequence model is a two-step process. First, the sequence of classes/categories is compressed with a symbol compressor. Then the compressed events are built into a PST model. In order to detect time-related changes, the system measures the likelihood of the current compressed sequence, compared to the PST model(s) learned during the training phase.

The symbol compressor that the system uses is a lossless, dictionary-based symbol compressor called the Lempel-Ziv-Welch (LZW) [Welch, 1984] encoder. It operates efficiently by sampling the string statistics in a manner that allows a compressed representation and exact reconstruction of the original string. Depending on the implementation, learning and generating can be off-line or on-line, real-time or non-real-time. Note that this two-step structure is developed to make the system more flexible when capturing the temporal sequence information. For example, system designers may substitute another symbol

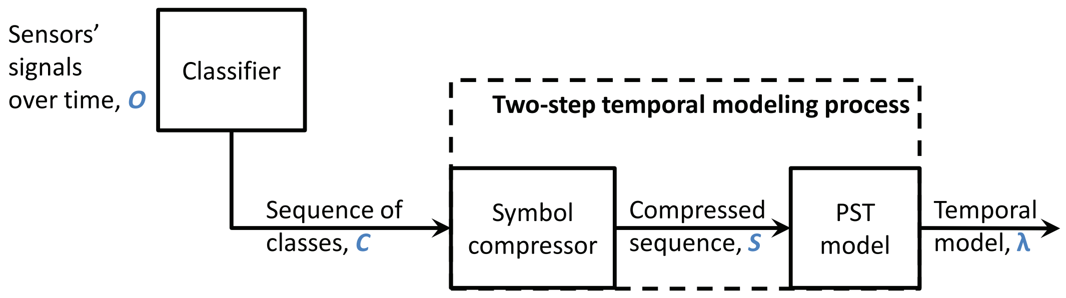


Figure 4.3: Signal processing flow on a single sensor node. The system first uses a symbol compression technique to extract semantics out of temporal sequences. Then, the temporal semantics are modeled using a PST model.

compression method or Markov-based model as desired. When modeling the temporal sequence, if time duration within a state (class) is not of interest, the designers can simply remove all consecutive repeated categories in the symbol compression stage and let the PST model only the state sequences.

The rest of this section is organized as follows. First, Section 4.2.1 addresses the issue of extracting semantic symbols from temporal sequences using the symbol compression module. Next, Section 4.2.2 presents a PST approach for modeling time events. A likelihood-ratio based anomaly detector is presented in Section 4.2.3. Finally, in Section 4.2.4, the approach is tested on a volcano monitoring dataset, and the results are analyzed.

4.2.1 Identifying temporal semantics from sequences

Compression algorithms can roughly be categorized into lossless or lossy compression [Sayood, 2000]. Lempel-Ziv77/78 and Lempel-Ziv-Welch (LZW) are the most widely used dictionary-based compression techniques [Sayood, 2000]. The main mechanism in both schemes is pattern matching: find string patterns that have occurred in the past and compress them by encoding a reference to the previous occurrence. The temporal sequence compression process is defined as follows. The encoder/compression algorithm takes a sequence of category labels $C = \{c_1, \cdots, c_T\}$ and compresses it into another sequence denoted by S, which encodes higher-level semantic meaning.

The developed approach uses the LZW algorithm to construct semantic symbols. LZW is an improved implementation of the Lempel-Ziv78 algorithm. The algorithm is designed

Table 4.1: Dictionary built by LZW algorithm

Dictionary				
Index (code)	Entry (categories)		Index (code)	Entry (categories)
0	1		7	33
1	2		8	31
2	3		9	123
3	12		10	32
4	22		11	2222
5	222		12	21
6	23		13	13

to be fast to implement but is not usually optimal because it performs only limited analysis of the data. With the limited resources of a WSN, it is important to reduce the amount of processing. In a typical compression run, the algorithm takes a string as input, and processes the string using a dictionary of distinct substrings. The algorithm encodes the string (and builds the dictionary) by making a single left to right traversal of a sequence. Initially, the dictionary contains the alphabet with its corresponding encoding. Then it sequentially adds every new substring that differs by a single last character from the longest match that already exists in the dictionary. This repeats until the string is consumed. The idea is that, as the string is being processed, it populates the dictionary with longer strings, and allows encoding of larger blocks of the string at each replacement. For example, suppose there is a source temporal sequence with a three-letter alphabet $\Sigma = \{1, 2, 3\}$, and the temporal sequence that needs to be compressed is C={1, 2, 2, 2, 2, 3, 3, 1, 2, 3, 2, 2, 2, 2, 1, 3}. Based on knowledge about the source sequence C, LZW algorithm builds the dictionary shown in Table 4.1. The output encoded string of S is {0, 1, 4, 1, 2, 2, 3, 2, 5, 1, 0, 2}.

Note that with the developed two-step temporal modeling process, it is not necessary to keep all entries in the dictionary. Especially with limited memory in the wireless sensor nodes, the system can only afford to build a small dictionary. If a system can afford to build a PST with order up to M, dictionary entries with length shorter than M can be pruned, since they can be modeled by an M^{th} order PST. System designers may choose to further prune the dictionary entries based on their knowledge of the application. To further prune the dictionary, system designers may select relevant features based on the application. The relevant features are the dictionary entries with real-world meanings that are relevant to the specified application. The interpretation of the temporal sequence compression process is as follows. Each entry in the dictionary denotes a sub-sequence in the initial temporal sequence to be compressed, while the corresponding index is the "new" compressed temporal symbol with higher semantic meaning. The compressed sequence carries the higher level semantic meaning of the initial temporal sequence. The length of each dictionary entry corresponds to a real-world event in discrete time. Specifically, the length of a single-alphabet entry in the dictionary denotes the time duration of the event's occurrence and the corresponding index carries that semantic meaning. For example, the entry "2222" in Table 4.1 indicates the environment is in state "2" for 4 time units. The corresponding index "11" indicates 4 time units of state "2" in the compressed temporal sequence. Dictionary entries with multiple alphabets may have a real world meaning as well. For example, the entry "12" could correspond in the real world to a person taking one minute to add coffee powder and water into a coffee machine, and then it takes the machine one minute to make a pot of coffee. The whole process is now associated with an index "3" that has the real semantic meaning "making-coffee". Therefore, the compressed temporal sequence is able to carry higher-level semantics of the initial temporal sequence with a shorter length.

4.2.2 Modeling semantic interactions using PSTs

Another significant extension to the previously described FSA approach is modeling semantic interactions using PSTs. The PST model, which was originally designed for classification purposes, has the advantage of improved extraction of statistical information from sequences. The trade-off is that it deliberately throws away some of the original sub-sequences during the analysis process to maintain a compact representation. In resource constrained WSNs, compact data models save energy in sensor nodes by reducing the amount of data being transferred among the nodes. In order to make the system easy to use, the system should build models of the environment that are able to support both interpretation and anomaly detection. This is achieved by using PSTs to efficiently encode the sequences of the categories corresponding to observed interactive behavior in the feature space. In the following, the definition of the PST is given first. Then, an explanation of how the PST is used to detect time-related anomalies in the WSN is described.

The PST is a stochastic model that uses a suffix tree as the index structure. This approach is based on the "memory" of natural sequences. That is, the root node of the PST gives the empirical probability of each symbol in the alphabet while each node at subsequent levels is associated with a vector that gives the probability of the next symbol given the label of the node as the preceding segment. For example,

$P(s_{i+1}|s_0...s_i) = P(s_{i+1}|s_{i-M}...s_i)$, $i > M$,

gives the empirical probability distribution P of the next symbol s_{i+1} given the last M symbols in the preceding segment. Furthermore, a tree of order M has M levels beyond the root.

To illustrate, consider a (compressed) sequence $S = \{1, 2, 3, 1, 2, 3, 2, 1, 3\}$ with a three letter alphabet $\Sigma = \{1, 2, 3\}$. Fig. 4.4 shows the order-2 PST inferred from the observation. Beginning with the root node, which represents the empty string, each node is a suffix of all its children. The probability vector below the nodes gives the conditional next-symbol probabilities. Moreover, all symbols are shown on the same level of the tree. Next symbol "transitions" jump from one branch to another, not from a parent node to its children. This transition pattern is due to the suffix format of the node labels. Some branches typically die out early while other branches propagate to the maximum depth of the tree. Additionally, if a child node carries an identical next-symbol probability distribution as its parent node, the child node is pruned. Detailed PST inference and pruning procedures can be found in Mazeroff [Mazeroff et al., 2008].

Let $S = \{s_u : u \in U\}$ denote a compressed temporal sequence of size U, where u denotes the discrete time unit after compression. To model the normal behavior using the Maximum Likelihood criterion, the system finds a model that maximizes the probability of a given sequence of observations. Given a PST λ, the total likelihood of the observations can be expressed mathematically as $L = P(S|\lambda)$. If the probability of the observation

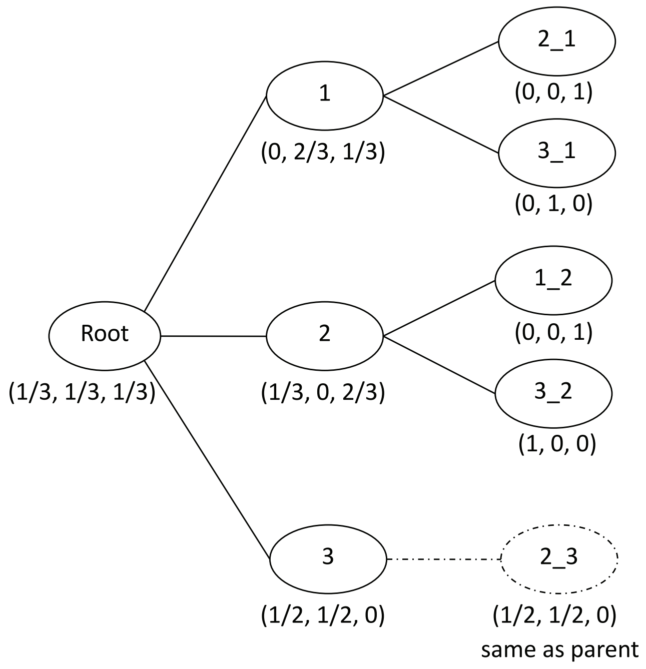


Figure 4.4: An example: an order-2 PST based on a compressed sequence $S = \{1, 2, 3, 1, 2, 3, 2, 1, 3\}$

sequence given the model is below a threshold θ, then an anomaly is detected by the system. A likelihood-ratio detector is addressed in detail in the following subsection.

The computation for this procedure is fast and inexpensive. The PST model has been shown to be implementable in linear time and space [Apostolico and Bejerano, 2000]. Let the length of the training sequence be n, the memory length of PST be M, the alphabet be Σ, and the length of a testing sequence be k. Apostolico and Bejerano's PST building algorithm takes $O(n|\Sigma|)$ time [Apostolico and Bejerano, 2000]. This procedure is a one-time overhead cost to the system during the initial period (unless the PST needs to be updated). To detect anomalies after the PST is constructed, the system has to calculate the likelihood that the testing sequence matches the built PST. The sequence matching procedure is a simple tree traversal, and the detection procedure takes $O(mk)$ time. Thus, it is practical for sensor nodes that have limited resources.

4.2.3 Likelihood-ratio detection scheme

A typical approach to anomaly detection is to build normal and abnormal event models, then classify the query instance into one of these built normal and abnormal classes. However, as previously emphasized this research is interested in detecting anomalies in an unknown environment, without no a priori knowledge of anomalies. The designed system achieves this by using a likelihood-ratio detector; the detection process is described in detail as follows.

Let S denote the entire (compressed) sequence of environmental states $S = \{\tilde{S}, \hat{S}\}$, where $\tilde{S}$ denotes the training sequence, and $\hat{S}$ denotes the target testing sequence. Given a segment of observations/class labels $\tilde{S} = \{s_1, \cdots, s_v\}$, and a segment of observations/class labels $\hat{S} = \{s_{v+1}, \cdots, s_U\}$, the task of detecting anomalies in the sequence $\hat{S}$ is to determine if $\hat{S}$ is the same as $\tilde{S}$. (Note that in some applications the types of anomalous events are known; in that case $\tilde{S}$ can represent abnormal observations, and an anomaly is detected when $\hat{S}$ matches $\tilde{S}$.) The likelihood-ratio detector is similar to that presented by Reynolds, et al., in [Reynolds et al., 2000] for speaker verification problems. The anomaly detection problem is formulated as determining if sequence $\hat{S}$ is different from the normal sequence $\tilde{S}$. The anomaly detection task can be restated as a hypothesis test between:

$$H_0 : \hat{S} \text{ is normal} \quad \text{and} \quad H_1 : \hat{S} \text{ is abnormal.} \tag{4.1}$$

The likelihood ratio test to decide between these two hypotheses is given by:

$$\frac{p(\hat{S}|H_0)}{p(\hat{S}|H_1)} \begin{cases} \geq \theta & \text{accept } H_0 \\ < \theta & \text{reject } H_0 \end{cases} \tag{4.2}$$

where $p(\hat{S}|H_i), i = 0, 1$, is the probability density function for the hypothesis H_i evaluated for the observed sequence $\hat{S}$, also referred to as the likelihood of the hypothesis H_i given the sequence. The decision threshold for accepting or rejecting H_0 is θ. For the temporal anomaly detection, the null and alternative hypothesises use PST models λ. Hence, the PST model for the null hypothesis is denoted as $p(\hat{S}|H_0; \lambda_0)$ and for the alternative hypothesis as $p(\hat{S}|H_1; \lambda_1)$. The likelihood ratio detector is $p(\hat{S}|H_0; \lambda_0)/p(\hat{S}|H_1; \lambda_1)$. Usually, the logarithm of this statistic is used giving the log-likelihood ratio,

$$\Lambda(\hat{S}) = \log p(\hat{S}|H_0; \lambda_0) - \log p(\hat{S}|H_1; \lambda_1) \tag{4.3}$$

During the training period, the system may encounter various event sequences that are normal, abnormal and/or undetermined. The model for H_0 can be estimated using a normal event sequence. However, the model for H_1 is less well defined since it potentially must represent every abnormal situation possible. Since the environment that WSNs operate in are typically unknown, it is not possible to train H_1 with every abnormal situation. Therefore, the universal background model (UBM) [Reynolds, 1997], which uses all hypothesized sequences of events, is more suitable. The UBM in the anomaly detection application is the entire set of training data, which may include normal, abnormal and undetermined event sequences.

4.2.4 Experiment: Monitoring volcano activities

A volcano monitoring dataset is used to evaluate the developed anomaly detection system. The dataset was collected by Werner-Allen, et al., from Harvard University at the Volcano Reventador [Werner-Allen et al., 2006]. The use of WSNs greatly simplifies the data gathering task, because the lack of road access prevents frequent trips to the deployment site. A detailed data collection process is described in [Werner-Allen et al., 2006]. The data used in the following experiments is obtained from one of the seismic stations, which samples the environment at 120 Hz. Since the data is voluminous, one day's worth of data, which is approximately 200 megabytes, has been analyzed. This data is from a single sensor node, rather than from a complete sensor network. The following experiments do not involve clusters or clusterheads. The detection process for the WSN is described in Section 4.2. In addition, an intruder detection system with multiple layers of clusters of sensor nodes, with each sensor node using multiple sensors is demonstrated in the lab environments (refer Chapter 6 for details). The intruder detection system uses heuristic discrete states to model time. Since the two-step temporal modeling approach developed here also uses discrete state representations, this should work with the existing system architecture when being implemented on the nodes. The current approach is more robust when detecting multiple anomalies in the environment and involves less communication by transmitting compressed temporal sequences. The volcano dataset here is used as proof-of-concept to illustrate how the enhanced time modeling module works on a real world dataset.

Preprocessing

The top part of Figure 4.1 presents the raw seismic readings recorded over a 24-hour time period on August 1, 2005. The raw seismic data O is normalized and categorized by the Fuzzy ART neural network as shown in the bottom part of Figure 4.1. The raw sensor data O is classified into $|\Sigma| = 13$ categories by setting the vigilance parameter ρ of the Fuzzy ART system to 0.93. Hence, the output temporal categorization sequence C has an alphabet size $|\Sigma|$ of 13. Note that the physical meaning of these 13 categories is unknown. In future work, if the ground truth of the seismic data is available, the Fuzzy ART neural network can assign more meaningful categories (e.g., magnitudes of eruptions) by adjusting its vigilance parameter ρ. The data is grouped into hour-long subsets. Based on visual inspection, there are no volcanic activities during the period between the 1^{st} hour and the 15^{th} hour; this period is regarded as the normal period. There is an anomaly/eruption between the 15^{th} hour and the 16^{th} hour. Thus, this hour is regarded as an abnormal period. It is not possible to determine whether the periods following the eruption are "normal after eruption" or "abnormal". As a result, the data after the 16^{th} hour is discarded. Due to the limited amount of the available "abnormal" data, and since "normal" and "abnormal" is relative. Therefore, hour 16 is treated as the "normal" period and hours 1-15 are treated as "abnormal" periods. The following experiments intend to demonstrate the usage of the developed techniques on real application data.

Performance metrics

The following performance metrics are used to evaluate the anomaly detection system: compression ratio, True Positive Rate (*TPR*), True Negative Rate (*TNR*), False Positive Rate (*FPR*), and False Negative Rate (*FNR*). The compression ratio refers to the ratio of the size of data before compression and the size of data after compression, i.e., uncompressed data/compressed data. The *TPR* or sensitivity is defined as the fraction of positive examples predicted correctly by the model, i.e., $TPR = TP/(TP + FN)$, where True Positive (*TP*) corresponds to the number of positive examples correctly predicted by the model and False Negative (*FN*) denotes the number of positive examples wrongly predicted as negative by the model. Similarly, the *TNR* or specificity is defined as the fraction of negative examples predicted correctly by the model, i.e., $TNR = TN/(TN+FP)$, where True Negative (*TN*) corresponds to the number of negative examples correctly predicted by the model and False Negative (*FP*) denotes the number of negative examples wrongly predicted as positive by the model. Finally, The *FPR* or false alarm rate is the fraction of negative examples predicted as positive class, i.e., $FPR = FP/(TN + FP)$, while the FNR or miss rate is the fraction of positive examples predicated as a negative class, i.e., $FNR = FN/(TP + FN)$. Ideally, the values of sensitivity and specificity are at 100%; the values of false alarm rate and miss rate are at 0%.

PST vs. Fixed length Markov model

The performances of the PST model and the traditional fixed length Markov model are evaluated on the volcano monitoring dataset. In this experiment, the "normal" period (hour 16) category sequence is used as training data, and "abnormal" periods (hours 1-15) category sequences are used as testing data. The performance is measured by the negative log-likelihood of the normal category sequence given the observation of the abnormal category sequence. Specifically, the system has constructed both PST and fixed length Markov models from the training data with Markov orders 1, 2, 3, 4, 5, and 10. For each PST/fixed length Markov model, the negative log-likelihood $P(C|\lambda)$ of the testing sequence C given the PST/fixed length Markov model λ is obtained. The larger the negative log-likelihood value is, the more dissimilar are the compared sequences. The hypothesis is that the dissimilarity between the normal and the abnormal period grows as the memory order grows.

The experimental results are summarized in Table 4.2. The empirical results indicate that the sizes of PST models are much smaller than the traditional fixed length Markov models as the order increases. For example, observe that the 10^{th} order fixed length Markov model uses 981 states, while the order-10 PST model only uses 205 states. The negative log-likelihood is the same between a sequence given a PST model and a fixed length Markov model with the same order, since the system eliminates nodes that have the

Table 4.2: Comparison of fixed length Markov models versus PST models in terms of number of nodes and negative log-likelihood

	Model Order	Number of Nodes	Negative Log-Likelihood
Fixed length Markov	1	12	−0.0141
	2	45	−0.0112
	3	104	−0.0092
	4	190	−0.0078
	5	296	−0.0069
	10	**981**	−0.0046
PST	1	12	−0.0141
	2	41	−0.0112
	3	76	−0.0092
	4	116	−0.0078
	5	146	−0.0069
	10	**205**	−0.0046

same probabilities as their parent nodes when constructing the PST models. The model is lossless in terms of capturing the information of the training data. Therefore, a PST model is preferred over a fixed-length Markov model because it is purely data-driven, it is flexible, and most importantly, it takes less space. Note that the PST models can be pruned to remove some low probability nodes (see [Ron et al., 1996]); however, this will lead to information loss. In future work, the detection system could be evaluated based on PST models with thresholds to remove low probability nodes. Investigation would be needed to determine the threshold values for these PST models in the developed detection system.

PST model with compressed temporal sequence

As shown in the previous subsection, using the PST model directly on the volcano monitoring dataset is still impractical for the resource limited sensor nodes; that is, it takes 205 states to build a PST model with a memory of 10 observations. An observation with 10 samples can hardly capture any meaningful sequences in the environment monitoring type of applications. In reality, daily life activities usually take more than 10 observations to capture. If modeling such a long sequence of actions is important to an application, the PST model will not be a practical solution if used directly on the raw samples. A practical two-step solution is developed to solve this problem. First, the temporal sequences are compressed into a higher level temporal symbol representation. After that, the PST model is built from the higher level symbol representation of the original sequence.

The system uses the standard LZW symbol compression on the category sequences of the dataset. The average dictionary size for 16 hours (excluding the pre-built characters in the dictionary) is approximately 61 entries per hour. Table 4.3 shows some sample entries. Based on the given data, one can observe that there are long periods of inactivity, which are denoted by sequences of category "1". The system compresses all 16 hours of temporal sequences independently. The average compression ratio for the 16 hours is approximately 33:1. The compressed representation of data saves on both processing power and reduces the amount of storage on the sensor nodes. Most importantly, when the local sensor nodes transmit the temporal models to the clusterheads, the WSN is able to save the transmission power as well.

To detect anomalies in the temporal sequence, a likelihood-ratio detection scheme as given in Equation (4.2) is employed. The detection procedure works as follows: during the training period, semantics (compressed symbols) from hours 1-16 are used to obtain the alternative hypothesis $p(S|H_1)$, and hour 16 is used to obtain the null hypothesis $p(S|H_0)$. In order to determine the detection threshold θ, a Receiver Operating Characteristic (ROC) graph can be used. ROC is a graphical plot of the TPR/sensitivity vs. the FPR/false alarm rate for a binary classifier system as its discrimination threshold is varied. Two PST models have been built for the dataset with orders 5 and 10. Figure 4.5 shows the ROC curve for the order-5 PST model. Figure 4.6 shows the ROC curve for the order-10

Table 4.3: Partial dictionary of the volcano data

Entry (categories)
...
155
6115
5122
21211
1111111122
211111115
51511
111112111111
11111111111111111111111111111...111111111115
51111111
...

PST. Note that each prediction result (or one instance of a confusion matrix) represents one point in the ROC space. The best possible prediction method would yield a point in the upper left corner representing no false negatives and no false positives. As shown in Figure 4.5, the optimal operating point for the order-5 PST model is with a *TPR* of 84% and a *FPR* of 14%. As shown in Figure 4.6, the optimal operating point for the order-10 PST model is with a *TPR* of 93% and a *FPR* of 4%. The ROC curves also show that order-10 PST model has better operating point than order-5 PST model. This implies that longer memories can better discriminate between normal and abnormal sequences. The optimal operating points are used as references to choose likelihood-ratio threshold θ values in Equation (4.2). Table 4.4 shows the confusion matrix obtained when adjusting the threshold θ. Based on the detection performances, the *FPR* for the order-5 PST model is 16.8% when $\theta = 0$, which is the closest value to the order-5 PST ROC optimal point. The *FPR* for the order-10 PST model is 3.4% when $\theta = 1.1$, which is the closest value to the order-10 PST ROC optimal point. Therefore, θ values of 0 and 1.1 are chosen for PSTs of orders 5 and 10, respectively. In addition, the miss rates for PST orders 5 and 10

Table 4.4: Performance results for PST orders 5 and 10 with compression

PST order	Threshold θ	FNR (Miss rate)	FPR (False alarm rate)
5	**0**	**17.5%**	**16.8%**
	0.25	0%	91.5%
	0.5	0%	85.9%
	0.9	8.8%	34.9%
	1.1	22.2%	10.7%
	1.5	99%	1.8%
10	0	1.8%	6.8%
	0.5	0%	77%
	0.9	0%	17.6%
	1.1	**2.3%**	**3.4%**
	1.5	97.7%	0.9%

PST models are 17.5% and 2.3%, respectively. The miss rates and false alarm rates are relatively low for both PST models. This indicates the ROC curves provide good reference when choosing the values for threshold θ. The tree sizes for the orders 5 and 10 PST models are 186 and 241 nodes, respectively. Note that the nodes on PST models that are built from compressed sequences represent higher semantic temporal meanings. Therefore, the nodes represent much longer observations compared to the nodes on the PSTs that are built from the uncompressed sequences, (i.e., the order 10 PST is modeling a sequence with approximately 330 observations compared to 10 observations). The detection results show that the developed PST model with symbol compression method is able to detect anomalies with high performance. Additionally, the UWB-based likelihood ratio detector is robust and able to detect multiple anomalies in a time sequence with high performance. The robustness and the ability of detecting multiple anomalies are significant enhancements from the previous heuristic Finite State Automata approach.

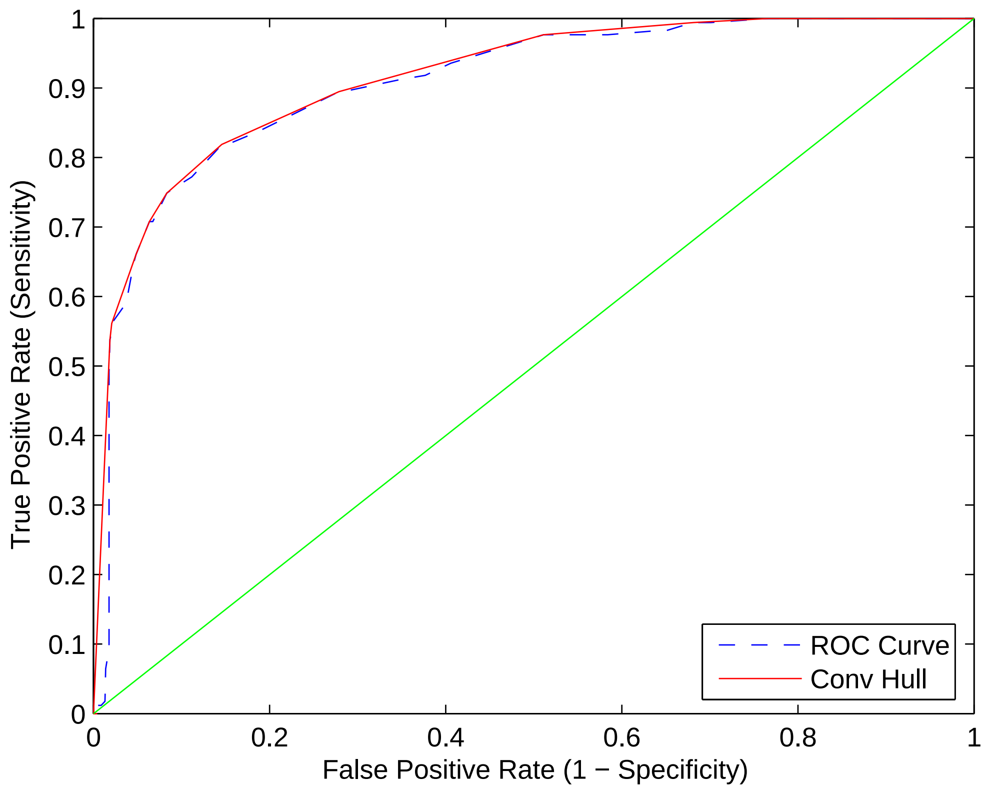


Figure 4.5: ROC curve for order-5 PST model. The optimal operating point for the order-5 PST model is with a *TPR* of 84% and a *FPR* of 14%.

4.3 Summary

This study has explored two temporal modeling techniques for detecting time-related changes in WSNs. The first approach is an heuristic Finite State Automata (FSA). The FSA models the time duration in each state as a normal distribution and uses an empirical rule to detect time duration anomalies. In addition, the FSA model records state transition probabilities. If an event sequence is not probable, an anomaly is detected. This approach is simple and flexible. However, because the FSA model assumes that time duration in each state follows a normal distribution, it is difficult to find a restarting point when multiple time-related anomalies occur. To overcome these limitations, an enhanced time modeling technique that uses the likelihood-ratio detection method is used. In the likelihood-ratio method, the system first extracts the temporal semantics using the Lempel-Ziv Welch

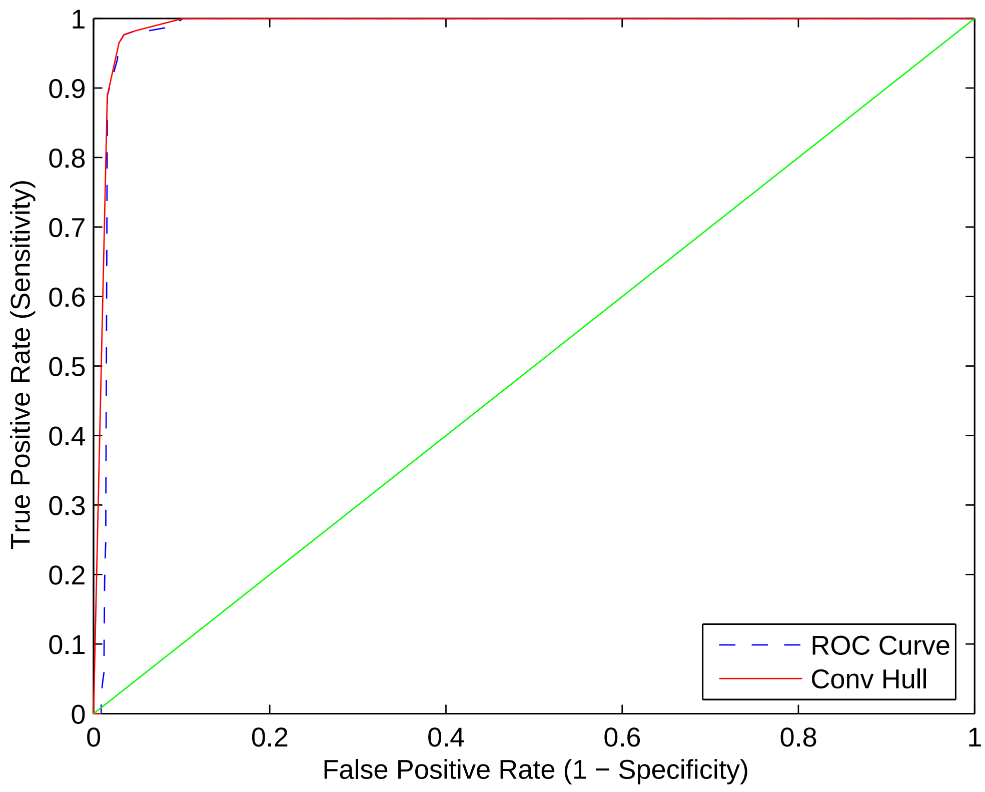


Figure 4.6: ROC curve for order-10 PST model. The optimal operating point for the order-10 PST model is with a *TPR* of 93% and a *FPR* of 4%.

(LZW) encoding method. Then, it uses a Probabilistic Suffix Tree to model the semantic sequences. Finally, the system uses a Universal Background Model likelihood-ratio detector to detect time-related changes. This two-step time modeling technique is suitable to enable resource constrained sensor nodes to perform time analysis. Additionally, the LZW compression method allows the system to transmit temporally compressed data across the WSN, resulting in energy savings. One major advantage of using the UBM likelihood-ratio detector is that it requires no prior knowledge of the anomalies. In summary, the developed temporal anomaly detector is autonomous, distributed, data-driven, modular, and requires no a priori knowledge of anomalies. All of the above characteristics are suitable for resource constrained WSNs.

Chapter 5

Missing data estimator

Wireless Sensor Networks (WSNs) are widely used for monitoring environments. The main challenge with WSNs research is to determine a systematic procedure to train these networks so that they are sensitive only to “unique” events that are of interest in specific applications. To address this challenge, machine learning techniques are incorporated into a WSN so that the networks can automatically learn to recognize normal and abnormal modes of operation. However, satisfactory performance of the WSNs is not attainable when too many of the sensor readings are missing. Missing sensor readings may be caused by synchronization problems, sensor faults, sensor power outage, communication malfunctions, malicious attacks, packet collisions, signal strength fading, or environmental interference (e.g., a microwave, walls, or human blockage). When decisions have to be made using machine learning techniques, missing data becomes a hindering factor because most machine learning techniques rely on complete data to maintain high performance. However, unstable wireless communications may create many false positives in detection applications. Imputation procedures that estimate missing sensor readings are generally required. Estimation and replacement of the missing data is generally performed before the use of machine learning techniques. Thus, better estimation of the missing data should improve the performance of machine learning techniques and the overall system in general. These estimation technique is important in sparse WSNs where every sensor reading is critical and/or with limited amount of training data and the system cannot afford to lose any information. Many missing data imputation methods have been developed in the literature, including Nearest Neighbors (NN) imputation [Chen and Shao, 2001], [Rancourt, 1999], Bayesian based imputation techniques [Jr. et al., 2007], [Amelia II, 2010], regression-based imputation [Fox, 2008], and so forth.

This chapter is organized as follows: First, initial findings about missing data in WSNs are presented in Section 5.1. Then, a nearest neighbor imputation technique is presented in Section 5.2. Finally, Section 5.3 summarizes the missing data imputation technique.

5.1 Missing data in WSNs

Incomplete data is common in WSNs and experimental findings show that, on average, approximately 40% of data from the sensor nodes are missing when presented to the clusterheads' classifier. The existence of missing data degrades the performance of the classifier.

A formulation of the missing data problem in a WSN is as follows. Let c denote the class decision made by an arbitrary classifier F; t denotes time and i denotes the sensor node ID number. Let s_j denote observations made by sensors j on each sensor node i. For each cluster member node i at time t, the node takes the raw observation s_j and maps it into a class c_i, i.e., $F(s_j^t) \longrightarrow c_i^t$. For each clusterhead node i at time t, the node takes a vector of categories c_j from its cluster members and maps it into a higher level class c_i, i.e., $F(c_j^t) \longrightarrow c_i^t$. $c_i^t =$ "NA" means the category of node i at time t is missing. The data is assumed *missing at random* (MAR); note that this assumption does not mean that the pattern itself is random, but rather the probability of the data missing is independent of the data values.

Due to the limited computational resources on the sensor nodes, the missing data imputation techniques should be simple and memory efficient. However, there are many possible simple strategies for imputation, and it is not clear in advance which strategy might be best for the Fuzzy ART classifier. The hypothesis is that they do not have the same performances. Therefore, a novel spatial-temporal imputation strategy is developed. In addition, this dissertation compares and contrasts several alternative strategies.

5.1.1 Existing techniques

Several missing data replacement strategies, such as those listed in Table 5.1, have been used to improve the performance of the classifiers. The first strategy is called "doing nothing". When new sensor data comes in, the new sensor data replaces the old data. If the sensor values are missing, the system keeps the last available data. Specifically, for received sensor observations a, if a_i^t="NA", let $a_i^t = b_i^{t-1}$, where b is the processed input used for the Fuzzy ART algorithm, t is the current time instance and $t-1$ is the previous time instance. This strategy is used as the base strategy in comparison to the other missing data strategies. The computational and space complexities of this imputation strategy for each node are $O(1)$.

Strategies 2 and 3 use a fixed constant to replace the missing data. Specifically, strategy 2 uses the minimum non-existing value (i.e., 0) and strategy 3 uses the maximum non-existing value (i.e., 1). "Non-existing values" means that the missing sensor readings or class labels are numbered as 0 or 1. The Fuzzy ART algorithm remains unchanged.

Strategies 4 and 5 use a moving average to replace the missing data. Specifically, strategy 4 uses the mean of the past 5 sensor values used, including the estimated missing values. Strategy 5 uses the mean of the past 5 seen observations of the current sensor i, excluding the processed missing values. Note that strategy 4 is different from strategy 5 when the past 5 sensor values contain data that are estimated. The computational and space complexity of both strategies for each node are constant, therefore, $O(1)$.

Strategies 6, 7 and 8 are designed by Granger, et al., in [Granger et al., 2000]. These three strategies were implemented exactly as designed. Strategy 6 replaces the missing data with a fixed minimum value (like strategy 2). In addition, strategy 6 modifies the original Fuzzy ART system by setting the complement coding of the missing data to the minimum value as well. Strategy 7 replaces the missing data with a fixed maximum value (like strategy 3). The complement coding of strategy 7 is also set to the fixed maximum value. However, as the number of prototypes grows, it becomes harder and harder to pass the vigilance testing. Thus, strategy 7 changes the vigilance testing rule to $\frac{|a_i \wedge A'|}{M} \geq \rho$. Instead of using fixed constant values for missing data, strategy 8 uses an indicator vector. When a sensor value is missing, the indicator sets the missing data to "0" and the complement coding to "1". The new prototype choice become, $T_j(A,\delta) = \frac{|w_j \wedge A' \wedge \delta|}{\alpha + |w_j \wedge \delta|}$. The new vigilance level testing become, $\frac{|w_j \wedge A' \wedge \delta|}{\alpha + |w_j \wedge \delta|} \geq \rho$, and the new prototype learning is $w_J^{'} = \beta((A \vee \delta^c) \wedge w_J) + (1-\beta)w_J$. The computational and space complexities of these strategies are $O(1)$ for each node. Since these strategies only modify the vigilance testing rules and/or learning rules from the original Fuzzy ART algorithm, there are no additional time and space requirements for these strategies.

To further test the imputation strategy, the standard missing data imputation strategy from the literature of statistics, i.e., the Expectation Maximization (EM)-based imputation strategy, was also applied to the same data. It is important to note that this EM-based imputation strategy is computationally intensive and, thus, is not practical in a WSN. However, it is considered here for comparison purposes.

Little and Rubin [Little and Rubin, 1986] provide a detailed explanation of the EM algorithm. The EM algorithm is an iterative procedure that finds the Maximum Likelihood (ML) estimation of the parameter vector by repeating the following steps:

- The E-step (expectation step): Given a set of parameter estimates, such as a mean vector and covariance matrix for a multivariate normal distribution, the E-step calculates the conditional expectation of the complete-data log-likelihood given the observed data and the parameter estimates. Specifically, let $\theta^{(t)}$ be the current estimate of the parameter θ. The E-step of EM algorithm finds the expected complete-data log-likelihood if θ were $\theta^{(t)}$:

$$Q(\theta|\theta^{(t)}) = \int L(\theta|y) f(Y_{mis}|Y_{obs}, \theta = \theta^{(t)}) dY_{mis} \tag{5.1}$$

  where observation $Y = (Y_{obs}, Y_{miss})$, Y_{obs} represents the observed part of Y, and Y_{mis} represents the missing part of Y.

- The M-step (maximization step): Given a complete-data log likelihood, the M-step finds the parameter estimates to maximize the complete data log-likelihood from the E-step. The M-step of EM algorithm determines $\theta^{(t+1)}$ by maximizing the expected complete-data log likelihood:

$$Q(\theta^{(t+1)}|\theta^{(t)}) \geq Q(\theta = \theta^{(t)}), \text{for all } \theta \tag{5.2}$$

Table 5.1: Comparison of missing value imputation strategies. The time and space complexities are calculated based on a single node, where n is the number of observations made from time 1 and k is the number of the node's neighbors.

	Imputation strategies	**Time**	**Space**
1. Doing nothing	If a_i^t=“NA”, set $a_i^t = b_i^{t-1}$, where b is the processed input used for the Fuzzy ART algorithm. t is the current time instance and $t-1$ is the previous time instance.	$O(1)$	$O(1)$
2. Replace by the minimum value	If a_i=“NA”, set $a_i = 0$ and complement coding $a_i^c = 1 - a_i$. Note: Neither 0 or 1 are used as normal category numbers.	$O(1)$	$O(1)$
3. Replace by the maximum value	If a_i=“NA”, set $a_i = 1$ and $a_i^c = 1 - a_i$.	$O(1)$	$O(1)$
4. Replace by 5 average used	If a_i=“NA”, set $a_i^t = \frac{(b_i^{t-1}+...+b_i^{t-5})}{5}$.	$O(1)$	$O(1)$
5. Replace by 5 average seen	If a_i=“NA”, set $a_i^t = \frac{(a_i^{t-1}+...+a_i^{t-5})}{5}$.	$O(1)$	$O(1)$
6. Replace by “0” [Granger et al., 2000]	If a_i=“NA”, set $a_i = a_i^c = 0$.	$O(1)$	$O(1)$
7. Replace by “1” [Granger et al., 2000]	If a_i=“NA”, set $a_i = a_i^c = 1$. New vigilance testing becomes $\frac{\lvert a_i \wedge A' \rvert}{M} \geq \rho$.	$O(1)$	$O(1)$
8. Replace by indicator vector [Granger et al., 2000]	If a_i=“NA”, $\delta_i = 0$; otherwise $\delta_i = 1$. $(\delta = (\delta_1, \delta_2, ..., \delta_{2M}))$. New prototype choice becomes $T_j(A, \delta) = \frac{\lvert w_j \wedge A' \wedge \delta \rvert}{\alpha + \lvert w_j \wedge \delta \rvert}$. New vigilance testing is $\frac{\lvert w_j \wedge A' \wedge \delta \rvert}{\alpha + \lvert w_j \wedge \delta \rvert} \geq \rho$. New prototype learning is $w_J^{'} = \beta((A \vee \delta^c) \wedge w_J) + (1-\beta) w_J)$.	$O(1)$	$O(1)$
9. EM imputation	See Equations (5.1) - (5.2).	$O(kn)$	$O(n)$
10. Replace by spatial-temporal data	See Algorithm 2.	$O(k)$	$O(1)$

These two steps are iterated until convergence. In the iterations, the observed data log-likelihood is non-decreasing at each iteration. Initially, the missing values are filled in with a guess which is estimated from available observed data. The E-step and M-step iterate until the maximum change in the estimates from one iteration to the next does not exceed a threshold.

5.1.2 Spatial-temporal imputation technique

Existing missing data imputation strategies do not take into account spatial-temporal information. The hypothesis is that if an environment is highly correlated in time and space, which could be true in sufficiently dense WSNs, using spatial and temporal information to estimate missing observations should have high accuracy. To confirm this hypothesis, one has to show that the environment is highly correlated both in time and space. To do this, this dissertation has utilized techniques from statistics, i.e., space and time correlation testing.

Two time-series tests are used to determine if the data are correlated in time, namely, the Durbin-Watson test and the Partial AutoCorrelation Function (PACF). The Durbin-Watson test determines whether or not the data set is time correlated, and the PACF gives us information on how the sensor data are correlated in time with each other. The value of the Durbin-Watson statistic lies in the range of [0, 4]. A value of 2 indicates that there appears to be no autocorrelation. If the Durbin-Watson statistic d is substantially less than 2, there is evidence of positive serial correlation. On the other hand, large values of d indicate that successive error terms are, on average, much different in value from one another, or are negatively correlated.

For space correlation testing, the Pearson correlation coefficients and R^2 testing are used. The Pearson correlation coefficient is a common measure of the correlation between two random variables X and Y. Pearson's correlation reflects the degree of association between two variables. Its values range from -1 to $+1$. A correlation of $+1$ means that there is a perfect positive association between variables, while a correlation of -1 means that there is a perfect negative association between variables. A correlation of 0 means there is no linear relationship between the two variables. R^2 is a statistical measure of how

Algorithm 2 Spatial-temporal imputation approach

```
1: for each missing input sensor $s_i$ do
2:     for all sensors $s_j$ within one hop of communication range of $s_i$ do
3:         if the sensing distance between $s_i$ and $s_j < \alpha$ then
4:             $a_i^t$ = the most common reading of $a_j^t$
5:         else
6:             $a_i^t = a_i^{t-1}$
7:         end if
8:     end for
9: end for
```

well a regression line approximates data points. R^2 is a descriptive measure between 0 and 1. An R^2 value of 1.0 indicates a perfect fit. Therefore, the closer it is to 1, the better the model.

In order to deal with the missing data, in addition to the time and space replacement models, different ways of modifying the Fuzzy ART algorithm that require only minor changes to the current Fuzzy ART network were studied. A simple replacement algorithm is needed which runs online and still has satisfactory performance. Thus, Algorithm 2 to estimate the missing data was developed. The developed missing data estimator first checks if a neighbor is within the missing sensor's minimum sensing range. If there are neighbors within the sensor's range, the neighbor's observation can be used, resulting in a spatially correlated replacement. If there are multiple neighbors within the sensor's range, and they do not have the same readings, the majority reading is chosen. Otherwise, use the last seen sensor-temporal reading. This strategy is employed because in WSNs, sensor data in the environment tends to be highly correlated for sensors that are geographically close to each other (spatially correlated), and also highly correlated for a period of time (temporally correlated).

The computational requirement for an individual node to estimate missing data using this algorithm is a linear function of the number of nodes, k, within the communication range. In this application, k is also the number of cluster members within one cluster. Therefore, the computational complexity for the developed imputation technique is $O(k)$.

Let m denote the number of nodes in a WSN, R denote the radius of the WSN, and S denote the communication range or sensing range of each sensor. In this problem, The assumption is that the sensing range is the same as the communication range and all sensors have the same communication/sensing range. If R is much larger than node i's communication range S, then k is much smaller than m. In the worst case, $k = m$. In general, sensor nodes are deployed such that each of them covers a local region, and together they cover the entire environment. Thus, searching within a local cluster of k nodes is typically not computationally intensive. The space requirement for the developed approach is $O(1)$, since only one previous observation needs to be stored.

5.1.3 Experiments

In this section, experimental results of the developed missing data technique and nine other missing data techniques are compared and contrasted using real world data and artificially generated data.

Performance metrics

Accuracy is used as the performance measure in these studies. A mismatch between the Fuzzy ART algorithm's categorization and the true category is considered an error. Accuracy (A) is defined as:

$$A = C/T \tag{5.3}$$

where C denotes the number of correct categorizations, and T denotes the total number of observations. To ensure a fair comparison, the vigilance parameter of the Fuzzy ART algorithm has been readjusted for each replacement strategy until the best performance is obtained.

In the conducted experiments, the developed technique is compared against other techniques. To determine the significance of the difference in the results, the Student's T-test is applied. The assumption of the test is that the underlying distributions of accuracies/errors are Gaussian, because of the Central Limit Theorem — as the number of testing sets approaches infinity, the distribution of the mean of accuracy/error approaches a Normal distribution.

Table 5.2: The partial autocorrelations

Lag	1	2	3	4	5
Correlation	0.997	-0.002	-0.002	-0.002	-0.002

Missing data experiment with real sensor data

In this experiment, a network of five cluster members and one clusterhead are used to test the approaches. The five cluster members are uniformly deployed around the clusterhead. The cluster members use light and microphone sensors, and the clusterhead uses the cluster members categories as input. In addition, all cluster members are within communication range of the clusterhead. Experimental results are obtained from three sets of trials. In each trial, each sensor node has made 6500 observations. For testing time and space correlations, only the first trial of collected data is used, since the other two trials repeat the first trial and the environment settings do not change. For the purposes of correlation testing, samples with missing values are removed. All testing results have been made from a data set of approximately 1500 samples with no missing values. The sensory data under the lab setting passed the Durbin-Watson test with a value of 0.0059 with 99.5% confidence level. A DW value of less than 2 indicates there is a high correlation in time. The DW value obtained from the lab setting is near 0, which is evidence that the sensory data does have time correlation.

The next step is to determine the sensory data's time correlations using partial autocorrelations. The partial autocorrelations are shown in Table 5.2. The result shows that the sensor data has high correlation with one previous data point, i.e., lag 1 value is close to 1; however, there is little association with 2 or more sensory observations made in the past, i.e., low lag 2-5 values.

To show space correlation, the correlation coefficients between the sensors at each observation as well as R^2 are calculated. The Pearson correlation coefficients between nodes are shown in Table 5.3. The correlation coefficients between sensor nodes are close to 1. Therefore, there are high positive associations between sensor nodes.

This study further tested the goodness of fit of the model of one sensor's observations replaced by other sensor's observations. As an example, the entire observations made by

Table 5.3: Pearson correlation coefficients

	sensor1	**sensor2**	**sensor3**	**sensor4**	**sensor5**
sensor1	1.0	0.998	0.997	0.995	0.995
sensor2	0.998	1.0	0.998	0.996	0.997
sensor3	0.997	0.998	1.0	0.996	0.998
sensor4	0.995	0.996	0.996	1.0	0.996
sensor5	0.997	0.997	0.998	0.996	1.0

sensor 1 are used to against the entire observations made by sensor 2 to obtain the R^2 value. The R^2 value is almost perfect (close to 1), which means if sensor 1's reading is used to replace sensor 2's reading when sensor 2's observation is missing, it should result in high accuracy, due to the model fitness being high. The sensory data under this setting passed both the time and space correlation tests, therefore, the sensory data is highly correlated in time and space.

The averaged accuracy and standard deviation of different imputation techniques are shown in Figure 5.1. The "do nothing" (strategy 1) is used as a baseline for comparison. To determine the significance of these results, the Student's T-test was applied to the accuracy results for the spatial-temporal scheme compared against other imputation strategies. This test confirms that the differences in these results are statistically significant, with a confidence level of 99.5%. The spatial-temporal imputation strategy outperformed the other strategies. The results suggest that the spatial-temporal imputation strategy (strategy 10) and EM (strategy 9) imputation work better than the "do nothing" as well as other replacement strategies. This is due to the fact that this application involves continuous data gathering for large scale and distributed physical phenomena using a dense wireless sensor network which results in high correlation both in time and space. If the nodes are densely deployed, the readings from nearby nodes are likely to be highly correlated and hence contain redundancies. By using this combination of spatial and temporal replacement for the missing input, the system is able to achieve good performance with relatively low computational cost.

The "do nothing" strategy has a better performance than moving average 5 (strategies

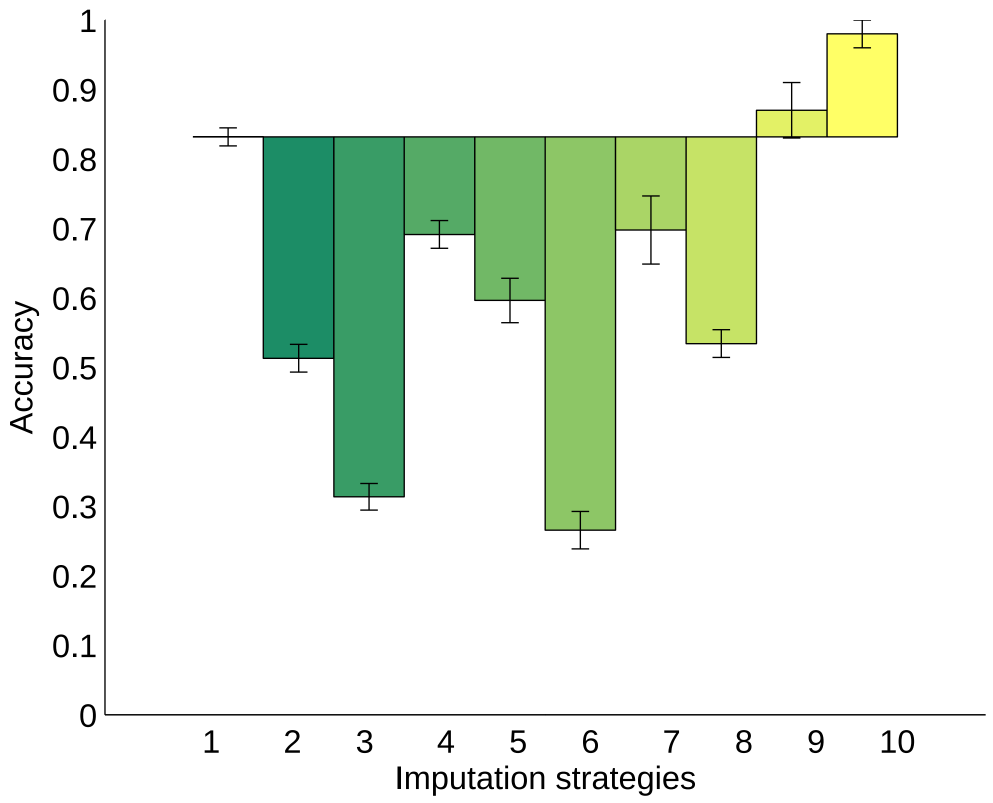


Figure 5.1: Accuracies of different imputation techniques, with the x axis indicating the strategy number. Error bars indicate one standard deviation. (Refer to Table 5.1 for the definitions of the strategies.)

4 and 5). This is because the partial autocorrelation results indicate that the sensory data is highly correlated with 1 past history data point, not 5 data points. If the partial autocorrelation indicates that the sensory data is highly correlated with 5 past history data points, (i.e., a high lag 5 value), the performance of strategies 4 and 5 may outperform "do nothing". This shows that the correct time model directly affects the performance. Strategies 2 and 3 just use a fixed value for missing data. As the missing data pattern changes, the Fuzzy ART treats the data as different classes as the number of classes/categories increase, resulting in more false alarms. Strategies 7 and 8 tried to learn the missing data patterns with a fixed value or an indicator. These strategies do better than strategies 2 and 3, but still have high mis-classifications due to changing missing data patterns. The EM imputation has a relatively high accuracy since it finds the best distribution for the observations in terms of likelihood. However, it is computationally expensive and requires offline learning. In a WSN application, if the sensory data is highly correlated in time and/or space (i.e., passes the time and space statistical tests), the developed spatial-temporal strategy works the best. Note that the time and space models in the algorithm are subject to change based on the environment. It is important to use the appropriate time and/or space model to estimate the missing values.

Missing data experiment with artificial data

An artificial data with no time and space correlations are generated to determine the performance of the developed imputation technique when the data is not time and space correlated. There are 4 classes of 2-dimensional data with 1024 data values for each class and giving a total of 8192 data samples. Each class has 2 Gaussian mixtures, each with a weight of 0.5. The sample means are shown in Figure 5.2. The reason for using Gaussian distribution for AI data is because not only is the Gaussian model the simplest model, but it also assumes the least of its data; therefore, it has the most information compared to other models. All mixtures have an identical covariance matrix (e.g. $\{1, 0\}$ and $\{0, 1\}$). This indicates that the generated data have no spatial correlations. Missing values have been randomly placed at 40% of the entire set. Since the data has been randomly sampled from different classes each time, this guaranteed that there are no temporal correlations in

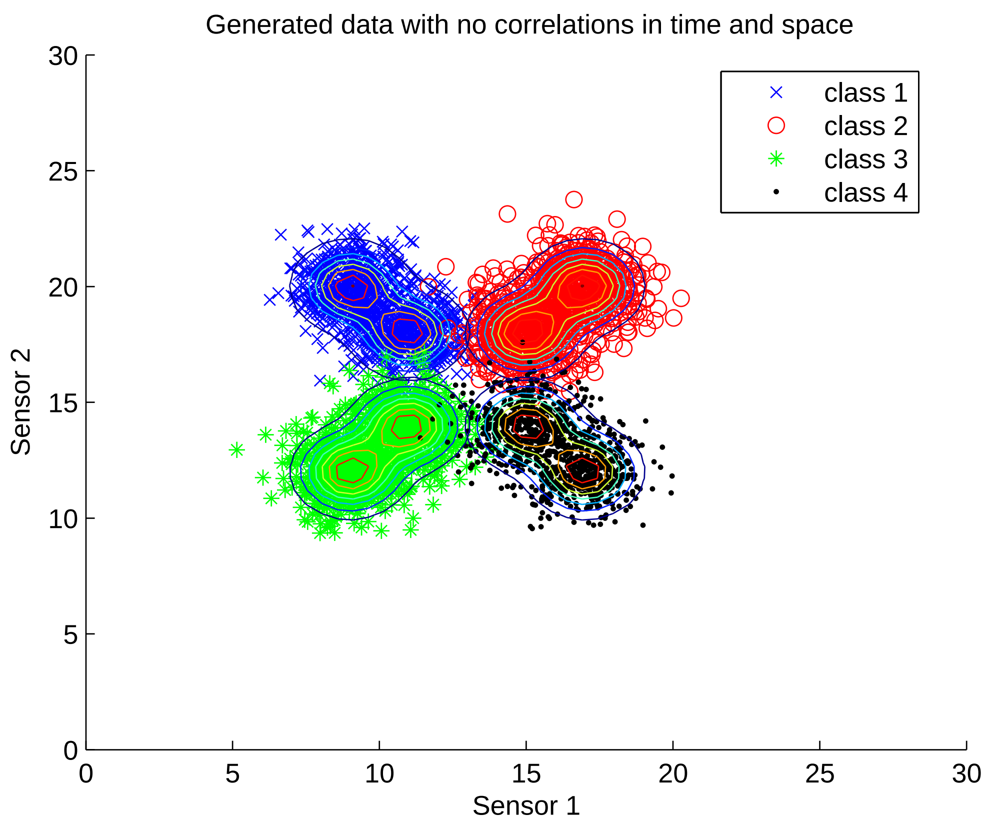


Figure 5.2: Generated 2-dimensional sensory data with no missing values. The true model contains 4 classes, and each class has 2 mixtures, each mixture has weight 50%.

the data. After the missing data values have been imputed by different techniques, they are put into the Fuzzy ART neural network for classification. The class categorizations from the Fuzzy ART neural network are compared against the ground truth to obtain the correct classification C.

The accuracies of different imputation techniques averaged over 10 sets of data are shown in Figure 5.3. All missing data imputation strategies performed relatively close to each other and all of them performed poorly (under 40%), due to the fact that there are no time or space correlations. To determine the significance of these results, the Student's T-test is applied to the accuracy results for the spatial-temporal imputation scheme compared against other imputation strategies. This test confirms that the differences in these results are statistically significant, with a confidence level of 99.5%. The spatial-temporal imputation strategy performed better than the other strategies. This is because this generated data contained many observations that have the same sensor 1 and sensor 2 readings (notice the middle section of Figure 5.2). However, if the generated data did not have as many observations that have the same sensor 1 and sensor 2 readings, the developed spatial-temporal strategy would perform about the same as the other approaches. The EM approach (strategy 9) did not perform better than other imputation strategies, since it assumes that there is one Gaussian distribution in the generated data. In addition, the EM imputation's performance is dependent on the initial values. To explain this situation, one needs to recall that the Fuzzy ART neural network builds the prototype online according to observations collected. In other words, different prototypes are kept in the Fuzzy ART neural network if the data is presented in a different order. The Fuzzy ART neural network has lower accuracy if the time correlation is removed from the data presented to the neural network. In summary, all the imputation strategies have similar accuracy performance (except for the spatial-temporal imputation) when the data have no correlations in time or space. The spatial-temporal imputation outperforms others because the data is slightly biased (i.e., many data instances have the same x and y values). Nevertheless, some imputation strategies require less computation and space, and thus they are preferred in actual implementation. The developed algorithm outperformed the other strategies under all testing conditions.

5.2 Nearest neighbor missing data imputation for WSNs

As shown in the initial missing data study, environmental data tends to correlate in space and time. Therefore, searching nearest neighbors from spatial and temporal datasets yields good performance. In the previous spatial-temporal missing data technique, the system makes use of the spatial property of the WSN and imputes the missing sensor value with the most common sensor reading within the same cluster. If no spatial data is available at the current time instance, the algorithm uses the missing sensor's latest known value (temporal correlation assumption of the environment). However, the spatial-temporal imputation poses some shortcomings. First, the right time and space models have to be used in order for this technique to have good performance. Additionally, the developed time and space

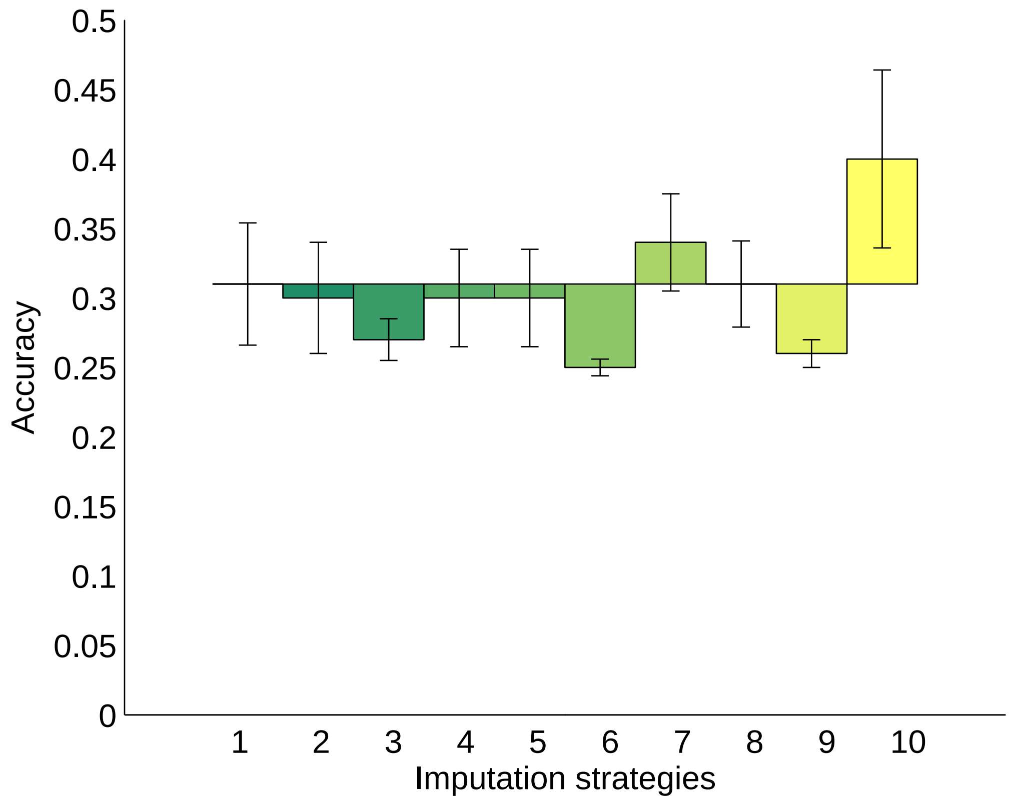


Figure 5.3: Accuracies of different imputation techniques. Error bar indicates one standard deviation. (Refer to Table 5.1 for the definitions of the strategies.)

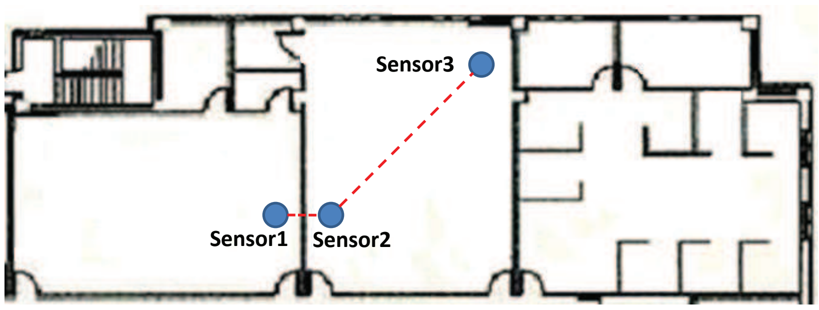


Figure 5.4: Example: correlations between sensors are not necessarily equal to their Euclidean distance. Sensors 1, 2 and 3 are deployed into two offices. Even though the Euclidean distance between sensors 1 and 2 is smaller than sensors 2 and 3, the sensor correlation between sensors 1 and 2 is lower than the correlation between sensors 2 and 3.

correlations tests must be performed offline. Further, it requires a brute-force search on available time and space correlated data, which can be expensive as the data size increases. Finally, the approach assumes sensors within a cluster have the same sensor readings, and the distances between the sensors are proportional to the sensor value correlation. However, these assumptions may not be true in certain environments. For example, Figure 5.4 shows that sensors 1, 2 and 3 are deployed in two offices. Even though the Euclidean distance between sensors 1 and 2 is less than the distance between sensors 2 and 3, the sensor correlations between sensors 2 and 3 is higher than the sensor correlations between sensor 1 and 2, since sensors 2 and 3 are located in the same room and sensors 1 and 2 are located in two different rooms. The spatial-temporal algorithm assumes that the Euclidean distance is the same as the sensor correlations. This research explores an automatic and efficient Nearest Neighbor (NN) imputation for missing data in WSNs.

The $\mathcal{K}$-Nearest Neighbor ($\mathcal{K}$-NN) method is a common *hot-deck* imputation method, in which $\mathcal{K}$ candidates are selected from the neighbors such that they minimize some similarity measure [Jonsson and Wohlin, 2004]. The *hot-deck* imputation is one of the most commonly used missing data imputation techniques, where it fills in missing values on incomplete records using values from similar, but complete records of the same dataset [Rubin, 1987]. Thus, the NN imputation approach used in this study is performed on a complete dataset, which is obtained by removing all the data instances with missing values in them. The

$\mathcal{K}$-NN imputation approach has many attractive characteristics [Rubin, 1987]: 1) it is a non-parametric method, which does not require the creation of a predictive model for each feature with missing data; 2) it can treat both continuous and categorical values; 3) it can easily deal having cases where there are multiple missing values; and 4) it takes into account the correlation structure of the data. The most important characteristic is its capability of using auxiliary information, such as space and/or time correlation between sensor node values. As shown in earlier missing data findings, using spatial correlations to impute missing data yields high performances.

It is important to search for closest match of the missing values efficiently. This study explores an efficient data structure for nearest neighbor imputation — kd-tree. The kd-trees are one of the earliest and most popular data structures for NN retrieval. They were initially introduced by Bentley [Bentley, 1975]. The kd-tree improves the previous spatial-temporal missing data imputation by automatically learning the sensor data correlations in time and space. For example, each sensor node in the network can store observations from their nearby sensor nodes in a kd-tree. When the querying node sees a missing attribute, it can retrieve the missing value by searching the nearest neighbor(s) from the built kd-tree. The advantages of using a kd-tree to store the sensor data are that the tree building process is simple and data-driven, and the search process for each observation vector is a localized search with computation time of $O(\lg n)$, where n is the size of the training data. In the designed WSN sensor imputation process, the kd-trees are able to capture the spatial-temporal correlations automatically without human supervision; hence, human operators do not need to have any initial knowledge of the environment or the sensor network deployment.

Traditional mean-variance kd-trees use variance and Euclidean distance to split k-dimensional data, since the search time can be greatly reduced by starting from the dimension that varies the most (i.e., the largest variance) and skipping dimensions that do not change much. The underlying assumption of using Euclidean distance is that all dimensions have equal weights. One problem with using the same weights for all dimensions is that when the range of values for each dimension are different, the estimated distance differs from the actual distance. In WSNs, some parts of the region may have more missing values than others. In the presence of missing values, the variance estimated from the incomplete data may not correctly represent the true distribution. Therefore, the system should also consider the missing rates. To overcome this problem, a weighted variance and weighted Euclidean distance measure are developed. The weight factors can be used to control the order of dimensions to be searched.

The main contributions of this imputation method are two-fold: First, a $\mathcal{K}$-NN missing data imputation technique is developed that enables the system to utilize space and/or time information; second, a weighted function is defined to make the kd-tree data structure more suitable for missing data in WSNs. In the remainder of this section, the overall imputation procedure is described in Subsection 5.2.1. Then, the developed approach is described in detail in Subsection 5.2.2. The time and space complexity analysis of the developed approach is given in Section 5.2.3. Finally, the experimental results are presented in Section 5.2.4.

5.2.1 Overall NN imputation technique

Sensor nodes in a WSN normally have two roles — sensing the environment and gathering data from other nodes in the network. The developed missing data imputation technique is mainly designed for nodes that gather data from other nodes (i.e., clusterheads, data hops, or a sink), because re-transmitting the missing values via wireless communication costs battery life on sensor nodes.

As illustrated in Figure 5.5, the imputation procedure works as follows. Each data gathering node builds a kd-tree using the partial dataset with no missing values during the initial training period. As mentioned earlier, removing training instances with missing values is a well established *hot-deck* imputation technique. The incomplete data set is not useful for imputation, since adding them in the tree would slow the search time. Therefore, training instances with missing values are discarded during construction of the kd-tree. This approach assumes some reasonable upper bound on the percentage of missing data. The kd-tree construction algorithm chooses the largest weighted variance of all dimensions and uses the mean value of the chosen dimension as the splitting point. The splitting process iterates until all data points have been separated in the plane. After the kd-tree is fully constructed, imputation begins. When the current observation has missing values,

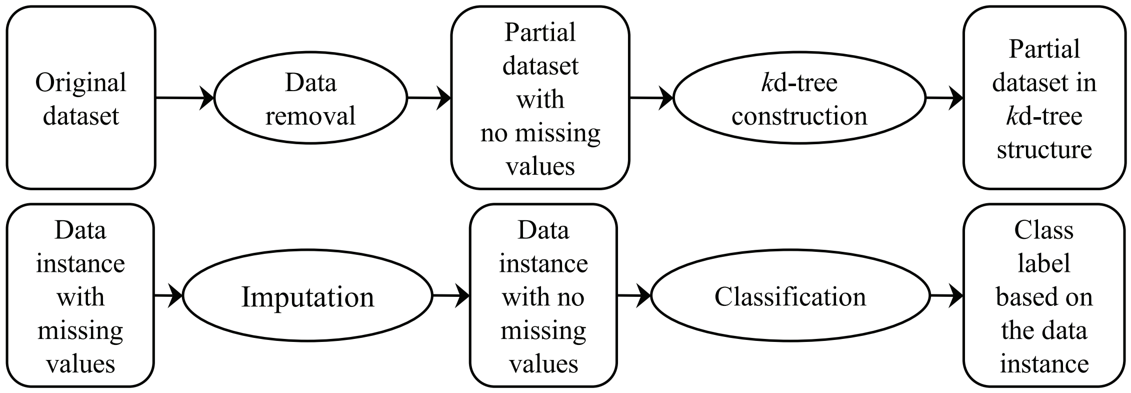

Original dataset
Data removal
Partial dataset with no missing values
kd-tree construction
Partial dataset in kd-tree structure
Data instance with missing values
Imputation
Data instance with no missing values
Classification
Class label based on the data instance

Figure 5.5: Overall view of the NN imputation approach. Top: the training procedure for constructing the kd-tree. Bottom: the imputation process when part of the current observation values are missing. The imputation procedure is a NN kd-tree search.

the algorithm searches for its nearest neighbor(s) by traversing the kd-tree and replacing the missing values in the current observation with the nearest neighbor values. The kd-tree search uses the designed weighted Euclidean distance to find the nearest neighbor values. Finally, the observation instance with no missing values can be evaluated using a classifier. The following section describes the developed metric for constructing and searching the kd-tree in detail, along with the definitions of the weighted variance and distance measure.

5.2.2 kd-tree data structure for NN imputation

In order to search for $\mathcal{K}$-nearest neighbors, a kd-tree data structure is utilized. The conventional kd-tree data structure assumes that each dimension has equal importance. Thus, the dimension with the largest variance will be split. The conventional kd-tree data structure is constructed by splitting the dimension with the largest variance. The variance σ^2 is defined as follows:

$$\sigma^2 = \frac{\sum_{n=1}^{N} x_n^2 - (\sum_{n=1}^{N} x_n)^2/N}{N-1} \tag{5.4}$$

where x is a random variable and N is the size of the dataset. The distance used is the conventional Euclidean distance, defined as follows:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{K} (x_i^{(k)} - x_j^{(k)})^2} \tag{5.5}$$

where, x_i and x_j are two data vectors with K dimensions. The average squared distance between objects across all K dimensions is given by:

$$\hat{D} = \frac{1}{N^2}\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}(x_i^{(k)} - x_j^{(k)})^2 \tag{5.6}$$

Due to missing data, the variance computed from the complete data is not accurate. Thus, in the presence of missing data, one can choose the dimension to split the kd-tree by using weighted Euclidean distance, defined as follows:

$$\hat{d}(x_i, x_j) = \sqrt{\sum_{k=1}^{K} w^{(k)}(x_i^{(k)} - x_j^{(k)})^2} \tag{5.7}$$

where w is the weight factor. The average squared distance between objects for all K dimensions is given by:

$$\hat{D}_w = \frac{1}{N^2}\sum_{k=1}^{K} w_i^{(k)} \sum_{i=1}^{N}\sum_{j=1}^{N}(x_i^{(k)} - x_j^{(k)})^2 \tag{5.8}$$

In kd-tree construction, there are many ways to choose which dimension to split from. The system should split a dimension/sensor in which the observations are well spread, since in general, the sensor with the largest size may have the most influence on the classification process. One possibility is to choose the dimension with the largest variance value. Specifically, each sensor k would compute the variance from its training observations. With missing values in a WSN, the system also needs to set weights to account for missing values in that dimension. The weights are used to control sensors with different variances. The weighted variance of dimension k can be viewed as the scoring function of dimension k, in which the score is proportional to the variance and inversely proportional to the percentage of missing data. In general, as the variance increases, the score increases as well; and, as the amount of missing data increases, the score should decrease. In this research, the weight of dimension k is set to $w_k = (1 - M_k)$, and the score of dimension k is $\sigma_k \times w_k$, where σ_k is the variance and M_k is the percentage of missing data for sensor k. This score function automatically accounts for both missingness and variance values. Note that the actual function depends on the application; system designers can choose a proper weight function. After choosing the split dimension, the system needs to determine the splitting value. Generally, it is good to choose a k such that the split is in the middle of the points along the splitting dimension. There are many ways to accomplish this. Splitting at the mean results in square hyper-rectangles, but may lead an unbalanced tree. Splitting at the median ensures a balanced tree. An unevenly distributed tree can result in long and skinny hyper-rectangles. The developed system chooses the splitting value to be the mean of the training observations of the splitting sensor, since it is determined by the distribution of the data.

To find nearest neighbors, a tree search needs to be performed based on the defined distance metric. The following are some well-known properties of a weighted Euclidean distance:

- If the $w^{(k)}$ for each dimension is set to be the identity matrix, then the weighted Euclidean distance is equivalent to the conventional Euclidean distance.
- If the $w^{(k)}$ is set to be the inverse of the variance squared at dimension k, then the weighted Euclidean distance is proportional to the Mahalonobis distance with the diagonal covariance matrix.

If the system sets the weight to be two times the inverse of the variance squared at dimension k, then the system has demoted the importance of dimension k by half. The weighted variances and Euclidean distance measure are illustrated through examples in the tree construction and NN imputation. If the system uses inaccurate estimation of the variance to construct the kd-tree data structure, then searching for the nearest neighbor takes longer.

The kd-tree construction

A kd-tree is a multidimensional data structure that decomposes a multidimensional space into hyper-rectangles. The constructed tree is a binary tree with both a dimension number and a splitting value at each node. The procedure for constructing the kd-tree takes a set of training points P as input, where P contains k dimensions and n numbers of training instances. The procedure returns a kd-tree storing the training data P. Each node corresponds to a hyper-rectangle, which contains the following fields: splitting dimension number, splitting value, left kd-tree, and right kd-tree. A detailed kd-tree construction algorithm is given in Algorithm 3. The developed implementation closely follows the algorithm that is presented by Friedman in [Friedman et al., 1977], except the added CHOOSESPLIT function. The developed CHOOSESPLIT function chooses the splitting order based on weighted variances instead of variances.

Algorithm 3 BUILDKDTREE(P)

Input: A set of observations P, and a weight vector w. **Output:** a kd-tree storing P.

```
1: if $P$ is empty then
2:     return $NULL$.
3: end if
4: if $P$ contains one observation then
5:     return a leaf storing this observation.
6: end if
7: $s \leftarrow$ CHOOSESPLIT($P, w$ ), where $s$ is the splitting dimension.
8: $\mu_s \leftarrow$ get mean of dimension $s$.
9: $pLeft \leftarrow \{x \in P : x_s < \mu_s\}$.
10: $pRight \leftarrow \{x \in P : x_s \geq \mu_s\}$.
11: $kdLeft \leftarrow$ BUILDKDTREE($pLeft$).
12: $kdRight \leftarrow$ BUILDKDTREE($pRight$).
13: return $k$d-tree with fields $[s, \mu_s, kdLeft, kdRight]$
```

CHOOSESPLIT(P, w)

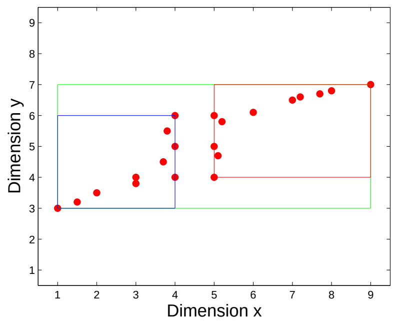
```
1: $\sigma \leftarrow$ get variances of all dimensions in $P$.
2: $score \leftarrow$ apply weights to all variances (i.e., $\sigma_i \times w_i$).
3: return $j \leftarrow$ dimension of the max $score$.
```

In WSN applications, the training data P contains n observation vectors made by k sensor nodes from time 1 to time t_n. Each vector has k dimensions, and each dimension corresponds to one sensor node in the WSN. When there are missing values from the current observation, the system can traverse the kd-tree and find which is the closest match to the current observation, and impute the missing values with the closest match. The training vectors in P are recursively split into two subsets. One subset contains observations smaller than the splitting value μ_s, the other contains the observations larger than or equal to the splitting value μ_s. The splitting value μ_s is stored at the root and the two subsets are stored recursively in two subtrees.

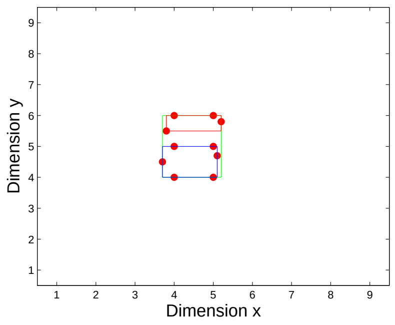
The following is an example to demonstrate the use of weight factors for missing data in a WSN during kd-tree construction. Suppose there are two sensor nodes observing the environment. The system collects a set of 2-dimensional data points over time (shown in Figure 5.6(a)), where dimension/sensor x has a variance of 4.62 and dimension/sensor y has a variance of 1.64. Sensor x has a larger variance than sensor y. Therefore, the kd-tree first splits dimension x, then dimension y when no weights are used. Suppose sensor y has approximately 52% missing values at both the beginning and the end of the data collection period. A new kd-tree constructed by using the remaining data instances is shown in Figure 5.6(b). With no weights included, the new variances become 0.39 and 0.58 for sensors x and y, respectively. The new kd-tree first splits dimension y, then dimension x. This split is not consistent with the complete data case. This is because sensor y has missing values and its variance obtained from incomplete data is different from the complete data. By adding weights (i.e., $w_1 = (1 - 0\%) = 1$ and $w_2 = (1 - 52\%) = 0.42$), and reconstructing the tree, the splitting order of the kd-tree is changed back to its original form (see Figure 5.6(c)). The final scores (weighted variances) are $0.39 \times 1 = 0.39$ and $0.42 \times 0.58 = 0.28$ for dimensions x and y, respectively.

NN imputation

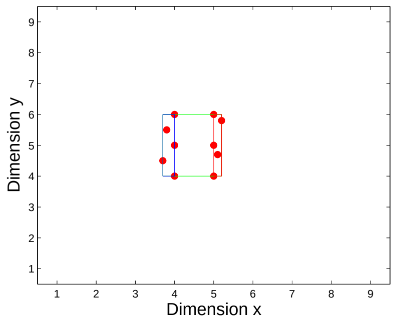
When the current observation has missing values, the developed NN imputation algorithm is activated. The imputation works as follows: First, find the current observation's closest match from the built kd-tree by using a NN search algorithm. Then, fill in each missing value with the corresponding value in the best match found. The developed kd-tree search algorithm closely follows the common NN searching implementation [Friedman et al., 1977] with slight modifications: the algorithm uses the weighted Euclidean distances given in Equation (5.8) instead of the Euclidean distance. The algorithm needs to search both sides of the subtrees when the current dimension has missing values, because without any information on that dimension, the nearest neighbors could reside on both sides of the (sub)tree. Therefore, keeping the dimensions with the most missing values towards the



(a) Complete data



(b) Incomplete data



(c) Incomplete data with weight

Figure 5.6: An example illustrates the usage of applying weights in the kd-tree construction. Subfigure (a) shows the kd-tree built with complete training data and this is the desired splitting. Subfigures (b) and (c) are the implications of using partial missing data and partial missing data with weighted variances, respectively. The green box represents the root partition, the blue box represents the left node of the root partition and the red box represents the right node of the root partition. The variances are 4.62 and 1.64 for dimensions x and y, respectively. The kd-tree first splits from the widest dimension x on the root, then dimension y on the second level; the process continues until all nodes are separated. The green region denotes the top node, the black region denotes the left branches, and the red region denotes the right branches. Subfigure (b) shows a kd-tree constructed with incomplete data from Subfigure (a). With incomplete data, the new variances become 0.39 and 0.58 for dimensions x and y, respectively. With some missing values, dimension y is the widest dimension. Hence, the new kd-tree splits from the widest dimension y at the root, then dimension x on the second level. Subfigure (c) shows a kd-tree with weighted variances using incomplete data. the variances are set based on the designed weight function, hence, the new variances are 0.39 and 0.28 for dimensions x and y, respectively. With the weighted variance, dimension x is adjusted to be the widest dimension. The kd-tree split from dimension x on the root, then dimension two on the second level and so on. Therefore, using weights to re-rank the dimension with more missing values is very important.

bottom of the tree can help the system localize the search, which can improve the search time.

5.2.3 Complexity analysis

To find the best matches from the constructed kd-tree, the designed algorithm is a direct modification of the algorithm presented in [Friedman et al., 1977]. The algorithm time complexity is $O(kn \lg n)$, where k is the number of sensor nodes that the data gathering node is listening from and n is the number of training instances. The kd-tree searching time is constrained by the amount of data that the algorithm can eliminate in each dimension. As shown in Figure 5.7, based on the distance and mean (split point) for each dimension, the tree search algorithm decides to traverse the left or right side of the kd-tree; hence, at each iteration, one side of the subtree can be eliminated. It is important to choose the top levels of the kd-tree wisely, since they determine the tree search order and can help the system limit the amount of search early on. In the $\mathcal{K}$-NN missing data imputation approach, each missing value corresponds to a data point in a dimension. If the dimension is at the top of the tree, the traversal can be expensive, since the search algorithm must continue to search both sides of the tree. If the missing value dimension is towards the bottom of the built kd-tree, the NN search is faster, since the system searches both sides of the subtree with fewer data points. Therefore, by using weights to lower the dimensions that have high missing rates toward the bottom of the kd-tree, the system can speed up the nearest neighbor searching process. The assumption made here is that if sensor nodes tend to miss values during the training period, they are likely to have similar missing rates online. The more one can constrain the search to part of the subtrees, the faster the system can find the NNs for the missing values.

The kd-tree construction can be performed online as well. The simplest way of constructing the kd-tree online works as follows: for each complete data instance, add the new instance to the tree as in other search trees. Then, use the kd-tree construction algorithm as shown in Algorithm 3. This is a simple modification to the existing algorithm. Assume there are j complete instances in the kd-tree. Then, the online construction algorithm takes $O(\lg j)$ time to traverse and add the new instance to the tree. Reconstructing the

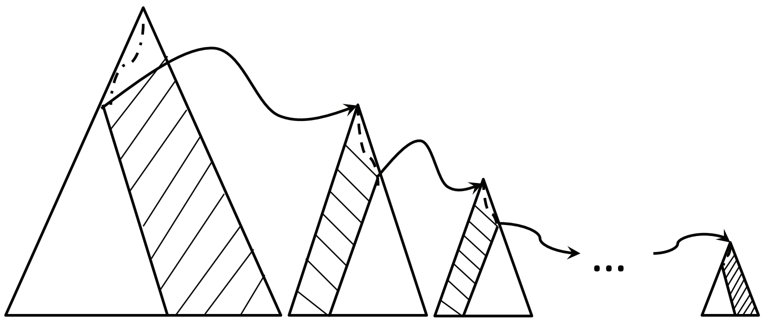


Figure 5.7: An example of a kd-tree search. At each level/dimension, eliminate some part of the tree based on the distance.

Table 5.4: Comparison of EM and kd-tree imputations.

Imputation	**Time**	**Space**
EM-based	E-step: $O(mkn)$ M-step: $O(mkn)$ Iterate E and M step until converge	$O(kn + m^2)$
kd-tree	Construct: $O(kn \lg n)$ Search: $O(\lg n)$	$O(kn)$

kd-tree takes $O(j \lg j)$ time.

The space cost of storing a kd-tree is linear, i.e., $O(kn)$ where k is the number of sensors in a cluster and n is the number of observations in the past history. A system designer can always trade off space for time. Suppose one tree is constructed for each combination of missing sensor values. When an observation has a missing pattern in it, the tree with only that missing pattern can be searched. The total possible number of trees for k sensors is $\mathcal{K}_k^1 + ... + \mathcal{K}_k^k \times k!$. For example, suppose there are four sensor nodes, there are 64 possible complete trees. Storing data into different trees may limit the search time; however, the space grows quickly with this method.

To compare with the EM-based imputation, time and space complexity is analyzed and presented in Table 5.4. The time and space complexities are calculated based on a single sensor node and find one imputation value, where n is the number of observations made from time 1 and k is the number of sensor nodes within proximity (dimensions). m is the number of Gaussian mixtures used for EM-algorithm. Both imputation strategies use space linear in k and n. The time complexity for EM-based imputation is much more than the kd-tree based imputation technique. Suppose EM-based imputation assumes m mixtures of Gaussian models are used, resulting in $O(mkn)$ for the E-step, and $O(mkn)$ for the M-step. The EM procedure continues until convergence. The space cost for EM-based imputation is $O(kn+2m+m^2)$, which includes storing all data points and Gaussian mixture models.

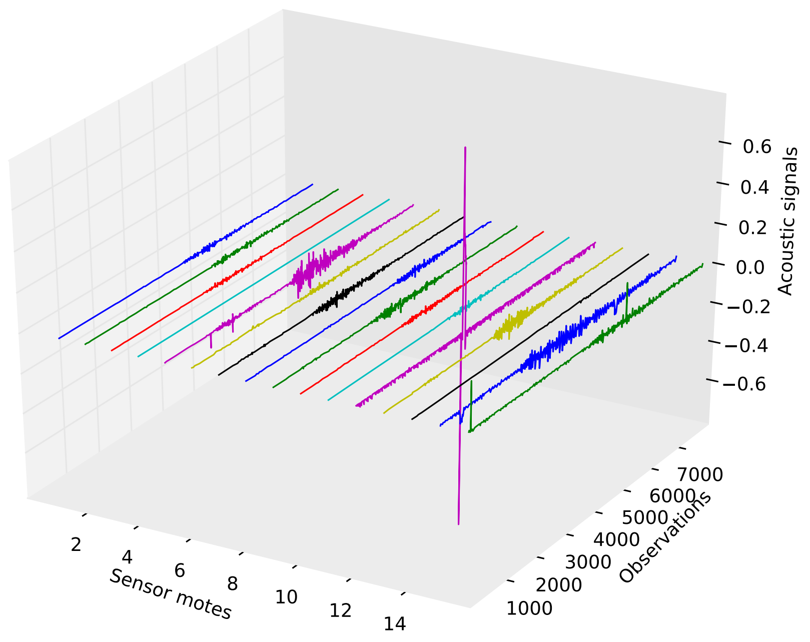
In a brute-force NN searching method (like the previous spatial-temporal imputation method in Section 5.1), assuming no sorting is performed, then searching may require that each data vector be examined. To find the nearest neighbor of each point in a data set of n vectors requires $O(n^2)$ time using a brute-force method. With k sensor nodes, and an $O(n \lg n)$ sorting algorithm, the brute-force search has a total search time of $O(kn^2)$.

The additional computational and space costs help to save communication costs in WSNs. If the developed missing data algorithm were not in place, the WSN would need to continue to broadcast and query the data transmitting sensor nodes repeatedly until all data are received.

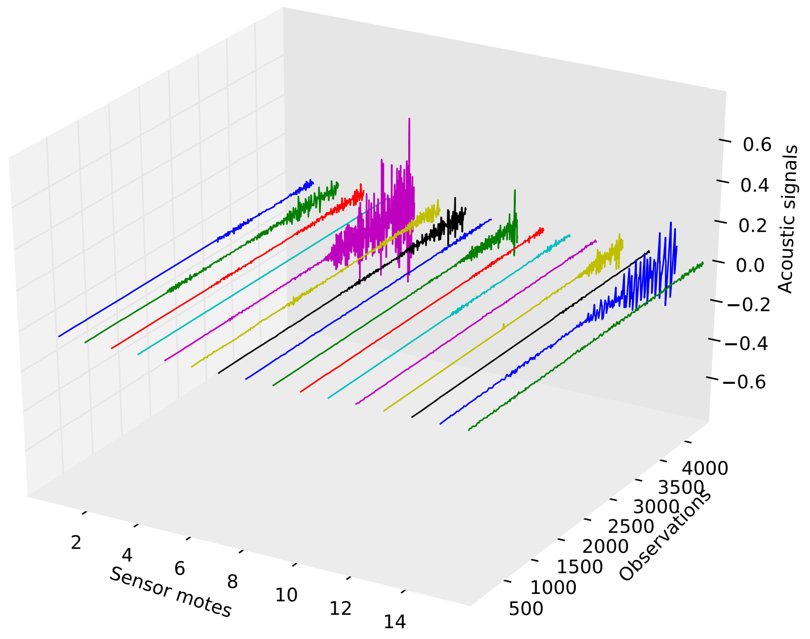
5.2.4 Experiment

A volcano monitoring dataset is used to test the developed missing data algorithm. The dataset was collected by Werner-Allen, et al., at the Volcano Reventador. The detailed data collection process can be found in [Werner-Allen et al., 2005]. The data used in the experiments was obtained from a network of 16 Crossbow sensor motes. Each of the sensors continuously sampled acoustic data at 100 Hz over a 19-day deployment. Motes used an event-detection algorithm to trigger on interesting volcanic activity and initiate data transfer to the base station. Each data collection event is "triggered" when the event-detection algorithm exceeds a threshold. The 60 second download window contains approximately 30 seconds before the trigger and 30 seconds after the trigger. Note that the trigger may not be exactly centered at the intended event within the download window.

The kd-tree software with weighted variance and distances is modified from the kd-tree implementation from the WEKA [WEKA, 2010] open source machine learning software. All of the following experiments have been conducted on a PC with an Intel Core duo



(a) Training data



(b) Testing data

Figure 5.8: Acoustic sensor readings from all 16 sensors. Part (a) shows the training data to build kd-trees, and part (b) shows the testing data.

processor E6320 running at 1867 MHz, and 2 GB of main memory.

Preprocessing

During the deployment, the network recorded 229 earthquakes, eruptions, and other acoustic events; however, only eight events have complete readings from all 16 sensor nodes. After discarding the data that does not have all 16 sensor' readings, only 8 minutes of data are left; this data is selected as the training and testing data. With this complete data and all 16 sensors' readings, the ground truth can be obtained for the following missing data experiments.

Two one-minute events are chosen (one minute as training and one minute as testing data) to evaluate the developed imputation method in the following experiments. The training and testing of the 16 sensor nodes' acoustic signals are plotted in Figure 5.8. The training data contains 7950 observations, and the testing data contains 4415 observations. To evaluate the developed $\mathcal{K}$-NN missing data technique, the ground truth is created by training an unsupervised classifier using the training data and the testing data, respectively. There are 16 discrete categories obtained from the training and testing data.

The system then removes data from the complete testing set to create new testing sets with missing sensors. Based on observing the volcano dataset, if a sensor has missing values at the beginning of the fetch procedure, the values in that sensor would stay missing during the entire procedure. Therefore, 15 missing datasets are generated from the testing data by randomly removing sensors. For example, the first missing dataset is created by randomly removing one sensor from the original testing data. The second missing dataset is created by randomly removing two sensors, and so on. The last missing dataset is created by removing 15 sensors. Thus, the percentage of missing data for the newly generated testing cases are chosen to be one of the following percentages: $1/16, 2/16, \cdots 15/16$. The system also generated 9 other missing datasets with different missing data percentages (i.e., $10\%, 20\%, \cdots, 90\%$). The missing data are removed at random across the complete testing dataset based on the missing percentage.

To compare the accuracies of the developed NN missing data imputation technique, two EM-based imputation products. The first EM-based software is the state-of-art missing data imputation software Amelia II [Amelia II, 2010]. Amelia II makes use of a bootstrapping-based EM-algorithm, which is a parametric approach to estimate missing values and that assumes the complete data (that is, both observed and unobserved) is a multivariate Gaussian distribution. Note that NN imputation is a non-parametric approach, which makes no assumption of the data. Since Amelia II is installed under R (R is a free software environment for statistical computing that runs under Linux), it is difficult to compare run times of Amelia II and the developed NN imputation approach. Therefore, the run time of another EM-based imputation software under WEKA is compared with the developed NN imputation technique. Note that the WEKA EM-imputation software is still under trial testing. Therefore, both EM-based techniques performances are included.

Performance metrics

To measure the quality of the imputed volcano data sets, micro-average accuracy and macro-average accuracy metrics are used. Micro-average accuracy is defined as the ratio of the observations that have been correctly categorized (A), to the total number of instances that have been categorized (T). Macro-average accuracy is defined as the average accuracy for each class, i.e., the ratio of the number of correctly categorized observations in category i to the number of observations in category i.

The two averaging procedures bias the results differently — micro-averaging tends to over-emphasize the performance on the largest categories, while macro-averaging over-emphasizes the performance of the smallest categories. Both of the averaging results are often examined simultaneously to get a good idea of how the developed algorithm performs across different categories.

To ensure a fair comparison, the parameters of the classifier have been readjusted for each replacement strategy until the best performances are obtained.

NN imputation accuracies

As shown in Figure 5.9, the developed nearest neighbor technique has competitive performances with the Amelia II missing data imputation software in terms of micro-averaging accuracies. The kd-tree is constructed using approximately 8000 observations with 500

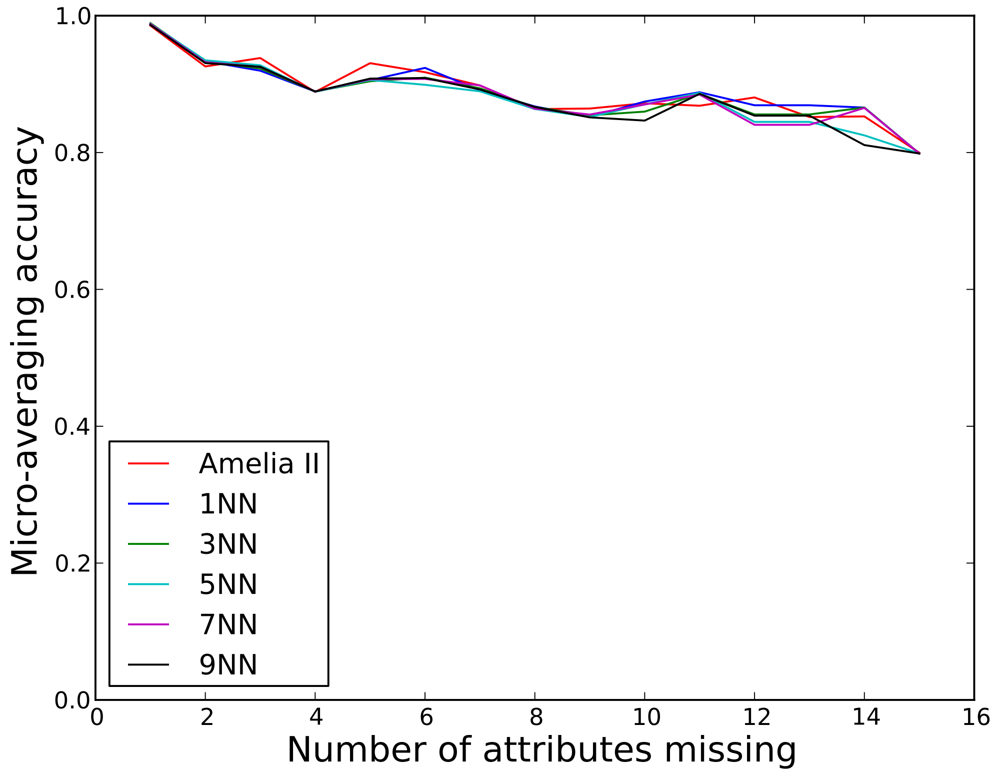


Figure 5.9: Micro-averaging of $\mathcal{K}$-NN ($\mathcal{K} \in \{1, 3, 5, 7, 9\}$) vs. Amelia II.

splits. The Amelia II missing data imputation software's performances are averaged over 30 trials.

As shown in Figure 5.10, the macro-average accuracies of the developed NN imputation technique are about the same as Amelia II. A shortcoming of Amelia II is that it assumes the underlying distribution is Gaussian. However, when the underlying distribution is not Gaussian, parametric approaches have a model mismatch problem. Additionally, parametric based approaches generally require large numbers of training instances to estimate the parameters of the models accurately. If the training instances are not sufficient to train the model, the imputation results would not be accurate. The developed NN approach is a non-parametric approach, with no assumptions of the data distribution. Therefore, its performances could be better than the parametric approach in some applications. Note that instead of replacing missing data with a single estimated value, $\mathcal{K}$-NN imputation can

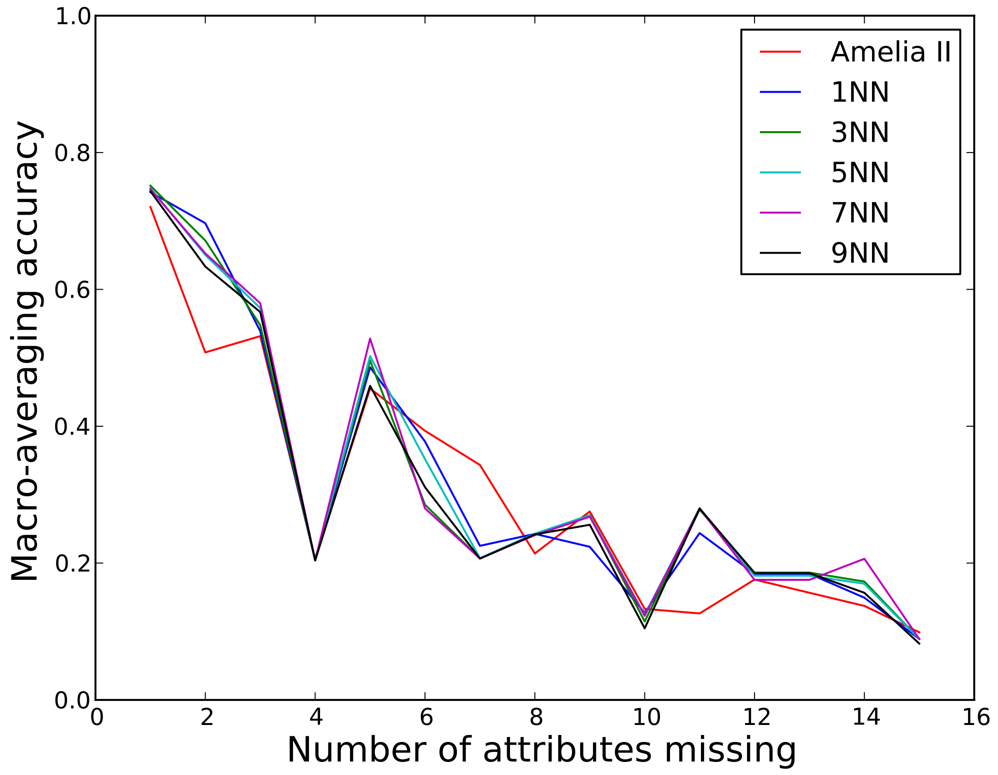


Figure 5.10: Macro-averaging of $\mathcal{K}$-NN ($\mathcal{K} \in \{1, 3, 5, 7, 9\}$) vs. Amelia II.

also replace the missing value with $\mathcal{K}$ different values, when $\mathcal{K}$ is larger than one. This technique is generally referred to as a type of Multiple Imputation (MI) methods. MI-based approaches replace each missing value with a set of plausible values that represent the uncertainty about the correct value to impute. In addition, the Amelia II imputation procedures have to iterate over patterns of missingness (that is, all the possible ways that a row has missing cells). Thus, the complexity grows quickly with the number of these patterns. For the developed kd-tree imputation approach, the cost is still linear by searching more nearest neighbors.

To obtain the statistical significance results on the comparisons between $\mathcal{K}$-NN imputation and EM-based imputation software, 10-fold cross-validations are performed on the training data. The EM imputation technique used in the following experiments is from WEKA machine learning software. The previous missing data accuracy results are determined based on the Fuzzy ART neural network's performance measure. In this experiment, the Total Squared Error performance is used to compare the imputation results between the NN imputation and the EM-based imputation algorithm. The Total Squared Error is defined as $\sum(Y - \hat{Y})^2$, where Y denotes the ground truth and $\hat{Y}$ denotes the imputation value. Figure 5.11 plots the mean and standard deviations of Total Squared Errors of 1-NN imputation and EM imputation over different percentages missing data. The mean and standard deviations are obtained from 10 sets of testing data. The 1-NN imputation has better performances than EM imputation technique in most cases except when 90% of the data is missing. To determine the significance of these results, the Student's T-test is applied to the Total Squared Error results for the 1-NN imputation technique compared against WEKA's EM imputation strategy. This test confirms that the differences in results are statistically significant (except for the 90% missing data case), with a confidence level of 99.5%. The NN imputation strategy performed better than EM-imputation in most cases because the training data set correlates in time and space, whereas, the EM-based imputation technique assumes there is only one Gaussian distribution in the training data. When there is too much data missing (e.g., 90% missing data), 1-NN imputation and EM-based imputation technique perform about the same because it is hard to find solutions when only a few sensors are available. In general, the distance error increases as the percentage of missing data increases.

The imputation times for each data instance of different imputation techniques are averaged over 10 sets of data and the results are shown in Figure 5.12. Nearest neighbor imputation uses less time compared to the EM imputation method, due to the fact that NN imputation does not have to iterate through all the data until convergence. To determine the significance of these results, the Student's T-test is applied to the time performances for the 1-NN imputation scheme compared against the EM-based imputation strategy. This test confirms that the differences in these results are statistically significant, with a confidence level of 99.5%. Therefore, the $\mathcal{K}$-NN imputation is preferred over the EM-based imputation technique for solving the missing data problem in WSNs.

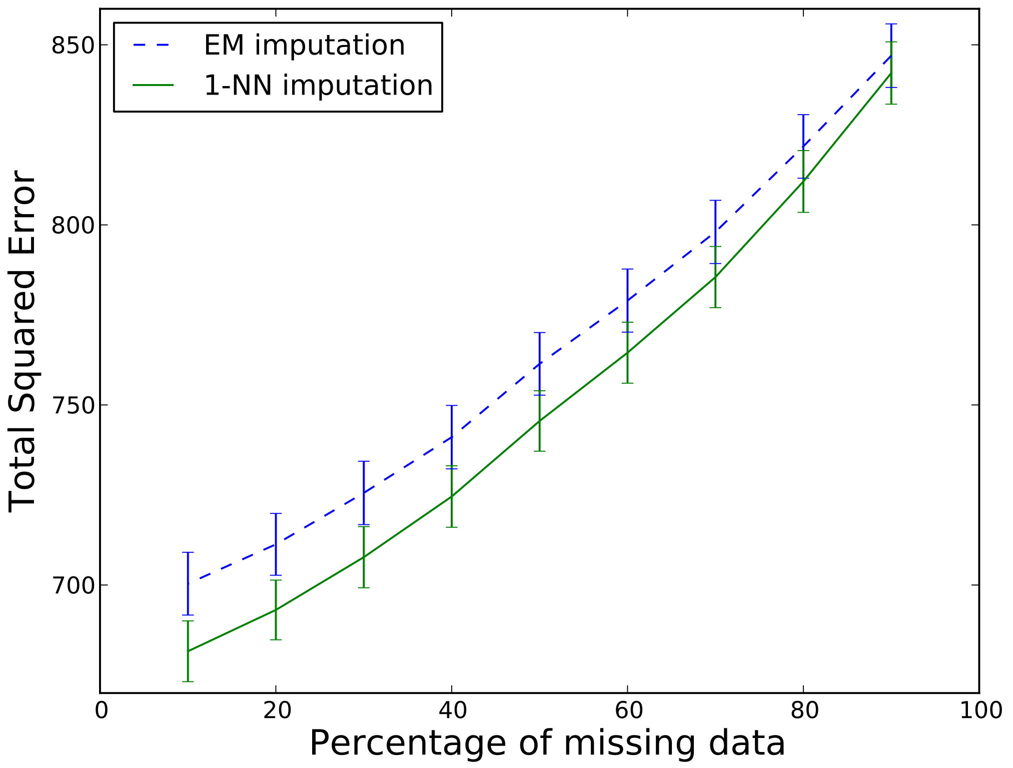


Figure 5.11: The Total Squared Error of 1-NN vs. EM-based imputations. The averaged total squared error and standard deviations are obtained from 10-fold cross validation on the training data. 1-NN imputation has less total squared errors than the EM-based imputation method except in the case of 90% missing data.

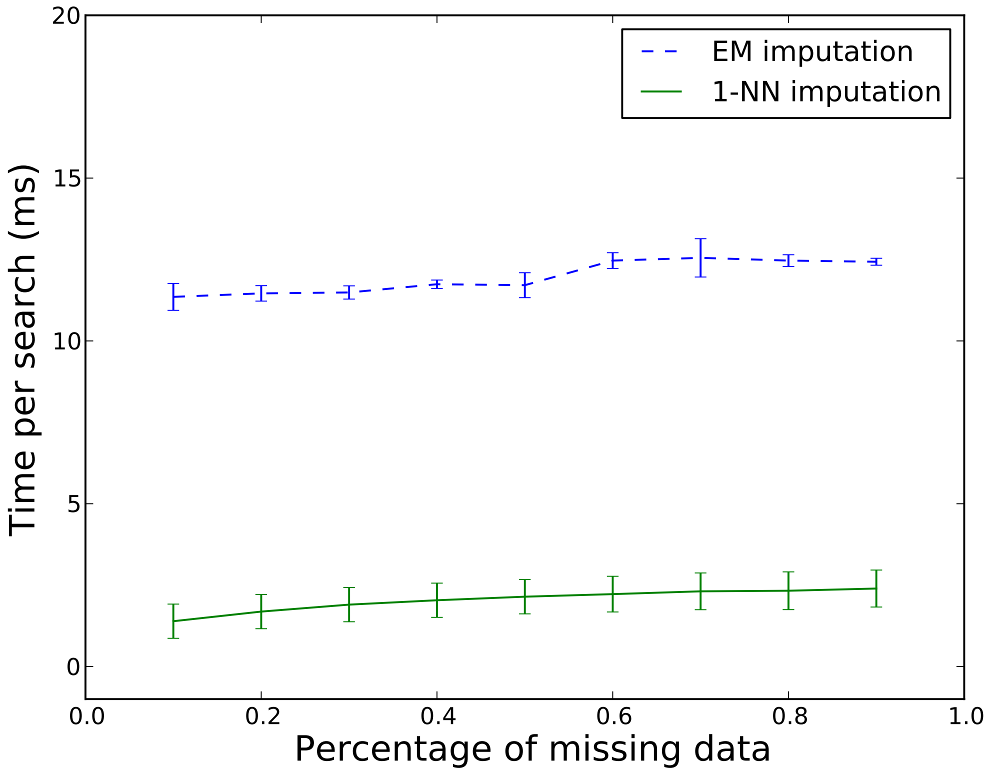


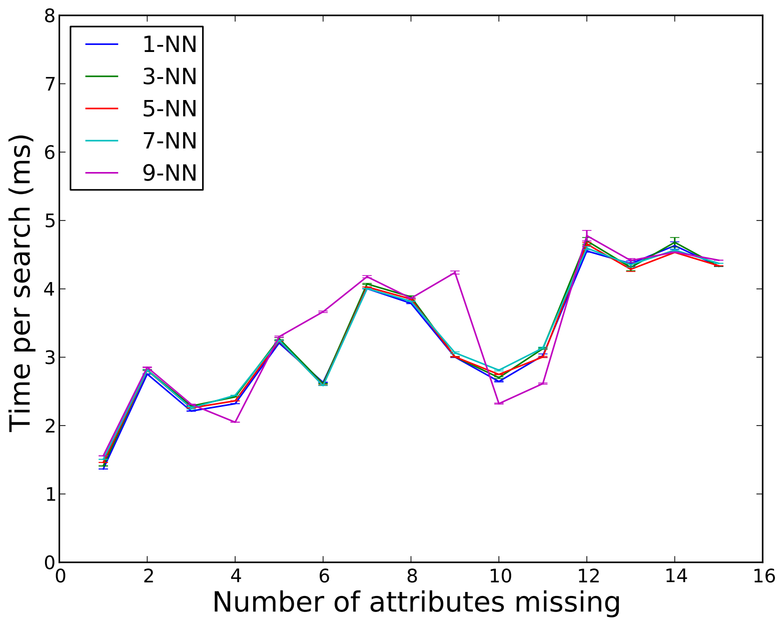
Figure 5.12: The imputation times of 1-NN imputation vs. EM-based imputation for estimating each data instance. The average time and standard deviations are obtained from 10-fold cross validation on the training data. For all missing percentages, 1-NN imputation has better imputation time than the EM-based imputation method.

Timing comparisons of searching different NNs

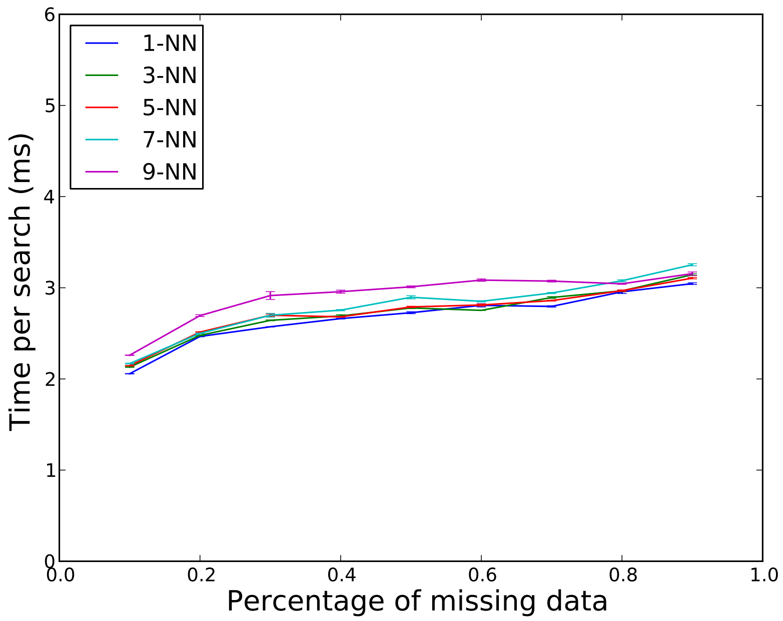
Figure 5.13 shows the average search time for $\mathcal{K}$-NN ($\mathcal{K} \in \{1, 3, 5, 7, 9\}$) per observation on the pre-built kd-tree, where Figure 5.13(a) plots the searching time for a different number of missing sensors, and Figure 5.13(b) plots the searching time for a different percentage of missing data. Note that the missing data is randomly selected at different percentages in Figure 5.13(b), and combinations of sensors missing are also randomly selected in Figure 5.13(a). The searching times for each nearest neighbor are averaged over 10 trials, and the variance to the searching time is plotted as error-bars. In most of the cases, the variances are too small to notice. All searching times per observation are between 1 to 5 ms. In general, the search time shows an upward trend as the volume of missing data increases. This includes both the different number of sensors missing and the different percentage of data missing. The searching time for finding 9 nearest neighbors is usually slightly longer than the rest in both missing data cases. The searching time for finding $1, 3, 5$, and 7 NN are approximately the same. The search time curves in Figure 5.13(a) are not as smooth as the curves in Figure 5.13(b). This is due to the combinations of missing sensors that are selected at random. Different combinations of the missing sensors create different search patterns for the kd-tree approach. Figure 5.13(b) shows much smoother curves than Figure 5.13(a), because the missing sensors for all the testing sets are selected at random. Depending on the sensor and combinations of sensors missing, some may take longer to traverse than others.

Timing comparisons after adding weights

To test the effect of the weight factor, the variances for all 16 dimensions (sensors) are evaluated. The sensor with the most variance (sensor 12) is selected, a very small weight is applied to its variance, and a kd-tree is built with the weighted variance. This selection is empirical and intended to demonstrate the usage of the designed weight factor. In the original kd-tree, sensor 12 has the highest rank; however, by adding the weight factor, sensor 12 ranks the last among all 16 dimensions in the new weighted kd-tree construction. The hypothesis is that the system uses more time to traverse the non-weighted kd-tree to find the nearest neighbors.



(a) Different number of sensors missing



(b) Percentage of data missing

Figure 5.13: Average searching time per observation for $\mathcal{K}$-NN ($\mathcal{K} \in \{1, 3, 4, 5, 7, 9\}$), where Part 5.13(a) shows the search time for different numbers of sensors missing. The missing sensors are selected at random. Part 5.13(b) shows the search time for different percentage of missing data. The searching time are averaged over 10 trials. Error bars plot the standard deviations of searching time. The missing data are selected at random.

Figure 5.14 shows the average search time and standard deviation per observation for different percentages of missing data. The solid lines denote the average search time for traversing kd-trees with added weights, while the dashed lines denote the average searching time for traversing regular kd-trees with no weights added. The searching times are averaged over ten trials and most of the variances are too small to notice. The performances are evaluated based on finding 1-NN with tree size d where $d \in \{50, 500, 1000\}$. The weighted kd-trees take less searching time than non-weighted kd-trees, because sensor 12 has the largest variance during training time; however, the testing data has missing values and the distributions are changed from the training data. It takes the system less time to localize its searches, because the system is able to eliminate more data as compared to the non-weighted kd-tree. Therefore, if the weights are used correctly, the system can localize the search faster.

The average search times for searching 1-NN on regular kd-trees are 2.9ms, 3.8ms, and 4.2ms for tree sizes of 50, 500 and 1000 splits, respectively. The average search times for searching 1-NN on weighted kd-trees are 2.8ms, 3.4ms, and 4.1ms for tree sizes of 50, 500, and 1000 splits, respectively. Thus, the percentage that the search time differs between the regular kd-trees and the weighted kd-trees are approximately 3%, 10%, and 2% for tree sizes of 50, 500, and 1000 splits, respectively. Note that the weight of only one dimension is changed (out of a total of 16 dimensions). Depending on the data distribution, re-ranking more dimensions may have more influence on the search time. The ranking of the dimensions in kd-tree construction according to the missing percentage is likely the most influential factor in the searching time.

This experiment shows that using weights are very important to retrieve nearest neighbors from the kd-tree data structure. The weight factors require some knowledge of the environment's and sensors' behaviors. For example, if a sensor constantly misses values, or transmits noisy data, the system can re-rank the search tree by applying small weights to the noisy sensor, saving search time. The designed weighted variance and Euclidean distance offer an automatic way of changing the importance of each sensor in the network. With changes in the environment and sensor nodes (i.e., power outage, replacements of new sensors, etc.) the system designers can choose any functions to calculate weights as desired. The weighted variance of dimension i should be proportional to the variance of

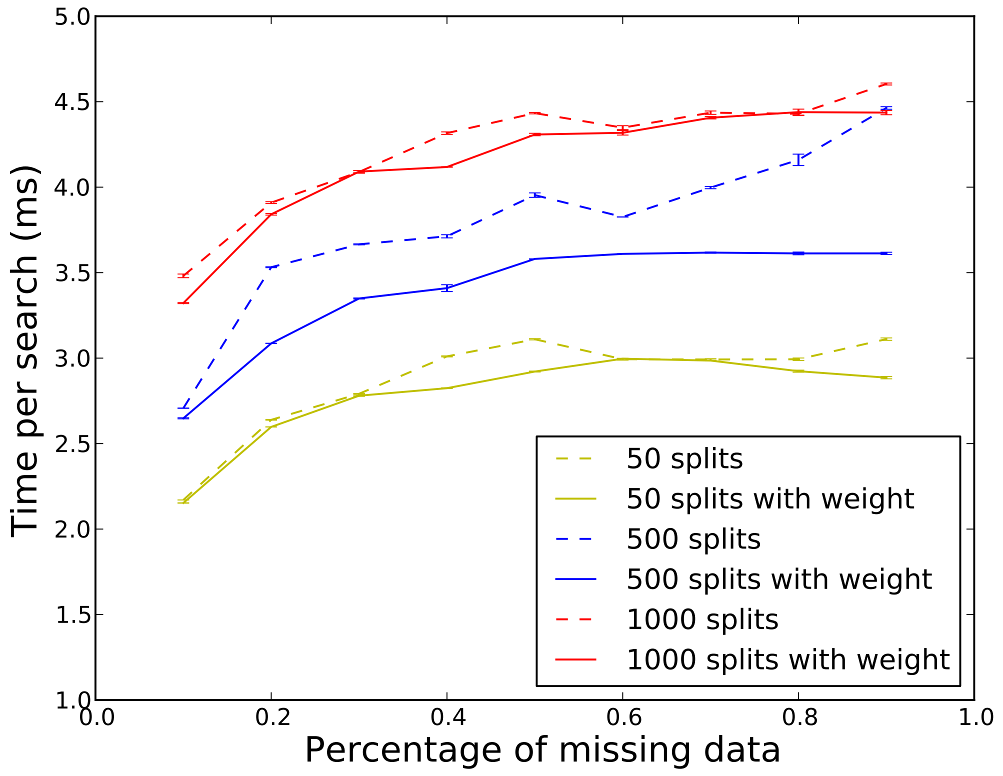


Figure 5.14: Searching time per observation for finding 1-NN from kd-tree sizes. The regular kd-trees are plotted using dashed lines and the weighted kd-trees are plotted using solid lines.

dimension i and inversely proportional to the missing percentage of dimension i. In summary, using a weight parameter allows the system to control the ranking of the kd-tree. As the environment or the network structure changes, the system can re-evaluate the weights and construct a new kd-tree easily without re-programming all sensor nodes in the WSN.

5.3 Summary

The environments in which WSNs operate in tend to be both time and space correlated. A novel spatial-temporal missing data imputation technique has been developed that takes advantage of such correlations. Then, several different missing data imputation strategies are compared and contrasted with the developed approach. The experimental results show that making use of the time and space information to estimate missing values allows the system to achieve high accuracy. A limitation of the spatial-temporal missing data imputation technique is that the process of finding the proper time and space models needs to be performed offline. In addition, this technique assumes that distances between sensor nodes correspond to the sensor correlations of those sensors. In order to overcome these limitations, an enhanced Nearest Neighbor imputation technique is developed that utilizes the temporal and spatial correlations among the sensor nodes. The NN imputation organizes a set of temporal and spatial correlated data into a kd-tree. To impute missing values, the system traverses the constructed kd-tree to find the nearest neighbor(s) of the querying data instance and replaces the missing values with the nearest neighbors. As opposed to traditional kd-trees, a weighted Euclidean metric is developed that considers the probability of missing values during tree construction and searching. The weighted metric makes kd-trees more suitable for WSNs. The overall approach to detecting time-related changes is a non-parametric technique that makes no assumptions of the underlying distribution of the data.

Chapter 6

Anomaly detection application — An intruder detection scenario

As a proof-of-concept, part of the anomaly detection system has been implemented on an intruder detection application to demonstrate its use. In this chapter, an intruder detection application is described.

6.1 Experiment: Intruder detection

The goal of this research is to design a robust anomaly detection system that is applicable to as many applications as possible. One possible application of the developed anomaly detection system is an intruder detector. This section presents an intruder detection system as a proof-of-concept for the designed anomaly detection system.

6.1.1 Hardware platforms

The wireless sensor network consists of static sensors (Crossbow motes) and a mobile robot (Pioneer 3 robots). A Crossbow [Crossbow, 2008] mote contains a processing unit, a sensor module (see Figure 6.1), and a communication module. The processing board contains an 8-bit processor at 8 MHz, a 128 KB programming memory, and a 512 KB additional data flash memory. The wireless transmission range is around 10 meters inside a building. In future works, this communication range could be extended by using intermediate sensor

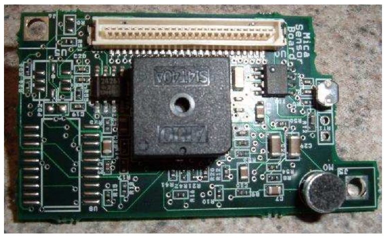


Figure 6.1: MTS300 Sensing board

motes as data routers. The sensor board has a buzzer, a light sensor, a microphone, 2 magnetometers, and 2 accelerometers. For the experiments reported in this work, only the light and sound sensing components are used. The sensor motes use TinyOS as their operating system and nesC as their programming language.

The mobile robot used in these experiments is a Pioneer 3 robot. The Pioneer 3 is a mobile robot with a two-wheel differential drive as shown in Figure 6.2. The CPU of the Pioneer robot is an Intel Pentium III processor with an 850 MHz clock rate. The mobile robot uses a Linux operating system and runs the Player-client/server device driver [Gerkey et al., 2001]. The robot uses a SICK LMS-200 range-finding laser for localization. An Orinoco wireless card is used for wireless communication with the base station at 916 MHz. The mobile robot can communicate with the sensor motes by having a mote attached to an MIB500 programming board (see Figure 6.3) through a serial connection. In the intrusion detection application, the robot runs the same Fuzzy ART program as the sensor motes. The robot takes the output from its cluster member motes and fuses them together to get the highest level representation of the environment. Thus, the mobile robot is a root mobile clusterhead with higher processing power and more sensing capabilities.

The base station is a laptop that can interact with the WSN by having a mote running the "TOS_Base" (part of TinyOS) program. This base node is attached to a programming board with a serial cable. The Deluge [Deluge, 2008] network programming system was integrated into the system, which allows a large number of sensor nodes to receive code updates at the same time over the air, instead of manually loading programs into the motes over a serial cable. Human operators can monitor the learning process of the WSN through a monitoring program. Operators can also send commands to the network. Sample commands include:

Figure 6.2: Pioneer 3 robot

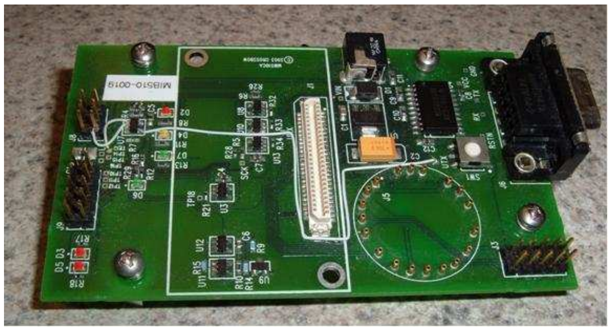


Figure 6.3: MIB510 programming board

- Set clusterhead configuration
- Set learning rate
- Set Fuzzy ART learning parameters
- Set sampling rate
- Pause the learning
- Restart the learning
- Clear Fuzzy ART's memory of categories
- Set sampling rate

- Set radio transmission power
- Set microphone sensitivity
- Report battery level

All interacting programs run on the laptop and are written in Java. Note that every sensor node runs the same Fuzzy ART program image. Sensor nodes are assigned a role, and can either be a cluster member or a clusterhead (using the clusterhead configuration commands running on the laptop).

6.1.2 Intruder detection application setting

An intruder detection system is implemented using a wireless sensor network (WSN) and a mobile robot. In order to detect abnormal events in a previously unknown environment, the sensor network first learns what is normal for the environment. Abnormal states of the environment are not kept in the sensor nodes due to memory limitations. Therefore, events that do not match the existing normal model will be treated as abnormal events by the sensor motes. When an intruder is detected, a mobile robot moves to the area to investigate. In this scenario, the robot knows the location of each cluster in advance. If the higher level clusterhead detects an anomaly (i.e., a class change after stabilization), the robot moves to the location of the cluster that detected the change. The mobile robot is the root clusterhead of the hierarchical learning system.

In order to navigate in the environment, the mobile robot first creates a laser map using Simultaneous Localization and Mapping (SLAM) [Thrun et al., 1998]. An example of the learned map is shown in Figure 6.4. After an intruder has been detected by the sensor network, the mobile robot uses a wavefront path planning algorithm to plan a path from its current position to the goal position. During motion, it localizes itself using Monte Carlo localization [Gerkey et al., 2001].

The intruder detection system is implemented on real sensor nodes along with a mobile robot and tested at both the University of Tennessee and Oak Ridge National Laboratory (ORNL). Figure 6.5 shows snapshots from the experiments at ORNL. The WSN divides into two clusters, four sensor motes in each cluster. Each cluster consists of one clusterhead and

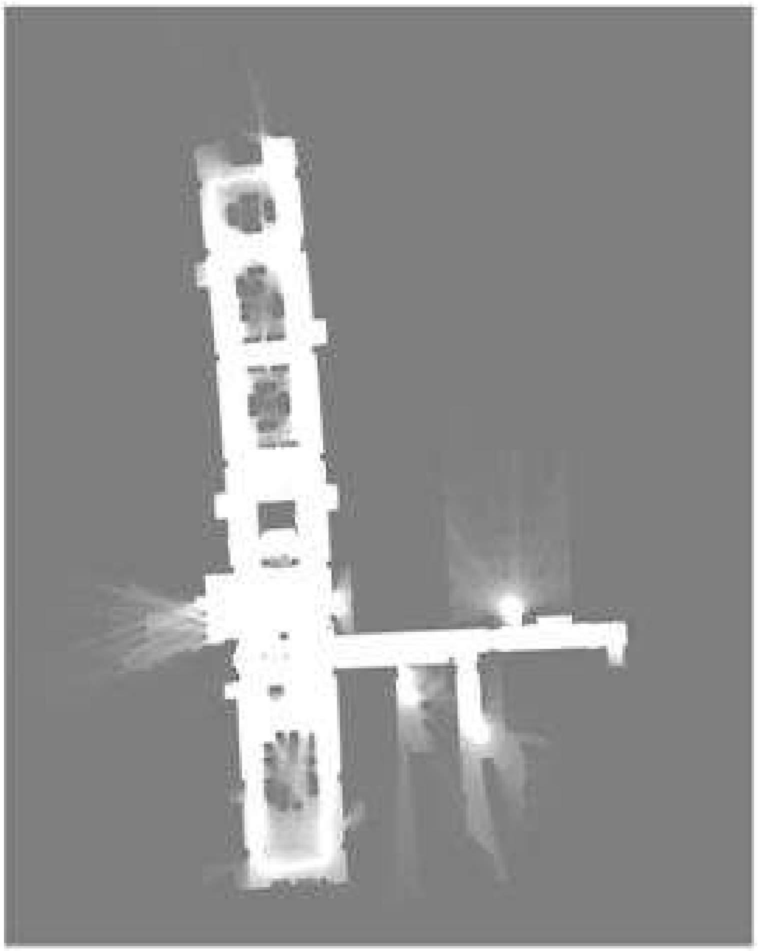


Figure 6.4: Robot-generated laser maps of Oak Ridge National Laboratory's JICS building.

three cluster members. The first cluster was deployed into a conference room of ORNL's JICS building. The second cluster was deployed in a nearby auditorium. The mobile robot is stationed in the hallway attached with the root clusterhead. The root sensor mote listens to abnormal changes. The root detects abnormal changes by learning the combination of changes of the two clusterheads (sensor motes) deployed in the two rooms. The mobile robot ran the same learning algorithm as the sensor motes, namely, the Fuzzy ART system. In the beginning, it was quiet and the lights were switched off in both rooms. The WSN has learned that "quiet" and "dark" are normal in this environment. Then, an intruder enters the conference room and turns on the lights. The WSN detects the abnormal event and notifies the robot. The robot planned a path using its wavefront path planner and

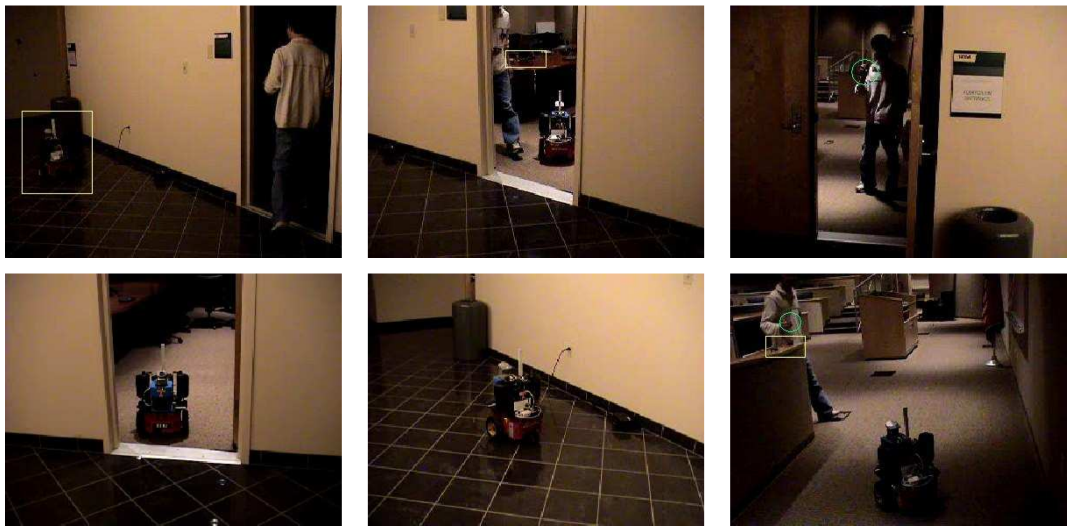


Figure 6.5: Snapshots of the intruder detection system in operation at ORNL. Motes and the mobile robot are indicated by rectangles on the picture. The sound device carried by the intruder is indicated by a circle (read left to right, top to bottom).

moved to the conference room to check on the abnormal event — "light on". The intruder then moved to the auditorium. He turned on the lights and a buzzer to make noise in the auditorium. The robot detected the abnormal activities in the auditorium — "buzzer on" and "lights on". The robot then planned a path and moved to the auditorium to check on the abnormal event. Once the robot arrived at the auditorium, it used its camera to track the intruder.

A video of this experiment is available at *http://www.cs.utk.edu/dilab/DILpicmovie.html*.

6.1.3 Performance metrics

To evaluate the anomaly detection system, statistics on the miss rate, false alarm rate, sensitivity, and specificity are collected. The miss rate is calculated as $FN/(TP+FN)$, where False Negative (FN) denotes the number of faults that the system failed to detect, and True Positive (TP) denotes the number of true faults that are detected by the system. The false alarm rate is defined as $FP/(FP+TN)$, where False Positive (FP) denotes the number of detected faults that were not true faults, and True Negative (TN) denotes number of "no faults" that were detected by the system. The sensitivity is defined as

$TP/(TP+FN)$. Ideally, the values of sensitivity and specificity are at 100%, and the false alarm rate and miss rate are at 0%.

To determine the significance of the difference in the results, the Student's T-test is applied. The assumption is that the underlying distributions of accuracies/errors are Gaussian, because of the Central Limit Theorem — as the number of testing sets approaches infinity, the distribution of the mean of accuracy/error approaches a Normal distribution. Note that the following experiments are based on synthetic data generated under lab settings. These experiments are only intended to show that by adding time analysis and a mobile robot's response, the system can improve its performances. The actual experimental result values can vary for different test data; however, the qualitative comparisons are valid, and properly illustrate the relative performances of the studied approaches.

6.1.4 Temporal change detection experiment

One of the problems of the original Fuzzy ART neural network is that it cannot detect time-related anomalies. This experiment is designed to demonstrate that the FSA model is able to detect time-related anomalies whereas the Fuzzy ART system alone cannot.

In this experiment, the system first learns the normal model; then, the testing begins. Both training and testing are performed online. All sensors sample the environment at a rate of 1 sample per second. Six sensor nodes have been used during this experiment. One sensor acts as a clusterhead, and the rest as cluster members of that node. The cluster members are uniformly deployed around the clusterhead and all cluster members are within communication range of the clusterhead. The vigilance level for cluster members is set to 0.90, which is sensitive enough to classify the environment into four distinct states: lights off and microphone off, lights on and microphone off, lights on and microphone on, and lights off and microphone on. The vigilance level for the clusterheads is set to 0.97, which is sensitive enough to sense category changes in one of the six cluster members.

The training process takes approximately 1.5 hours per trial; a total of three trials are performed. During the training period, states are visited multiple times. The average time is computed over multiple visits of the same state; this is treated as a normal environment. Two sensors are used by cluster member nodes — light and microphone. Raw light readings

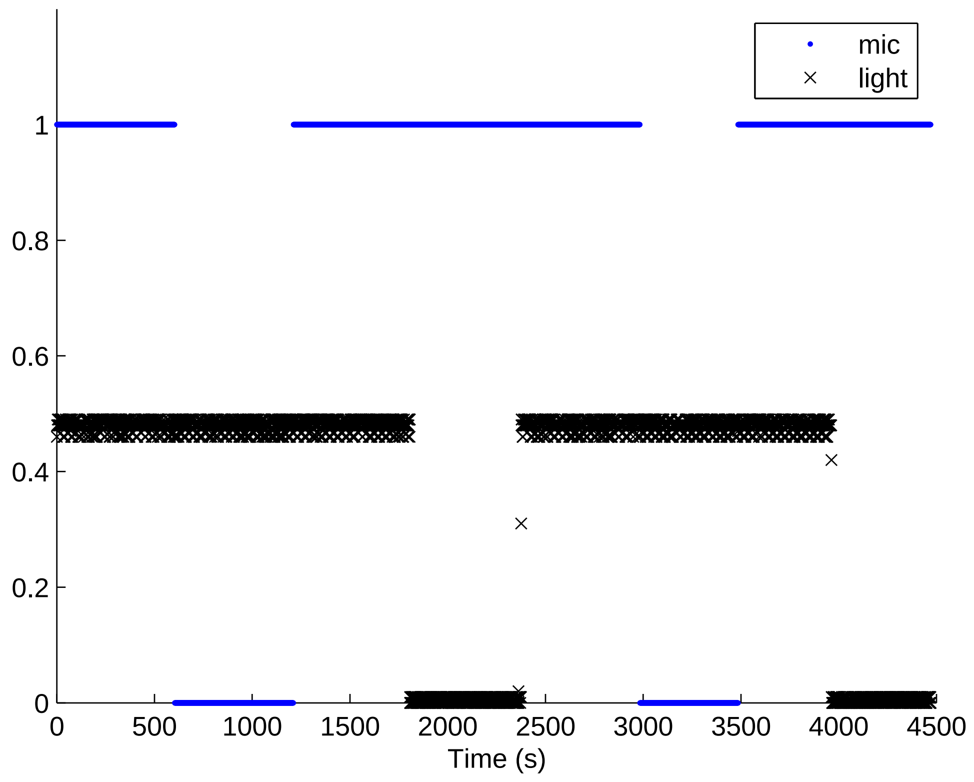


Figure 6.6: An example: normalized light and microphone readings collected by a sensor node. From time 0 to 510, the light is on (0.5), and the microphone is on (1). From time 510 to 1200, the light is on (0.5), and the microphone is on (0). From time 1200 to 1800, the light is on (0.5), and the microphone is off (1). From time 1800 to 2300, the light is off (0.1), and microphone is off (1), etc.

Table 6.1: Time duration in each state

Category	C1	C2	C3
Mean time (seconds)	571	555	538
Standard deviation (seconds)	62	75	43

between 0 and 2000 indicate dark and light, respectively. Microphone readings came from a hardware detection system onboard. The values were binary — 1 indicates no noise is detected, and 0 indicates noise is detected. A buzzer is used as a sound source, which operates at 4 KHz. The sound sensor can detect the buzzer within a radius of three to four meters in the testing environment. Figure 6.6 shows an example of the typical sensor readings collected from the environment. The light sensor readings are normalized between 0 and 1. Figure 6.7 shows the categories learned by the Fuzzy ART neural network from the data shown in Figure 6.6 during the training period. After the classification process, an FSA model has been built by using Algorithm 1. Figure 6.8 shows an FSA model built from the data shown in Figure 6.6 and Figure 6.7. Table 6.1 shows the mean and standard deviation values of the time the environment remained in each class/state before transiting to a different class/state. The figures and table show one of the trials of the experiment. In this particular example, the numbers on the FSA model are nicely rounded (e.g., 0.5 and 1). However, in more realistic situations, the Markov model can be much more complex; these experiments are designed to illustrate the developed approach.

There are three different testing suites with four trials run for each testing suite. In test suite 1, the environment starts from "light and quiet" (category 1), and remains in that state for 600 seconds. Then, it transitions to "light and noisy" (category 2), and remains in that state for 600 seconds. Finally, it transitions to "dark and noisy" (category 4), and remains in that state for 600 seconds. Note that "dark and noisy" has never occurred before during the training phase. This is an abnormal event. This testing suite only contains a new abnormal state; however, it does not include any temporal-related changes.

In test suite 2, the environment starts from "light and noisy" (category 2), and remains in that state for 600 seconds. Then, it transitions to "dark and noisy" (category 4), and remains in that state for 300 seconds. Lastly, it transitions to "dark and quiet" (category 3), and remains in that state for 600 seconds. The environment starts with abnormal

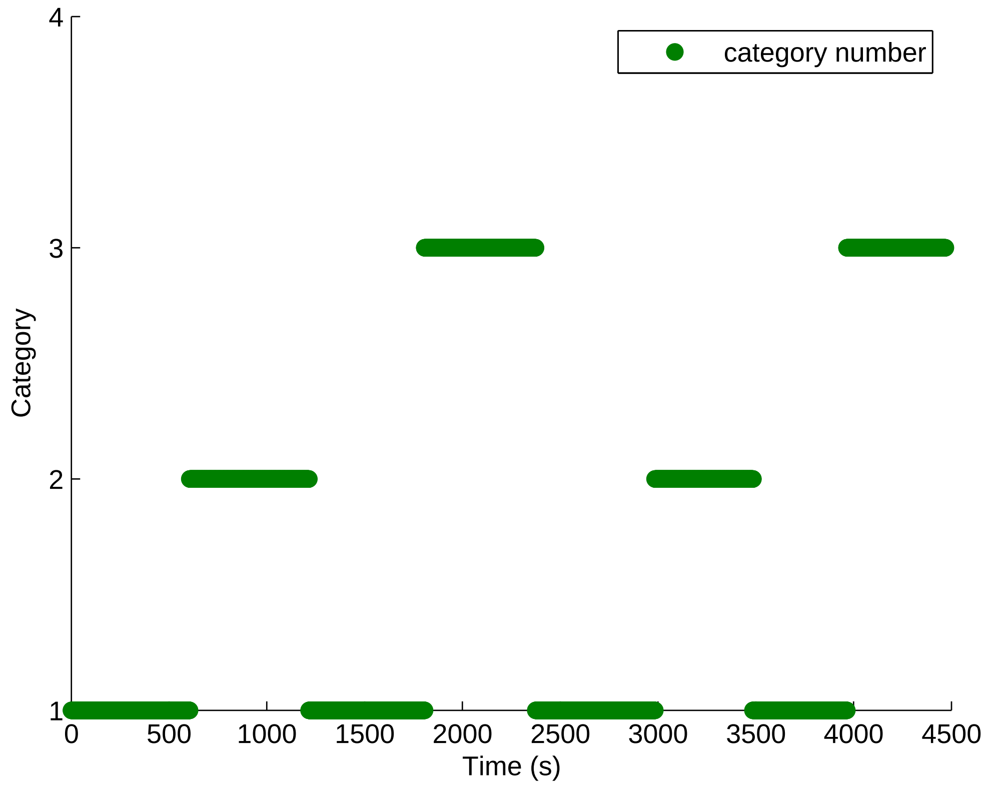


Figure 6.7: An example of the detected changes by the Fuzzy ART neural network for a cluster of six sensor nodes during the training phase. The input is the sensory data shown in Figure 6.6.

transitions to state 2, then the abnormal state 4 is detected. Lastly, an abnormal transition occurs from abnormal state 4 to state 3. This testing suite contains both abnormal events of a new abnormal state and abnormal time transitions.

In test suite 3, the environment starts from "light and quiet" (category 1), and remains in that state for 300 seconds. Then, it transitions to "dark and quiet" (category 3), and remains in that state for 900 seconds. The environment abnormally remains at state 1 too briefly and in state 3 for a longer period of time. This testing suite only contains time-related abnormal changes.

These testing suites are used to compare the performance of the basic Fuzzy ART system (Kulakov and Davcev's implementation) and the enhanced Fuzzy ART system. The

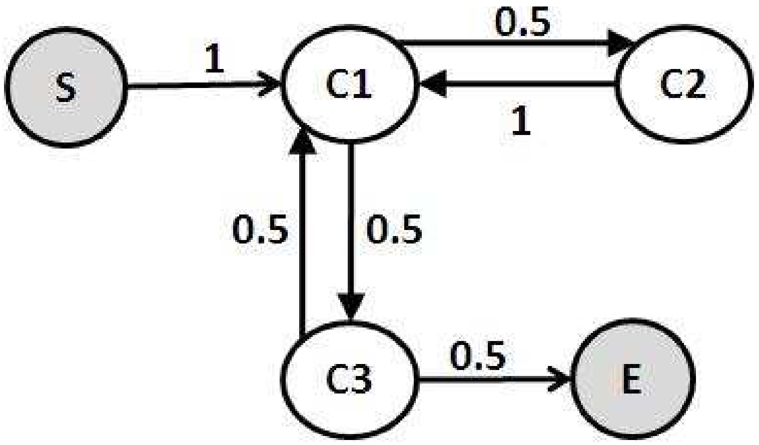


Figure 6.8: An example of a learned FSA model for the training phase. The model was the normal model of the environment. States “S” and “E” denote the start and end states, respectively. They were manually added to the system. State C1 denotes lights were on and buzzer was off. State C2 denotes lights were on and buzzer was on. State C3 denotes lights were off and buzzer was off.

experimental results are shown in Table 6.2, which are averaged over three testing suites (for a total of 12 trials). Approximately 1500 observations were made from each sensor for each trial. The experimental results illustrate that the enhanced Fuzzy ART system is able to detect more anomalies than the original Fuzzy ART system (i.e., approximately 86% vs. 41%). This is due to the fact that the enhanced system learns a time series, and is able to detect time-related anomalies, whereas the original Fuzzy ART system cannot. Both the Fuzzy ART system and the enhanced Fuzzy ART system have a low false alarm rate (approximately 6%). To determine the significance of these results, I have applied the Student’s T-test to the miss rate and sensitivity results for the original Fuzzy ART system and the enhanced Fuzzy ART system. The student’s T-test confirms that the differences in these results are statistically significant, with a confidence level of 99.5%. Thus, the enhanced Fuzzy ART approach provides a significant improvement over the original Fuzzy ART approach.

In these experiments, the time duration in each state is manually selected to illustrate the concept. The time duration could be very different in practical applications, such as

Table 6.2: Performance evaluation between the basic Fuzzy ART and enhanced Fuzzy ART

		False Alarm	Miss Rate	Sensitivity	Specificity
Original	mean	6%	59%	41%	94%
Fuzzy ART	stdev	12	38	38	12
Enhanced	mean	6%	14%	86%	94%
Fuzzy ART	stdev	12	20	20	12

for comparisons between daytime versus nighttime expectations. However, in general, the developed FSA model would be implemented similarly. Additionally, in some applications, if the duration of time within a state is not of interest, but, instead, the order of the states is what is most important, the system would not have to maintain the time duration in each state. For example, if people always turn on the light before making noise in the room, regardless of the time duration in each state, then making noise in the dark room would be abnormal. It can instead simply keep track of the expected state transitions. As mentioned before, the exact values obtained from this experiment are not important; if more time-related changes are introduced in the experiments, then the miss rate would increase for the Fuzzy ART system. However, the relative performance comparisons are valid. The purpose of this experiment is to show that by enabling the system to detect time-related anomalies, the system is able to improve its performance.

6.1.5 Intruder detection application using mobile and sensor network

Unlike many other wireless sensor network applications, in this experiment, a mobile robot is added to the anomaly detection system. This is because mobile robots are expected to bring mobility and additional computation and sensor capabilities to make the detection system more robust. The following experiment is designed to prove that including a mobile robot can make the detection system more robust.

When a change is detected in the environment, it does not necessarily mean that an intruder caused the anomaly. To determine if the anomaly is caused by an intruder, a mobile robot is sent to investigate using an additional sensor (i.e., a camera). In the enhanced intruder detection system, the sensor motes run the developed anomaly detection system, and a mobile robot serves as the root clusterhead. Once a change is detected by the mobile robot, it travels to the area and checks for an intruder using the camera mounted on the top of the robot. The camera tracks the intruder using a motion tracking program. The motion tracking program only detects moving objects. Note that the mobile robot should carry its own light source or use thermal images to detect a human in any lighting conditions, rather than just a moving object.

In general, if the mobile robot detects the intruder, the alarm is confirmed by the mobile robot. However, if the robot does not detect any (human) intruder within 120 seconds, it turns off the alarm and claims that there is no change in the environment. Therefore, the robot does not miss any abnormal events occurring in the environment and at the same time reduces false alarms.

The intruder detection system is run for the same sets of experiments in Section 6.1.4 except the abnormal state number 4 (“light off and noise”) which is not caused by an intruder. Instead, it is a normal state of the environment that never occurred in the initial learning process.

The performances of the basic Fuzzy ART system, the enhanced Fuzzy ART system, and the enhanced Fuzzy ART system with intelligent mobile robot responder are compared. The results are shown in Table 6.3, which are averaged over three testing suites for a total of 12 trials. The Student’s T-test is applied to the miss rate and sensitivity results for the original Fuzzy ART and the enhanced Fuzzy ART. The Student’s T-test is also applied to the false alarm rate and specificity for the enhanced Fuzzy ART and the enhanced Fuzzy ART system with the intelligent mobile robot responder. The tests confirmed that the differences in the results are statistically significant, with a confidence level of 99.5%. The experimental results illustrate that the enhanced Fuzzy ART system and the enhanced system with mobile robot is able to detect more anomalies than the original Fuzzy ART system (i.e., 83% vs. 30%). This is due to the fact that the enhanced system learns a time series and is able to detect time-related anomalies, whereas the original Fuzzy ART cannot. The enhanced Fuzzy ART system with intelligent mobile robot responder approach is able to reduce the false alarms compared to original Fuzzy ART system and the enhanced Fuzzy ART system (i.e., 26% vs. 46%). Thus, the enhanced Fuzzy ART with

Table 6.3: Performance evaluation of the intruder detection system

		False Alarm	Miss Rate	Sensitivity	Specificity
Original Fuzzy ART	mean	46%	70%	30%	54%
	stdev	36	42	42	36
Enhanced Fuzzy ART	mean	46%	17%	83%	54%
	stdev	36	20	25	40
Enhanced Fuzzy ART with mobile robot	mean	26%	17%	83%	74%
	stdev	19	25	24	19

mobile robot approach provides a significant improvement both in miss rate and false alarm rate over the original Fuzzy ART approach. Again, the specific values of the experiment are meaningless; the missing false alarm rate is influenced by the numbers of "artificial" intruders introduced. This experiment is intended to show that the anomaly detection system could reduce the false alarm rate by incorporating the mobile robot's feedback.

The expectation is that if the mobile robot could provide feedback to the sensor motes regarding false alarms and the motes could correct their learning models based on this information, then the detection performance could be further improved. Additionally, the mobile robots could save their battery power by avoiding repeated checks of similar false alarms. Thus, in future work (not part of this dissertation), the detection system could be enhanced by adding a feedback loop to the learning model, enabling learning from false alarms.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

This dissertation makes several contributions to anomaly detection in an unknown environment using Wireless Sensor Networks. The most important contribution is the design of anomaly detection — a novel approach that detects both sensor anomalies and time-related anomalies in an online, unsupervised, and distributed fashion. The distributed learning algorithms used are particularly suitable in a hierarchical, resource-constrained sensor network for environment monitoring purposes. The system utilizes various machine learning algorithms to model normal sensor data; this model is then used to detect anomalies. In contrast to most learning algorithms, the algorithms that are developed do not require large computational time or space and maintain high quality system performance. In some versions of this work, a mobile robot serves as the root clusterhead of the sensor network. Upon detection of an anomaly, this mobile robot can respond to further investigate the anomaly. In addition, the learning algorithms can take advantage of time and/or space correlations in the sensor data.

This approach has been shown to have the following characteristics:

- A hierarchical Fuzzy ART learning structure is appropriate for distributed anomaly detection in a WSN. In the future, sensor motes will have more computational power; the sensor nodes will also be smaller in size as compared to the present day sensor nodes but with the same processing/communication capabilities that the current sensor nodes possess. Therefore, simple and effective algorithms like the one developed in this research will continue to be needed for WSNs.

- A two-step temporal modeling procedure can be used to analyze and extract semantic symbols from a sequence of observations. The algorithm is distributed, and supports a hierarchical learning structure, which would scale to a large number of sensors and will be practical for resource constrained sensor motes.

- The system detects time-related changes online by using a likelihood-ratio detection scheme. The developed temporal modeling technique is able to capture high-order temporal dependencies in some parts of the behavior and lower-order dependencies elsewhere.

- An iterative temporal learning approach captures the temporal dependencies in data and removes redundancies, which should translate into energy savings in the WSN.

- A non-parametric nearest neighbor missing data imputation technique for WSNs utilizes temporal and spatial information from the environment to estimate missing values.

- A kd-tree data structure is used to organize temporally and spatially correlated data, which is able to speed up the search by localizing it. To make the traditional kd-tree more suitable for WSNs, a novel weighted variance and weighted Euclidean distance is introduced to give designers more control of the tree structure.

Using resource constrained WSNs for environment monitoring of applications such as volcanic eruptions is challenging, because the event of interest is usually preceded by a long period of inactivity and the event itself lasts only for a short period of time. The main objective of this research is to design a distributed anomaly detection system in an unknown environment using a WSN. The detection system first detects sensory level anomalies by using the Fuzzy ART neural network. The classifier is also used to classify multi-dimensional sensor data into discrete classes, where each class label represents a higher semantic meaning of the sensory data. The labeled classes form a sequence of classes over a time period. Then, the system detects time-related changes by using a likelihood-ratio detection scheme.

Specifically, a symbol compressor is used to extract temporal semantics out of the original time sequence, a PST is then constructed from the compressed temporal sequence, and then a likelihood-ratio detector detects time-related anomalies in WSNs. The developed temporal modeling technique is able to capture high-order temporal dependencies in some parts of the behavior and lower-order dependencies elsewhere. The developed approach is verified using a volcano monitoring data set. Results show that the detection system yields a high performance. The iterative temporal learning approach captures the temporal dependencies in data and removes redundancies, which potentially saves the transmission and processing powers in the WSN. To make the learning more robust, the system uses a novel nearest neighbor imputation technique that takes advantages of spatial and temporal correlations in the sensor data. The NN imputation uses a kd-tree to organize time and space correlated data. A weighted variance and Euclidean distance metric is introduced to make the kd-tree more suitable for the missing data problem in WSNs. All learning algorithms are distributed, support a hierarchical learning structure that scales to a large number of sensors and will be practical for resource constrained sensor motes.

7.2 Future works

More work can be done to refine and extend the system. Several promising research directions based on this work are as follows:

- Implement and evaluate the time analysis module and the missing data imputation module on a real wireless sensor network rather than a single sensor on a single node.
- Incorporate an efficient data structure, (e.g., kd-tree) in the Fuzzy ART neural network. When a new observation presented to Fuzzy ART's neural network, the system compares the observation with all available prototypes in the network and finds its closest match. Instead of a brute-force search, the system can use a kd-tree to organize the learned prototypes and perform nearest neighbor search. By using an efficient data structure, the searching time can be saved, which in turn saves power in resource constrained nodes. The challenge here is to determine how to build the kd-tree using Fuzzy rules. Additionally, if such a tree can be built, the system can

combine the developed missing data module with the Fuzzy ART module to save programming space and memory on the sensor motes.

- Incorporate Collaborative Filtering (CF) into the detection process. In contrast to the existing approach that relies on local processing, different ways can be explored to improve the decision making process by sharing information from other sensor nodes/robots.
- Enable the system to adapt to feedback from upper level nodes, robots or a human operator. An important way to improve the detection performance is to incorporate feedback from other sources.

Bibliography

Bibliography

[Aha, 1992] Aha, D. (1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36:267–287.

[Akyildiz et al., 2002] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422.

[Amelia II, 2010] Amelia II (2010). Missing data imputation software. http://gking.harvard.edu/amelia/.

[Apostolico and Bejerano, 2000] Apostolico, A. and Bejerano, G. (2000). Optimal amnesic probabilistic automata or how to learn and classify proteins in linear time and space. In *Proceedings of the fourth annual international conference on Computational molecular biology*, pages 25–32, Tokyo, Japan. ACM.

[Banerjee et al., 2005] Banerjee, S., Grosan, C., and Abraham, A. (2005). Ideas: intrusion detection based on emotional ants for sensors. In *The 5th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 344–349, Wroclaw, Poland.

[Barrenetxea et al., 2008] Barrenetxea, G., Ingelrest, F., Schaefer, G., and Vetterli, M. (2008). Wireless Sensor Networks for Environmental Monitoring: The SensorScope Experience. In *The 20th IEEE International Zurich Seminar on Communications (IZS 2008)*.

[Begleiter and Yona, 2004] Begleiter, R. and Yona, G. (2004). On prediction using variable order markov models. *Journal of Artificial Intelligence Research*, 22:385–421.

[Bentley, 1975] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.

[Boyan and Littman, 1994] Boyan, J. A. and Littman, M. L. (1994). Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems 6*, pages 671–678. Morgan Kaufmann.

[Branch et al., 2006] Branch, J., Szymanski, B., Giannella, C., Wolff, R., and Kargupta, H. (2006). In-network outlier detection in wireless sensor networks. In *Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS)*, pages 102–111. IEEE Computer Society.

[Caro et al., 2005] Caro, G. D., Caro, G. D., Ducatelle, F., Ducatelle, F., Gambardella, L. M., and Gambardella, L. M. (2005). Anthocnet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 16:443–455.

[Carpenter and Grossberg, 1991] Carpenter, G. and Grossberg, S. (1991). Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771.

[Chen and Shao, 2001] Chen, J. and Shao, J. (2001). Jackknife variance estimation for nearest-neighbor imputation. *Journal of the American Statistical Association*, 96:260—269.

[Chu et al., 2006] Chu, D., Deshpande, A., Hellerstein, J. M., and Hong, W. (2006). Approximate data collection in sensor networks using probabilistic models. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*, page 48, Washington, DC, USA. IEEE Computer Society.

[Crossbow, 2008] Crossbow (2008). Crossbow technology inc. http://www.xbow.com/.

[D'Costa et al., 2004] D'Costa, A., Ramachandran, V., and Sayeed, A. (2004). Distributed classification of gaussian space-time sources in wireless sensor networks. *IEEE Journal of Selected Areas in Communications*, 22(6):1026 – 1036.

[Deluge, 2008] Deluge (2008). Network programming system. http://www.cs.berkeley.edu/ jwhui/research/deluge/.

[Di Caro and Dorigo, 1998] Di Caro, G. and Dorigo, M. (1998). Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365.

[Dowling et al., 2005] Dowling, J., Curran, E., Cunningham, R., and Cahill, V. (2005). Using feedback in collaborative reinforcement learning to adaptively optimize manet routing. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35:360–372.

[Duarte and Hu, 2004] Duarte, M. F. and Hu, Y. H. (2004). Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838.

[Egorova-Förster and Murphy, 2007] Egorova-Förster, A. and Murphy, A. L. (2007). A feedback-enhanced learning approach for routing in wsn. In *Proceedings of the 4th Workshop on Mobile Ad-Hoc Networks (WMAN)*, Bern, Switzerland. Springer-Verlag.

[Eskin et al., 2002] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security*. Kluwer.

[Fix and Hodges, 1951] Fix, E. and Hodges, J. (1951). Discriminatory analysis: Nonparametric discrimination: Consistency properties. Technical Report 4, USAF School of Aviation Medicine, Randolf Field, Texas.

[Fletcher et al., 2004] Fletcher, A. K., Rangan, S., and Goyal, V. K. (2004). Estimation from lossy sensor data: jump linear modeling and Kalman filtering. In *The third international symposium on Information processing in sensor networks (IPSN)*, pages 251–258, New York, NY. ACM Press.

[Fox, 2008] Fox, J. (2008). *Applied regression analysis and generalized linear models*. SAGE Publications Inc., California, USA.

[Friedman et al., 1977] Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226.

[Gerkey et al., 2001] Gerkey, B. P., Vaughan, R. T., Stoy, K., Howard, A., Sukhatme, G. S., and Mataric, M. J. (2001). Most valuable player: a robot device server for distributed control. In *The IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1226–1231.

[Granger et al., 2000] Granger, E., Rubin, M. A., Grossberg, S., and Lavoie, P. (2000). Classification of incomplete data using the fuzzy artmap neural network. In *The IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, volume 6, pages 35–40, Washington, DC.

[Guestrin et al., 2004] Guestrin, C., Bodk, P., Thibaux, R., Paskin, M. A., and Madden, S. (2004). Distributed regression: an efficient framework for modeling sensor network data. In *Proceedings of the Third International Symposium on Information Processing in Sensor Networks (IPSN 2004)*, Berkeley, California.

[Hai et al., 2007] Hai, T. H., Khan, F., and nam Huh, E. (2007). Hybrid intrusion detection system for wireless sensor networks. *Computational Science and Its Applications ICCSA*, 4706:383–396.

[Heinzelman et al., 2000] Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. (2000). Energy-efficient communication protocols for wireless microsensor networks. In *International Conference on System Sciences*, Maui, HI.

[Holden and Freitas, 2004] Holden, N. and Freitas, A. A. (2004). Web page classification with an ant colony algorithm. In *Parallel Problem Solving from Nature - PPSN VIII, LNCS 3242*, pages 1092–1102. Springer-Verlag.

[Hughey and Berry, 2000] Hughey, M. and Berry, M. (2000). Improved query matching using kd-trees: A latent semantic indexing enhancement. *Information Retrieval*, 2(4):287–302.

[Jain et al., 2004] Jain, A., Chang, E. Y., and Wang, Y.-F. (2004). Adaptive stream management using Kalman Filters. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 11–22, Paris, France. ACM.

[Janakiram et al., 2006] Janakiram, D., Reddy, V., and Kumar, A. (2006). Outlier detection in wireless sensor networks using Bayesian belief networks. In *First International Conference on Communication System Software and Middleware*, pages 1–6.

[Jonsson and Wohlin, 2004] Jonsson, P. and Wohlin, C. (2004). An evaluation of k-nearest neighbour imputation using likert data. In *Proceedings of the 10th International Symposium on Software Metrics*, pages 108 – 118, Washington, DC. IEEE Computer Society.

[Jr. et al., 2007] Jr., E. R. H., Hruschka, E. R., and Ebecken, N. F. (2007). Bayesian networks for imputation in classification problems. *Journal of Intelligent Information Systems*, 29:231–252.

[Karl and Willig, 2005] Karl, H. and Willig, A. (2005). *Protocols and Architectures for Wireless Sensor Networks.* John Wiley and Sons, West Sussex, England.

[Kedem and Fokianos, 2002] Kedem, B. and Fokianos, K. (2002). *Regression Models for Time Series Analysis.* John Wiley & Sons, Inc., New Jersey, USA.

[Kulakov and Davcev, 2005] Kulakov, A. and Davcev, D. (2005). Tracking of unusual events in wireless sensor networks based on artificial neural-network algorithms. In *Information Technology: Coding and Computing (ITCC)*, volume 2, pages 534–539.

[Kumar and Miikkulainen, 1997] Kumar, S. and Miikkulainen, R. (1997). Dual reinforcement Q-routing: An on-line adaptive routing algorithm. In *Proceedings of the Artificial Neural Networks in Engineering Conference*, pages 231–238, St. Loius, USA. ASME Press.

[Kurtz, 1999] Kurtz, S. (1999). Reducing the space requirement of suffix trees. *Software–Practice & Experience*, 29:1149–1171.

[Langford and Zadrozny, 2005] Langford, J. and Zadrozny, B. (2005). Relating reinforcement learning performance to classification performance. In *Proceedings of the 22nd* *international conference on Machine learning (ICML '05)*, pages 473–480, New York, NY, USA. ACM.

[Li and Liu, 2007] Li, M. and Liu, Y. (2007). Underground structure monitoring with wireless sensor networks. In *Processing of the 6th International Information Symposium on Sensor Networks (IPSN)*, pages 69–78.

[Li and Parker, 2008] Li, Y. and Parker, L. E. (2008). A spatial-temporal imputation technique for classification with missing data in a wireless sensor network. In *IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems (IROS)*, Nice, France.

[Lim and Shin, 2005] Lim, J.-J. and Shin, K. G. (2005). Energy-efficient self-adapting online linear forecasting for wireless sensor network applications. In *IEEE International Conference on Mobile Adhoc and Sensor Systems*, Washington, DC.

[Lin et al., 2004] Lin, J., Keogh, E., Lonardi, S., Lankford, J., and Nystrom, D. (2004). Visually mining and monitoring massive time series. In *Proceedings of the 10th ACM international conference on Knowledge discovery and data mining*, pages 460–469. ACM.

[Little and Rubin, 1986] Little, R. J. and Rubin, D. B. (1986). *Statistical analysis with missing data*. John Wiley & Sons, Inc., New York, NY.

[Loo et al., 2006] Loo, C. E., Ng, M. Y., Leckie, C., and Palaniswami, M. (2006). Intrusion detection for routing attacks in sensor networks. *International Journal of Distributed Sensor Networks*, 2:313–332.

[Mainwaring et al., 2002] Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. (2002). Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA)*, pages 88–97, Atlanta, Georgia, USA. ACM.

[Manjhi et al., 2005] Manjhi, A., Nath, S., and Gibbons, P. B. (2005). Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *ACM SIGMOD/PODS 2005 Conference*, Baltimore, Maryland.

[Marrón et al., 2006] Marrón, P. J., Sauter, R., Saukh, O., Gauger, M., and Rothermel, K. (2006). Challenges of complex data processing in real world sensor network deployments. In *Proceedings of the workshop on Real-world wireless sensor networks REALWSN '06*, pages 43–47, Uppsala, Sweden. ACM.

[Matin and Hussain, 2006] Matin, A. W. and Hussain, S. (2006). Intelligent hierarchical cluster-based routing. In *Proceedings of the International Workshop on Mobility and Scalability in Wireless Sensor Networks (MSWSN)*, San Francisco, CA.

[Mazeroff et al., 2008] Mazeroff, G., Gregor, J., Thomason, M., and Ford, R. (2008). Probabilistic suffix models for api sequence analysis of windows xp applications. *Pattern Recognition*, 41(1):90–101.

[Moore, 1990] Moore, A. (1990). *Efficient Memory-Based Learning for Robot Control*. PhD thesis, Computer Laboratory, University of Cambridge, Cambridge, UK.

[Naumowicz et al., 2008] Naumowicz, T., Freeman, R., Heil, A., Calsyn, M., Hellmich, E., Brändle, A., Guilford, T., and Schiller, J. (2008). Autonomous monitoring of vulnerable habitats using a wireless sensor network. In *Proceedings of the workshop on Real-world wireless sensor networks (REALWSN)*, pages 51–55, Glasgow, Scotland. ACM.

[Oida and Sekido, 1999] Oida, K. and Sekido, M. (1999). An agent-based routing system for qos guarantees. *Conference Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC '99)*, 3:833–838.

[Omohundro, 1987] Omohundro, S. M. (1987). Efficient algorithms with neural network behavior. *Complex Systems*, 1:2:273–347.

[Onat and Miri, 2005] Onat, I. and Miri, A. (2005). An intrusion detection system for wireless sensor networks. In *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications*, Los Alamitos. IEEE Computer Society Press.

[Peng et al., 2006] Peng, W.-C., Ko, Y.-Z., and Lee, W.-C. (2006). On mining moving patterns for object tracking sensor networks. In *Proceedings of the 7th International*

Conference on Mobile Data Management (MDM '06), page 41, Washington, DC, USA. IEEE Computer Society.

[Polastre et al., 2004] Polastre, J., Szewczyk, R., Mainwaring, A., Culler, D., and Anderson, J. (2004). Analysis of wireless sensor networks for habitat monitoring. *Wireless sensor networks*, 1:399–423.

[Rabbat and Nowak, 2004] Rabbat, M. and Nowak, R. (2004). Distributed optimization in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks (IPSN '04)*, pages 20–27, New York, NY, USA. ACM.

[Rancourt, 1999] Rancourt, E. (1999). Estimation with nearest-neighbor imputation at statistics canada. In *Proceedings of the Survey Research Methods Section*, pages 131–138. American Statistical Association.

[Reynolds, 1997] Reynolds, D. A. (1997). Comparison of background normalization methods for text-independent speaker verification. In *The 5th European Conference on Speech Communication and Technology*, pages 963 – 966, Rhodes, Greece.

[Reynolds et al., 2000] Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. (2000). Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10:19–41.

[Ron et al., 1996] Ron, D., Singer, Y., and Tishby, N. (1996). The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149.

[Rubin, 1987] Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys.* Wiley, New York.

[Santini et al., 2008] Santini, S., Ostermaier, B., and Vitaletti, A. (2008). First experiences using wireless sensor networks for noise pollution monitoring. In *Proceedings of the workshop on Real-world wireless sensor networks (REALWSN)*, pages 61–65, Glasgow, Scotland. ACM.

[Sapojnikova, 2003] Sapojnikova, E. (2003). *ART-based Fuzzy Classifiers: ART Fuzzy Networks for Automatic Classification.* PhD thesis, Computer and Information Science, University of Tubingen, Tubingen, Germany.

[Sayood, 2000] Sayood, K. (2000). *Introduction to Data Compression.* Morgan Kaufmann.

[Schulz et al., 2008] Schulz, J., Reichenbach, F., Blumenthal, J., and Timmermann, D. (2008). Low cost system for detecting leakages along artificial dikes with wireless sensor networks. In *Proceedings of the workshop on Real-world wireless sensor networks (REALWSN)*, pages 66–70, Glasgow, Scotland. ACM.

[Shen and Jaikaeo, 2005] Shen, C.-C. and Jaikaeo, C. (2005). Ad hoc multicast routing algorithm with swarm intelligence. *Moble Network Applications*, 10(1-2):47–59.

[Stoianov et al., 2007] Stoianov, I., Nachman, L., Madden, S., and Tokmouline, T. (2007). Pipenet: A wireless sensor network for pipeline monitoring. volume 1, pages 264–273, Cambridge, Massachusetts, USA. ACM.

[Stone, 2000] Stone, P. (2000). TPOT-RL applied to network routing. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML '00)*, pages 935–942, San Francisco, CA. Morgan Kaufmann Publishers Inc.

[Subramanian et al., 1997] Subramanian, D., Druschel, P., and Chen, J. (1997). Ants and reinforcement learning: A case study in routing in dynamic networks. In *The 15th Joint Conference on Artifical Intelligence (IJCAI)*, pages 832–839. The MIT Press.

[Sun et al., 2002] Sun, R., Tatsumi, S., and Zhao, G. (2002). Q-MAP: a novel multicast routing method in wireless ad hoc networks with multiagent reinforcement learning. In *2002 IEEE Region 10 Technical Conference on Computers, Communications, Control and Power Engineering*, volume 1, pages 667–670.

[Sun and Deogun, 2004] Sun, Z. and Deogun, J. S. (2004). Local prediction approach for protein classification using probabilistic suffix trees. In *Proceedings of the second conference on Asia-Pacific bioinformatics*, volume 29, pages 357 – 362, Dunedin, New Zealand. Australian Computer Society, Inc.

[Techateerawat and Jennings, 2006] Techateerawat, P. and Jennings, A. (2006). Energy efficiency of intrusion detection systems in wireless sensor networks. In *Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology (WI-IATW)*, pages 227–230, Washington, DC, USA. IEEE Computer Society.

[Thrun et al., 1998] Thrun, S., Fox, D., and Burgard, W. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5(3-4):253–271.

[Tsai et al., 2007] Tsai, H., Yang, D., Peng, W., and Chen, M. (2007). Exploring group moving pattern for an energy-constrained object tracking sensor network. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 825 – 832, Nanjing, China. Springer.

[Tulone and Madden, 2006] Tulone, D. and Madden, S. (2006). PAQ: Time series forecasting for approximate query answering in sensor networks. In *The 3rd European Workshop on Wireless Sensor Networks (EWSN)*, volume 3868, pages 21–37, ETH Zurich, Switzerland. Springer.

[Wang and Krishnamurthy, 2008] Wang, A. and Krishnamurthy, V. (2008). Signal interpretation of multifunction radars: Modeling and statistical signal processing with stochastic context free grammar. *IEEE Transactions on Signal Processing*, 56(3):1106–1119.

[WEKA, 2010] WEKA (2010). Machine learning software. http://www.cs.waikato.ac.nz/ml/weka/.

[Welch, 1984] Welch, T. A. (1984). A technique for high-performance data compression. *IEEE Computer*, 17:8–19.

[Werner-Allen et al., 2005] Werner-Allen, G., Johnson, J., Ruiz, M., Lees, J., and Welsh, M. (2005). Monitoring volcanic eruptions with a wireless sensor network. In *Proceeedings of the Second European Workshop on Wireless Sensor Networks*, volume 1, pages 108–120, Istanbul, Turkey.

[Werner-Allen et al., 2006] Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., and Welsh, M. (2006). Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th symposium on Operating systems design and implementation (OSDI)*, pages 381–396, Berkeley, CA, USA. USENIX Association.

[Xu et al., 2004] Xu, N., Rangwala, S., Chintalapudi, K. K., Ganesan, D., Broad, A., Govindan, R., and Estrin, D. (2004). A wireless sensor network for structural monitoring. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, pages 13–24, New York, NY, USA. ACM.

[Zarchan and Musoff, 2000] Zarchan, P. and Musoff, H. (2000). *Fundamentals of Kalman Filtering: A Practical Approach.* AIAA (American Institute of Aeronautics & Ast), Massachusetts.

[Zhao et al., 2003] Zhao, J., Govindan, R., and Estrin, D. (2003). Computing aggregates for monitoring wireless sensor networks. In *The 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, Anchorage, Alaska.

Vita

YuanYuan Li was born in P.R. China, on June 6, 1981. She came to U.S. alone when she was 15. After graduating in 1998 from Totino-Grace High School, Fridley, Minnesota, she attended Minnesota State University (MNSU) in Mankato where she received a Bachelor of Science degree, *cum laude*, in 2000 from the Computer Science department. She continued her graduate study at MNSU until 2003. She transferred to the University of Tennessee, Knoxville to complete her Master's degree in Computer Science in 2006. During the fall of 2003, she joined the Distributed Intelligent Laboratory (DILab) where she completed her Doctor of Philosophy degree in 2010. She is a member of IEEE and ACM. She has been a reviewer for IEEE/RSJ International Conference on intelligent Robots and Systems (IROS), IEEE International Conference on Robotics and Automation (ICRA) and Journal of Robotics and Autonomous Systems (JRAS). Her research interests are in cooperative multi-robot systems, distributed robotics, wireless sensor networks and artificial intelligence.