2011-01-01

Assessing Data Quality In A Sensor Network For Environmental Monitoring

Gesuri Ramirez
University of Texas at El Paso, gesuri@gmail.com

ASSESSING DATA QUALITY IN A SENSOR NETWORK

FOR ENVIRONMENTAL MONITORING

GESURI RAMIREZ GARCIA

Department of Computer Science

APPROVED:

Olac Fuentes, Chair, Ph.D.

Craig E. Tweedie, Ph.D.

Christopher Kiekintveld, Ph.D.

Benjamin C. Flores, Ph.D.
Dean of the Graduate School

A la memoria de mi abuelo

Lucio Ramírez Marín.

ASSESSING DATA QUALITY IN A SENSOR NETWORK FOR ENVIRONMENTAL MONITORING

by

GESURI RAMIREZ GARCIA

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

December 2011

Acknowledgements

I want to thank God first of all for all His invaluable daily guidance.

I would like to thank Dr. Olac Fuentes from the Department of Computer Science, for his complete faith and support in me.

I would also like to thank Dr. Craig E. Tweedie, from the Department of Biology and Environmental Science, for his comments, vital additional guidance regarding my research, and for his steadfast support during the course of my Master's studies that was greatly needed and deeply appreciated.

I am also thankful to Dr. Christopher Kiekintveld for his contributions and the valuable time he dedicated to this work.

Additionally, I want to thank the Computer Science Department professors and staff, Systems Ecology Lab, and the Sharing Resources to Advance Research and Education through Cyberinfrastructure (CyberShARE) Center of Excellence. I spent many good moments and experiences with so many different people in those buildings that it's impossible to thank every single person here; however, I hope this acknowledgment will be accepted as a sign of my gratitude and respect.

I especially want to thank my fiancé, Myriam, for her constant efforts, never-ending patience, and for inspiring me to work harder.

I want to thank all my friends in El Paso and Puebla because somehow they have always found the way to support me by either listening or counseling me, making me laugh,

going on adventures with me, or simply being there for me.

I also want to thank my family. My parents (Gloria and Abisai) and my brothers (Geovany, Libna, Dilan, Abi, and Aranza) are the foundation of my life; they have given me values that will stay with me forever and that played a key role in completing this thesis. All the love and support I got from them is beyond measure and I can only pay them back by loving them back and making them proud. I also want to thank the rest of my family- there are no words to thank you enough for your support. I am now eager and hopeful to see the youngest members of my family take on quests of their own. To all of you, thank you.

Finally, I offer my thanks and regards to all of those who supported me in any respect during the completion of this thesis.

This material is based upon work supported in part by the National Science Foundation (NSF) under CREST Grant No. HRD-0734825. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

NOTE: This thesis was submitted to my Supervising Committee on December 15, 2011.

Abstract

Assessing the quality of sensor data in environmental monitoring applications is important, as erroneous readings produced by malfunctioning sensors, calibration drift, and problematic climatic conditions, such as icing or dust, are common.Traditional data quality checking and correction is a painstaking manual process, so the development of automatic systems for this task is highly desirable.

This study investigates machine learning methods to identify and clean incorrect data from a real-world environmental sensor network, the Jornada Experimental Range, located in Southern New Mexico. We evaluated several learning algorithms and data replacement schemes, and developed a method to identify the problematic sensor. The evidence found and its analysis allowed us to conclude that learning algorithms are an effective way of cleansing these types of datasets and identifying noisy sensors.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Technology has advanced to the point where almost anyone can obtain small, inexpensive sensors and computers. With this improvement in technology, we can now get small and inexpensive sensor nodes with more power to process and store data [2], [5].

1.1 Wireless Sensor Networks

A wireless sensor network is a set of autonomous sensor nodes that are spatially distributed to monitor physical or environmental conditions, such as pressure, temperature, vibrations, sound, and pollutants. The data from the sensor network is cooperatively passed to the main node, from which it is then passed to a server. A sensor network consists of anywhere from a few sensor nodes to thousands of sensor nodes. A sensor node is an autonomous device that has an embedded microcontroller, memory, I/O ports, A/D converters, a radio transceiver, a power source, actuators, and sensors. [5, 26, 2, 24, 30]. The sensors connected to each sensor node can be acoustic, imaging, infrared (IR), seismic, magnetic, temperature, pressure, vibration, or humidity sensors.

Each sensor node has at least communication, sensing, storing, and processing module. Communication modules or wireless modules are used to communicate the data with other sensor nodes or with the main node. The sensing module or sensor board digitalizes an analog signal of a transducer. The storing module temporally holds the sensor data, later that data can be transmitted or used to process data. The processing or programming board has multiple communications, sensing, and memory interfaces. [2].

The wireless communication uses public or private protocols [2], such as cellphone data

networks (CDMA, GSM, etc.), Wi-Fi networks (802.11 a/b/g/n, etc), ZibBee [31], and Wireless HART [22].

There are thousands of sensors to monitor almost anything with great precision. However, there are many challenges in processing and cleaning all the data from many sensors. Due to this, there is the need of producing new ways of cleaning and analyzing the data [5].

Some applications of wireless sensor networks are in:

- Military.
- Environmental applications.
  - Tracking movements of small animals and insects.
  - Monitoring environmental conditions that affect crops and livestock.
  - Forest fire detection.
  - Meteorological or geophysical research.
  - Mapping of the environment.
  - Flood detection.
  - Precision Agriculture.
- Health applications.
- Home applications.
- Other commercial applications.

Wireless sensor networks (WSN) allow for the monitoring of large areas that are remote, difficult to access, and/or dangerous to humans. Sensor networks are capable of covering large areas and producing measurements almost in real time [9, 28]. In this project, we focus only on light sensors, but the methods are general enough to be applied to other modalities.

1.2 Jornada Experimental Range

This study was conducted on the Jornada Basin Experimental Range (JER), which hosts the Jornada Long Term Ecological Research (LTER) Program managed by the United States Department of Agriculture (USDA) Agricultural Research Service (ARS). The JER is located 37 km north of Las Cruces, in southern New Mexico, USA in the northern Chihuahuan Desert, the largest desert in North America (see Figure 1.1). Detailed information about Jornada Basin Experimental Range can be found in [15].

The study site (latitude: 32.581956, longitude: -106.635025) is located west of the San Andres Mountains, at an elevation of 1,188 m, and a moderate slope of 2 degrees. Dominant plant species include the shrubs Creosotebush (*Larrea tridentata*) and Mesquite (*Prosopis grandulosa*). The study site includes a range of scientific infrastructure established and maintained by the Systems Ecology Laboratory at the University of Texas at El Paso (`http://www.sel.utep.edu`). UTEP-JER web site in `http://arctic.utep.edu/JornadaResearchFacility/`.

1.2.1 Power at UTEP-JER site

The system is powered by 600 W ten panel solar array. The solar panels are mounted on an aluminum structure located 35 m north of an eddy covariance (EC) tower. The panels face due south to maximize the battery charging. The system uses five 12 VDC @ 100 A sealed deep cycle battery systems.

1.2.2 Internet connectivity at UTEP-JER site

Data from the sensor network are retrieved remotely using an internet connection provided by the JER headquarters. The signal is sent and received by two bidirectional and omnidirectional antennas. Both antennas are attached to the EC tower at 25 and 9 meters height from the ground respectively. The system provides a WI-FI bubble, approximately 500 m in radius, to operate other wireless sensors (see Figure 1.2).

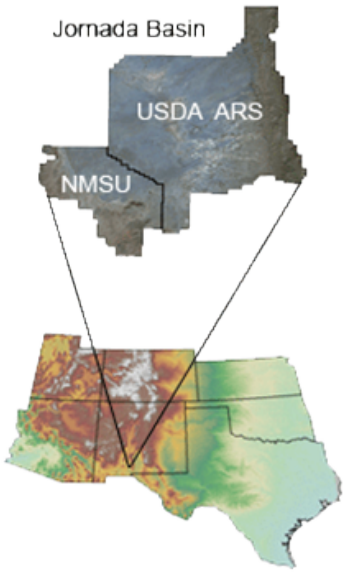

Figure 1.1: The USDA-ARS Jornada Experimental Range (JER) in southern New Mexico. A field station managed by New Mexico State University is located immediately adjacent to the JER. Map courtesy of Jornada Experimental Range `http://jornada-www.nmsu.edu/`.

1.3 Projects at UTEP-JER site

There are four projects at the UTEP-JER site, which will be described in the following paragraphs.

1.3.1 Extended Open Path Eddy Covariance System

A 10 m-tall tower hosting an open path eddy covariance system designed to measure land-atmosphere flux exchange was deployed in November 2009. The EC provides digital output of the fluctuations of carbon dioxide density, latent heat, sensible heat, momentum, temperature, humidity, horizontal wind speed and wind direction, net radiation, soil heat, soil temperature, and soil water content [3].

The eddy covariance tower has a total of 22 instruments: a three-dimensional sonic anemometer, an infrared gas analyzer, a four component net radiometer, a photosyntheti-

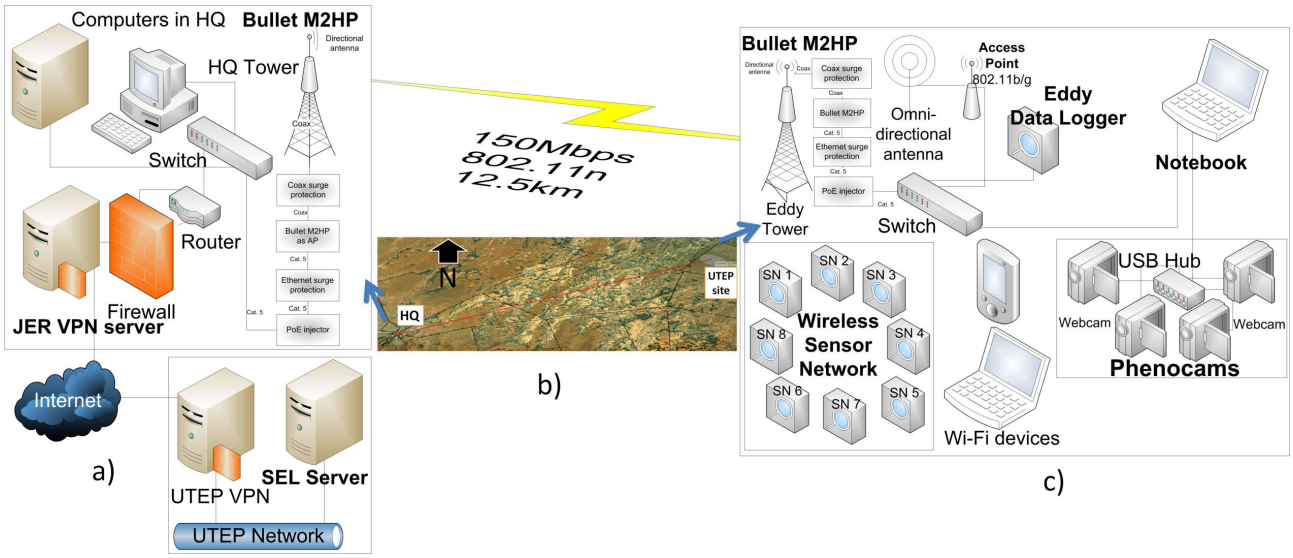

Figure 1.2: Internet link at UTEP-JER site. a) UTEP servers located at UTEP campus. b) Satellite image of the headquarters and the UTEP site showing Jornada HQ to UTEP-JER site connection. c) UTEP-JER site with different projects interconnected, 150 Mbps connection to HQ at 12.5 Km with a Local Wi-Fi reaching around 500 meters.

cally active radiation sensor, four soil heat flux plates, a temperature and humidity sensor, a barometric pressure sensor, a two-dimensional anemometer, eight soil temperature and volumetric water content probes, a rain gauge, and two leaf wetness sensors. The data are stored in a Campbell Scientific CR3000 datalogger (see Figure 1.3). More information can be found in [19].

1.3.2 Robotic Tram System

A robotic tram system equipped with a dual-detector spectrometer has been installed at the site to monitor this arid ecosystem's optical properties. These measurements can be used to calculate a range of vegetation indices that could be used for modeling estimated carbon fluxes (e.g. NDVI, PRI, EVI, SAVI).

This 110 m tram system, along with the sensor network, have been installed as components of a field-based infrastructure and phenology-monitoring program, which will be

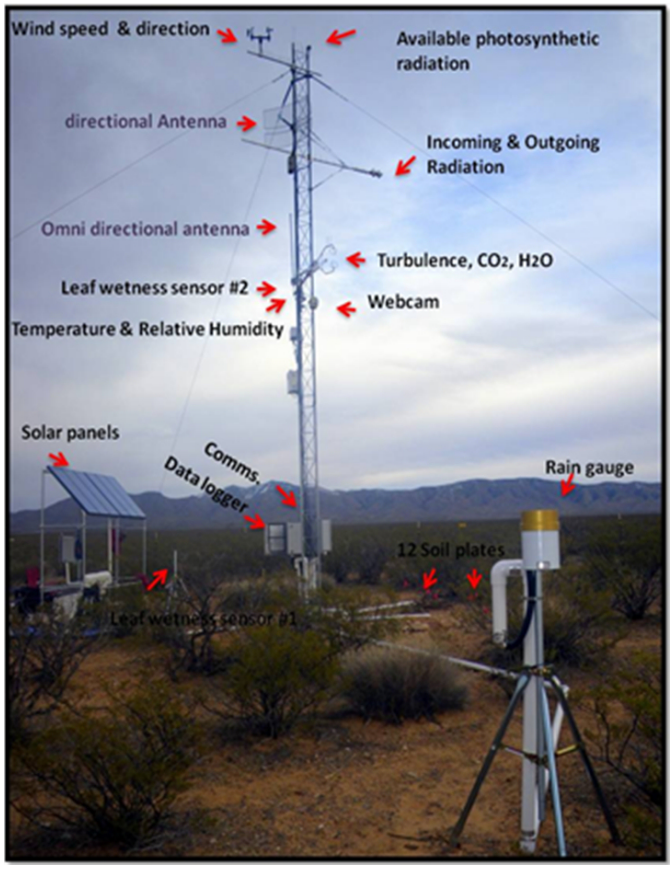

Figure 1.3: Eddy covariance tower at UTEP-JER site.

Figure 1.4: Robotic Tram System at UTEP-JER site

linked to national and international science networks (e.g. SpecNET, Ameriflux, Fluxnet, NPN).

Spectral properties and other properties recorded from this tramline will be related to land-atmosphere fluxes recorded on the eddy-covariance tower to develop scalable models useful for extrapolating processes at landscape and regional scales by using remotely-sensed data from satellites (see Figure 1.4). More information can be found in [11, 18].

1.3.3 Phenology

Phenology is the study of periodic plant cycle events and how these are influenced by seasonal variations in climate. Three transects were installed to monitor noticeable stages in the annual cycle of predominant species. The plants are being monitored at sites every 50 m along two 300 m transects and one 110 m transect within the towers footprint. This monitoring results in the detailed observation of selected plants, in each of their phenological states, from which percentages of greenness index can be calculated (see Figure

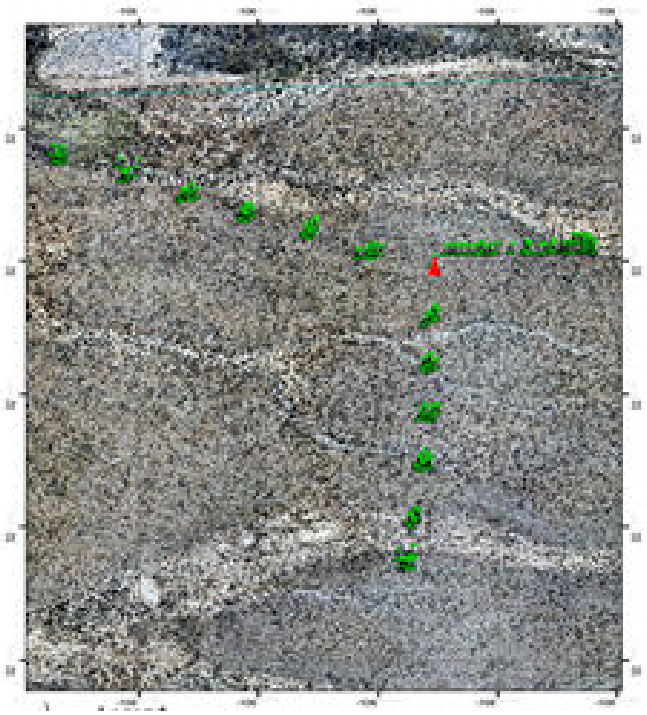

Figure 1.5: Three monitoring transects; Map created by Aline Jaimes

1.5). Documenting a plant's phenology response to global change by employing web-cams improves the understanding of phenological events. More information can be found in [12].

1.3.4 Wireless Sensor Network At Jornada Experimental Range

The sensor network at JER was comprised of eight sensor nodes (SN_01, SN_02...SN_08) that were placed alongside a 110-meter long tramline. The sensor nodes were placed at one-meter intervals from each sensor node. Each sensor node contained one or more sensors located in a tower. The main tower held the data logger and sensors and the secondary tower just held more sensors (see Figure 2.3). These sensors measured different attributes, such as humidity, and reflectance, of five different plant species. The species that were used for this research were: tarbush (FLCE), honey mesquite (PRGL), creosote (LATR), fluff grass (DAPU), and bush muhly grass (MUPO). Figure 1.6 shows the distribution of the sensor network.

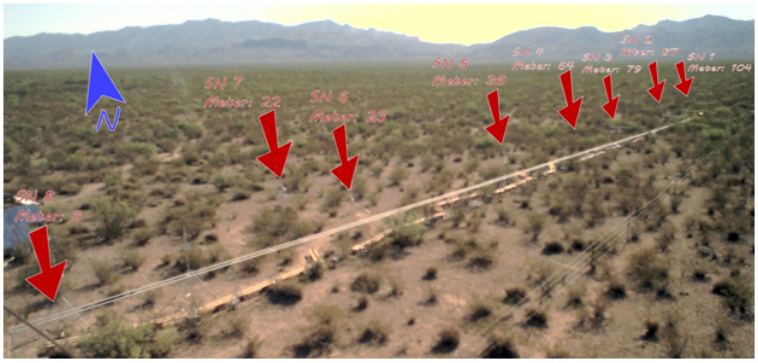

Figure 1.6: The sensor network at UTEP-JER site consists of eight sensor nodes measuring attributes of five perennial plant species and bare ground.

1.4 Problem: Data-Cleaning

In sensor networks, there is usual problem with the data that is handled or under consideration- the data is not always free of error. The frequent data errors may originate from corrupt communication, e.g. the sensor becoming disconnected from the data logger, the logger receiving a wrong voltage, the malfunction of the sensor hardware, interference due to the sensor being covered, and/or a faulty connection. The collected data are raw and must be cleaned before any scientist in the domain can manipulate the data.

The raw data must be manually inspected by a person who has experience with the kind of data to be checked. This person has to check every data point by looking for any error in the data. If there is any error in a data point, he/she must check other nodes sources that have data at the same time (as the time of the error) to look for any relation with the incongruent data point and try to guess the correct data point.

Data-cleaning using traditional methods is problematic and time-consuming because it requires manual review of the data. Thus, the design of methods to automate this process has become an active research area.

Some automatic methods detect and clean incorrect data from a sensor network as soon as the data are collected, but this usually implies some kind of limitations with the hardware in the sensor node. There are at least three limitations to the current hardware of sensor nodes: electric power, communication, and storage capacity.

The first limitation is the electric power needed to maintain a working sensor node. The electric power is needed to maintain the sensor node working and running basic processes in the embedded operating system.

The second limitation is the bandwidth associated with the network communications [5]. These communications are very important in the sensor node because the communications are used to transmit the data to the main node or to other sensor node. In methods proposed by [13, 29, 16], it is necessary to transmit the current data to other sensor nodes to check the correctness of other sensor data. But if the neighbor sensors are not placed close enough, if there is some obstruction in the transfer of the wireless signal by the network, or if the sent data is lost because the communication traffic is congested, the aforementioned correctness process cannot be achieved.

The third limitation is the node's storing capacity. Some processes to detect incorrect values use historical data from neighboring sensors and the sensor itself like [13, 29, 16]. All historical data must be stored in the local sensor node to check for a value's quality. These limitations and methods to detect incorrect data values are further discussed in [28, 2, 10].

While all the data must be stored, each sensor node cannot store a massive amount of data because it has a limited amount of memory. To solve this problem, all the data are stored away from the sensor network, in servers/computers which later clean and process the data. Then, all of the data must be available in devices of different capacities, from supercomputers to mobile devices, such as smart phones [5].

There are many methods that take advantage of the data when it is stored elsewhere from the sensor network. One such method is proposed by Deresynski and Dietterich [6]. That method uses long-term (multi-year) historical records of a single sensor's readings in order to derive a probabilistic model of its behavior over time. Afterwards, the quality

of future readings can be assessed by computing their likelihood given the model. Our approach is similar, but instead of probabilistic models, we use neural and instance-based learning algorithms and derive predicted sensor readings from observations made by multiple sensors. The learning algorithms exploit the redundancy provided by sensors that monitor neighboring and overlapping areas in order to learn about the interdependencies of the sensors' behaviors. This allows us to make accurate predictions without needing the long-term data used in [6].

1.5 Objectives

The objectives in this project are the following.

- Apply machine learning algorithms to predict a correct value in a wireless sensor network.
- Find the best strategy to replace suspected incorrect value.
- Find the noisy sensor assuming that there is one noisy sensor in the complete light sensor network.

In this project, we describe an automatic data-cleaning system that is based on machine learning. The system takes advantage of the redundancy of data provided by sensors that monitor neighboring areas at similar wavelengths to detect inconsistent sensor readings that may indicate malfunctions or excessive noise. We present experimental results which show the application of our system to clean data from a real-world environmental sensor network, the Jornada Experimental Range. We analyze several learning algorithms, data replacement schemes, and a method to detect a noisy sensor and conclude that learning algorithms are an effective way of cleaning this type of datasets.

the rest of this document is organized as follows.

We provide a detailed description of how we solved the problem and the dataset in Chapter 2. Then we discuss the details of the test and the results of this project in Chapter 3. In Chapter 4, we discuss different works that deal with the data-cleaning in sensor networks. We conclude this document and suggest some ideas about future work regarding this work in Chapter 5.

Chapter 2

Solution

The general solution consists of learning to predict the value of each individual sensor given the values of a set of related sensors; thus a network of n sensors requires n predictors. This is computationally expensive and thus makes our method better suited to post-processing (see Section 4.4), rather than on-site monitoring (see Section 4.2). We compute the likelihood of a given reading to be erroneous by comparing it with the predicted value. In order to predict a value we test three machine learning algorithms, which are discussed in Section 2.1. If it is found to be inaccurate, we replace its value using a replacement strategy that took into consideration the prediction made by the learning algorithm as well as the sensed value. The replacement strategy is discussed in Section 2.2. After testing different ways to predict a value, the next problem was to find a noisy sensor. The solution is discussed in Section 2.3.

2.1 Machine Learning

One of the main goals of this project is to predict a correct value of a light sensor based on the values of other light sensors at a specific time in a wireless sensor network. We decide to use Machine Learning Algorithms (MLA) because they are good real-values predictors and are not too complex to implement. We used the following three well-known learning algorithms to predict sensor readings: a feed-forward artificial neural network, k-nearest neighbors, and locally-weighted regression. For a detailed description of the algorithms see [25].

2.1.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) or Neural Networks (NNs) is a computational model that is inspired by the structure and functional aspects of biological neural networks. A neural network consists of an interconnected group of artificial neurons. An artificial neural network is an adaptive system that changes it's weights based on external or internal information that flows through the network during the learning phase. To train the network, the backpropagation algorithm is usually used. More information can be found in [25] Chapter 4.

2.1.2 K-Nearest Neighbor

The k-Nearest Neighbor (KNN) algorithm is a type of instance-based learning or lazy learning in which the target function is only locally approximated and all calculations are deferred until classification. This algorithm is the simplest of ML algorithms. An object is classified by taking a majority vote of its neighbors; as a result, the object is assigned to the class that os most common amongst its k nearest neighbors. The k-nearest neighbors' are calculated with the Euclidean distance. The k is a positive integer that is typically small. If $k = 1$, then the object is assigned to the class of its nearest neighbor. When approximating a real-value target function, instead to classified an object, the algorithm calculate the mean value of the k nearest training examples. More information about KNN can be found in [25] Chapter 8.2.

2.1.3 Locally Weighted Regression

Locally Weighted Regression (LWR) is a generalization of the KNN in which the target function $f(x)$ is approximated to a single query point $x \approx x_q$. The LWR algorithm constructs an explicit approximation to f over a local region surrounding x_q. LWR uses nearby or distance-weighted training examples to form this local approximation to f. More information can be found in [25] Chapter 8.3.

2.2 Replacement Strategies

The approaches taken to replace a suspected incorrect value are the following:

- Approach I, *Ignore*: do not replace or alter a noisy value.
- Approach II, *Replace-by-prediction*: replace a noisy value with a value predicted by the learning algorithm.
- Approach III, *Replace-by-average*: replace a noisy value with the average value that is calculated from the incorrect and predicted values.

2.3 Identifying One Noisy Sensor

The output of the learning algorithm deteriorates if there is a bad or noisy sensor in the input. If there is one noisy sensor in the input of the algorithm, the output or prediction will get low precision in comparison to the situation that occurs when all the inputs are clean.

After we tested different learning algorithms to predict a correct value with good precision, the next step was to find a sensor that was not working well. If we assumed that there is one noisy sensor in the data to be monitoring that cause low precision to the rest of the sensor prediction, then we need to find that sensor.

To find the noisy sensor, the idea was to predict the values of all the monitoring sensors and then calculate the errors of the predicted values with the current values of each sensor. Therefore, the hope was that the highest error represents the noisy sensor but not always the noisy sensor is the highest error. We determined an automatic threshold (AT) to find the noisy sensor. All errors greater than the AT are called noisy sensor candidates (NS_candidate). Each element of NS_candidate must be checked in order to find the real noisy sensor. To find the real noisy sensor was necessary to run again the learning algorithm as many time as the number of elements in NS_candidate. In each run, the algorithm

took one element of NS_candidate to be checked and removed the remainder elements in NS_candidate. The expected result is to get low value errors when in the input there is not the noisy sensor and one high error value when there is the real noisy sensor in the input.

2.4 Data Source - Dataset

This chapter describes the source of the dataset that was used in this project and a brief description of synthetic datasets to evaluate the different tests discussed in Chapter 3.

2.4.1 Data Source - WSN Set-up at the Jornada

The data we used originates from a wireless sensor network that was installed at a UTEP (University of Texas at El Paso) site on the Jornada Experimental Range, located near Las Cruces, New Mexico, to collect pilot data for a given sensor type (light sensors). The data collected was filtered and corrected to be used as dataset for test the Learning Algorithms to predict the sensor readings.

The light sensors consist of the following:

- Photosynthetically active radiation (PAR) sensors. The PAR Smart Sensor measures light intensity for photosynthesis. This sensor has a measurement range of 0 to 2500 $\mu mol/m^2/sec$ over wavelengths from 400 to 700 nm.
- Solar radiation (SR) sensors. The SR Smart Sensor (silicon pyranometer) measures in a range of 0 to 1280 W/m^2 over a spectral range of 300 to 1100 nm.

The light sensors are set up in the following manner. There is one PAR and one SR sensor facing upward to measure the incoming sun light and 15 PAR and 15 SR sensors facing downward to measure reflected sun light throughout the entire sensor network sample area (see Figure 2.1). The sensors were placed in pairs to monitor each dominant land cover type representative of the sampled ecosystem, which consists of plant species and bare ground. These dominant plant species are FLCE, PRGL, LATR, DAPU, and MUPO.

The Figure 2.1 shows one pair of light sensor facing upward and one pair of sensor facing downward.

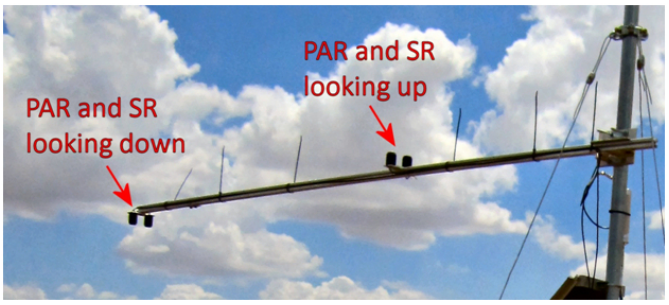

Figure 2.1: Light sensors looking up and down in sensor node SN_03.

The sensor network was installed by the Systems Ecology Lab (SEL) at the Jornada Experimental Range in Las Cruces, NM. The wireless sensor network was installed along a transect where six different species are being studied. The sensor types include:

- 1 pressure.
- 6 rainfall.
- 8 leaf wetness.

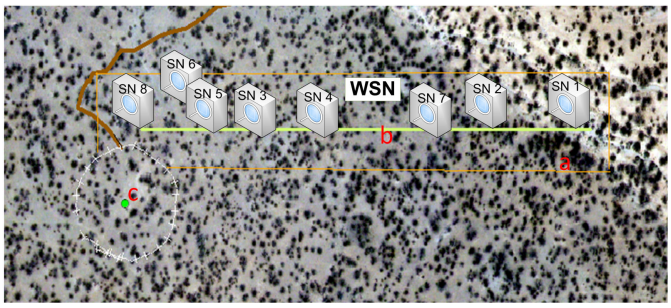

Figure 2.2: Aerial image of the UTEP site with WSN. *a)* is the fence, *b)* robotic tram cart, *c)* Eddy tower and phenocams.

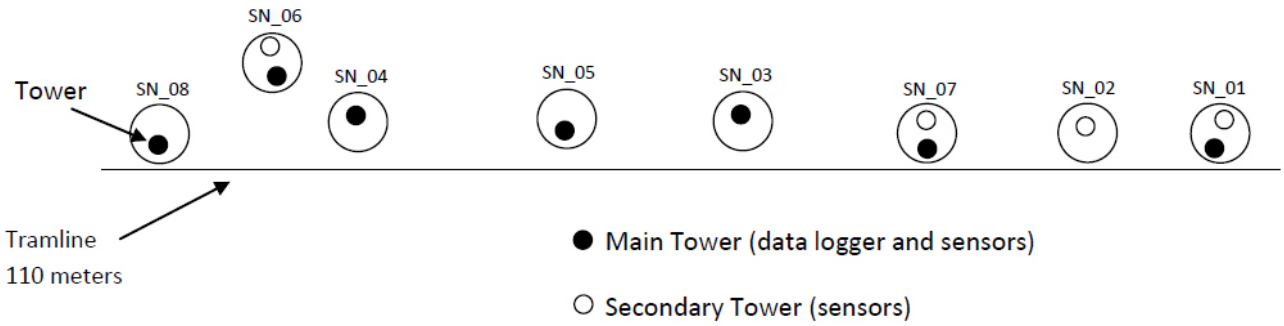

Figure 2.3: Distribution of the tower in the sensor network.

- 16 soil moisture.
- 16 Photosynthetically Active Radiation (PAR).
- 16 Solar Radiation (SR) sensors.

The sensor network is comprised of eight sensor nodes placed in a 110-meter long tramline. The Figure 2.3 shows the distribution of these nodes along the tramline transect.

Along the transect, there are 17 selected plants, 4 different soils, and 2 open areas. The open areas are used for control purposes. More information about the site can be found in Herrera et al. [17]. The main purpose of this facility is to measure the attributes of the studied species, compare data from the sensor network with data from a robotic tram system (see Section 1.3.2 and [11, 18]), and to analyze the data for monitoring carbon, energy, and water balance in the Chihuahuan desert. The entire sensor network consists of 87 sensors in 8 sensor nodes but for the purpose of this study, we were only interested in light sensors. 32 light sensors were installed in the wireless sensor network.

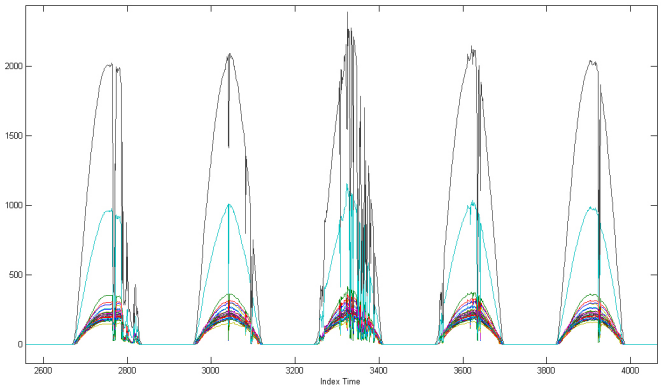

Figure 2.4: Five days of the dataset. There are 32 light sensors, each represented by a single line.

2.4.2 Dataset

The dataset was collected at the UTEP site in the Jornada Experimental Range; it is constructed from recordings of the mean values for five-minute periods using a thirty-second collection interval. It contains $10,656$ measurements over a 37-day period (see Figure 2.4). The original values were manually verified to ensure that the entire dataset was error-free.

To assess the effects of noise in a sensor, we simulated dust coverings as explained in section 2.4.2.

Simulated Dust

A new noisy dataset was constructed to simulate errors caused by the dust. The simulated errors are present only for a certain period of time. To simulate dust, numbers were randomly generated in the range from 0 to 50 to represent the percentage of obstruction due to accumulating dust. After the dust concentration reaches 50, wind can increase or decrease that percentage. However, dust can never reach 100 or 0. Figure 2.5 shows an example of the simulation of dust concentration over time. We generated 32 datasets simulating these errors, one for each light sensor.

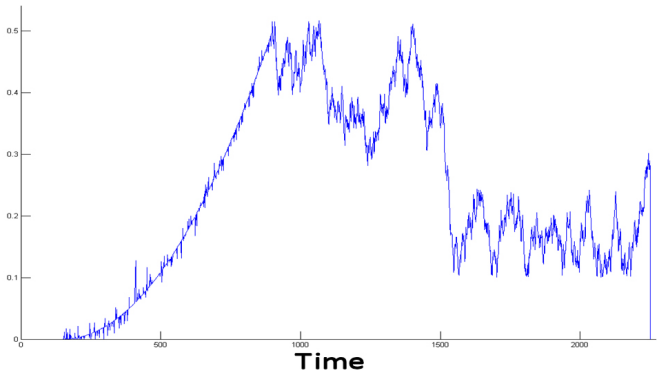

Figure 2.5: Simulation of dust on a sensor causing error in the data. The y axis is the percentage of dust.

One Noisy Sensor

Using the error-free dataset, we create a new dataset, called noisy dataset, which was inserted one noisy sensor. The noisy dataset tries to simulate a malfunction in a sensor, where the effectiveness of one of the 32 light sensors is reduced by 25%. 32 datasets where created in which each dataset only had one noisy sensor and the noisy sensor. The main goal of this dataset was to find the noisy sensor in the 32-light sensors.

Chapter 3

Testing and Results

3.1 Testing

Three sets of tests were conducted to evaluate the correctness of this work. The first test compared the performance of learning algorithms to determine which is the best one for predicting values. The second test evaluated the effectiveness of the four different replacement strategies in combination with the three learning algorithms. The third test searched for a noisy sensor using the LWR algorithm, which was used previously in the first and second tests.

3.1.1 General Conditions

The general conditions for conducting the tests are the following: with respect to the algorithms used (ANN, KNN, and LWR) in "Performance Of Three Learning Algorithms" (3.1.2) and "Replacement Strategies" (3.1.3), each test used the standardized dataset, in which each attribute was re-scaled to have both zero-mean and unit variance, 31 inputs, and only one output. The dataset was normalized to the range of -1 to 1 for the test presented in "Identifying One Noisy Sensor" (3.1.4). The artificial neural network had 26 units in the hidden layer, the KNN algorithm utilized $k = 5$, and the LWR algorithm utilized 200 data points to construct a local approximation. All tests were performed using a 10-fold cross-validation process with an error that was calculated as the square root of the Mean Square Error (MSE). To measure whether each value was correct, we calculated the distance between the error-free data value and the predicted value for each instance.

A relatively small distance value signified a good quality value and thus, quality decreased when the distance increased.

3.1.2 Performance Of Three Learning Algorithms

The first test used the data set that was collected from the network, without additional noise, to compare the performance of the three learning algorithms (KNN, LWR, and Feed-Forward ANN) in predicting the readings of a single sensor, given the readings of the other 31 sensors on the network as the input.

3.1.3 Replacement Strategies

The main purpose of the second test was to determine the best strategy for replacing an incorrect value. This was done by using a simulation of a dust scenario which could affect the entire data set. The approaches are the following:

Approach I: Ignore

In this approach, the predicted values are not used to replace the noisy value. Thus, the noisy value is not replaced or altered.

Approach II: Replace-by-Prediction

In this approach, we replaced a noisy value with a value predicted by the learning algorithm.

Approach III: Replace-by-Average

Approach III replaced a noisy value with the average taken from the noisy and predicted values.

3.1.4 Identifying One Noisy Sensor

This test searched for a noisy sensor by using the LWR algorithm. If there was one noisy sensor in the input of the algorithm, the output or prediction would produce low-precision results in comparison to the situation that occurs when all of the inputs are clean. After we tested the different learning algorithms to predict a correct value with a good level of precision, the next step was to find a sensor that was not working well. In this experiment, we assumed that there was only one noisy sensor in the data to be monitored, which produced low precision in the rest of the sensor predictions. Then, we found that noisy sensor using the learning algorithm.

To find the noisy sensor, the first step we took was to predict the values of all of the sensors and then calculate the error produced by both the predicted and current values of each sensor. The current value could be taken from the noisy sensor. As a result of the first step, we obtained 32 error values that represented each sensor, which were then stored in *errorValues*. Then the automatic threshold (AT) that was obtained from the addition of the mean and the standard deviation of the *errorValues*; $AT = mean(rrrorValues)+stdev(errorValues)$ was calculated. From *errorValues*, we obtained a list of noisy sensor candidates, *NS_candidates*. The noisy sensor candidates (*NS_candidates*) were the values that were greater than the AT.

Each noisy sensor candidate ($NS\_candidates_i$) must then be checked to find the real noisy sensor. To do this, it was necessary to run the implementation of the LWR algorithm again as many times as the number of elements in *NS_candidates* while checking one element of *NS_candidates* at a time and taking out the remainder of the elements in *NS_candidates*. Then, each run generated a list of errors with respect to one suspected noisy sensor that was then stored in *errorsInNSXX*, in which XX is the checking-sensor number. Consequently, there will be as many *errorsInNSXX* as the number of elements in *NS_candidates*. If we calculated the standard deviation of each *errorsInNSXX*, we could find another threshold, which was helpful in determining which was the real noisy sensor.

On each element of *errorsInNSXX*, we checked for those values that were greater than the new threshold and if we found one, it meant that the real noisy sensor was the actual sensor that we were checking (XX).

3.2 Results

3.2.1 Performance Of Three Learning Algorithms

Table 3.1 shows the tests with the error-free dataset. The mean error shown in Table 3.1 is the mean of the 32 outputs from each algorithm. The algorithm that has the best accuracy predicting new values is LWR, but all three provide satisfactory results.

Table 3.1: Mean errors with no noise inserted.

Algorithm	**KNN**	**LWR**	**ANN**
Max error	0.0648	0.0661	0.0650
Mean error	0.0434	**0.0228**	0.0284
Min error	0.0275	0.0123	0.0136
Figure	3.1	3.2	3.3

For each test in Table 3.2, there is a figure that shows three different plots. The line plot is the error-free dataset on SR on bare ground, the dot plot shows the results of the replacement approaches, and the circle is the distance between the original value and the replacement.

Table 3.2: Combination of different replacement approaches and learning algorithms.

				Approach
Replacement	**KNN**	**LWR**	**ANN**	**Figure**
Ignore	0.4966	0.4966	0.4966	-
Replace-by-prediction	0.0434	0.0228	0.0284	3.8
Replace-by-average	0.2506	0.2487	0.2489	3.9
Algorithm Figure	3.4	3.5	3.6	

3.2.2 Replacement Strategies

The mean error shown in Table 3.2 is the mean of the 32 outputs from each algorithm. For the second test, Table 3.2 shows summarized results of the combination of the three learning algorithms and the three replacement approaches. Approach I, no replacement, has the worst performance, which simply proves that it's worthwhile to replace data that are suspected to be erroneous. As expected, the best approach consists of replacing the noisy value with that provided by the best predicting algorithm, LWR in our experiments. Averaging predictions and measurements (approach III) also decrease the errors, but the improvements are much smaller than those obtained by replacement.

Figures 3.8, and 3.9 show an example of each approach using LWR prediction, and Figures 3.4, 3.5, and 3.6 show an example of each algorithm using approach II for replacement.

3.2.3 Identifying One Noisy Sensor

The Table 3.3 shows only 6 of 32 the results after running the algorithm to search for one noisy sensor. The complete table is located in Appendix A. In Table 3.3, the first column is the real noisy sensor that was inserted, the next two columns are error statistics that were calculated in first step, the column "AT" is the automatic threshold, the column "NS

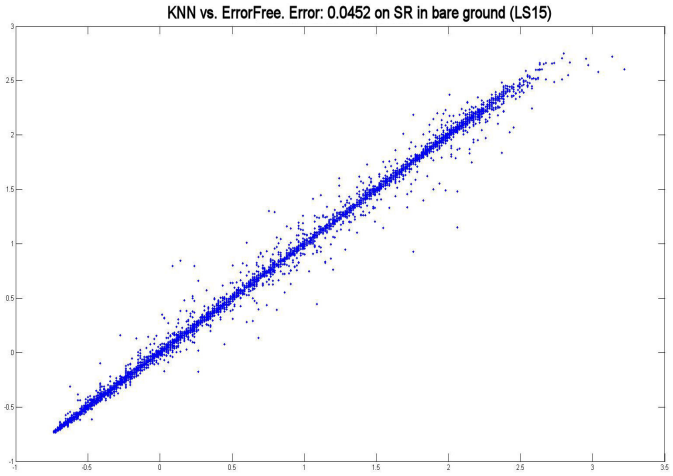

Figure 3.1: Output vs. prediction of a SR sensor over bare ground using KNN with $k = 5$.

candidate" has the noisy sensor candidate that was checked in second step, and the last two columns are the statistics of each sensor of the previous column. The bold number in column "NS candidate" is the real noisy sensor that the identifying one noisy sensor algorithm found. The last column (Std) in Table 3.3 shows the standard deviation of each sensor, in which we can see the difference in the real noisy sensor and the other clean sensors.

The Figure 3.10 shows a plot of the error in the first step of the algorithm that searches for the noisy sensor. The circles in the plot indicate the noisy sensor candidate. We can see in the Figure that the real noisy sensor, represented by the asterisk symbol, is the lowest value of the noisy sensor candidates but in the second step of the algorithm, the algorithm found the real noisy sensor. The Figure 3.11 shows the result of the second step, which shows the error of the noisy sensor. The Figures 3.12, 3.13, and 3.14 show the error plot of the noisy sensor candidates that actually are clean.

Figure 3.15 shows the result of the first step where there are three errors candidates. The Figure 3.16 shows the real noisy sensor but we can see that there are two error picks.

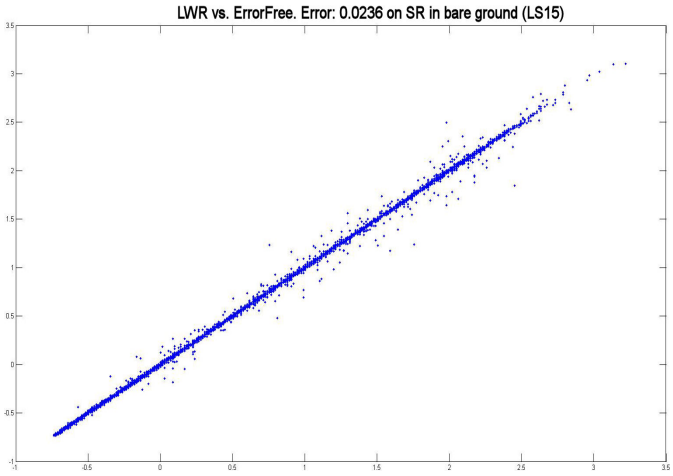

Figure 3.2: Output vs. prediction of a SR sensor over bare ground using LWR.

The second value is from sensor 25 that is not a noisy sensor candidate, so that sensor is discard as real noisy sensor.

The Figure 3.17 shows the plot of the error from first step where there is only one noisy sensor candidate that is the real noisy sensor. In this case the output of the second step is the same plot.

After running all of the experiments in identifying one noisy sensor, we found that the sensor 27 appeared as noisy sensor candidate on every experiment, and the sensor 25 appeared 23 times of the 32 experiments. Also, the pair of sensors (25 and 27) had the lowest number of noisy sensor candidates in the first step. See Figures 3.18 and 3.17. This represents a problem with the installation of the sensor. Those sensors were set as a pair upon a MUPO plant but those sensors are situated around $30cm$ from the top of the MUPO. See Figure 3.19. Also, the sensor measures are affected by a branch of a LATR. The abnormality can be caused by wind that moved the branch to the sensed area of the

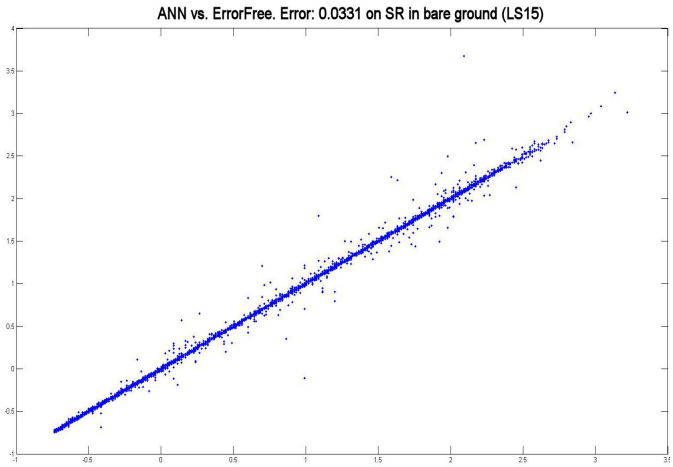

Figure 3.3: Output vs. prediction of a SR sensor over bare ground using ANN and 26 units in the hidden layer.

sensors, causing a variation in the measure. This abnormality does not mean that there is an error in the sensor or in the reading; it is a problem in the installation of the sensor. In conclusion, it is an abnormality that did not affect the algorithm final results.

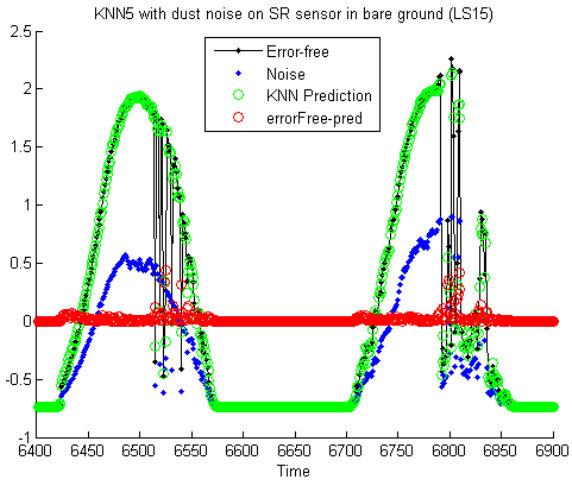

Figure 3.4: Plot of test using dust dataset in SR sensor over bare ground and using KNN with $k = 5$.

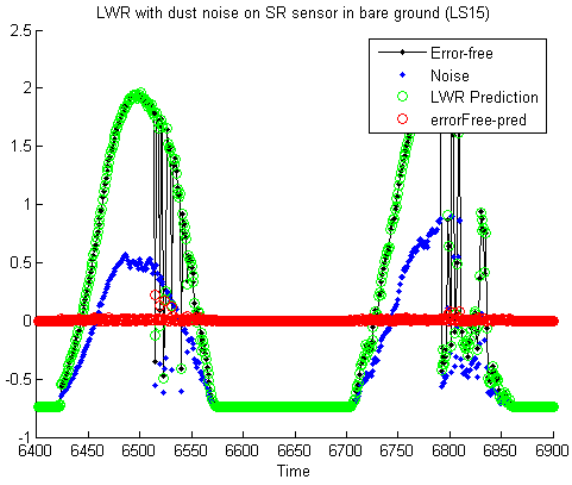

Figure 3.5: Plot of test using dust dataset in SR sensor over bare ground using LWR.

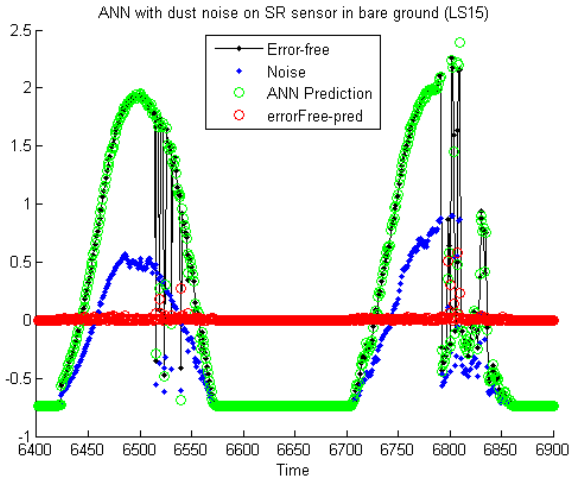

Figure 3.6: Plot of test using dust dataset in SR sensor over bare ground using ANN and 26 units in the hidden layer.

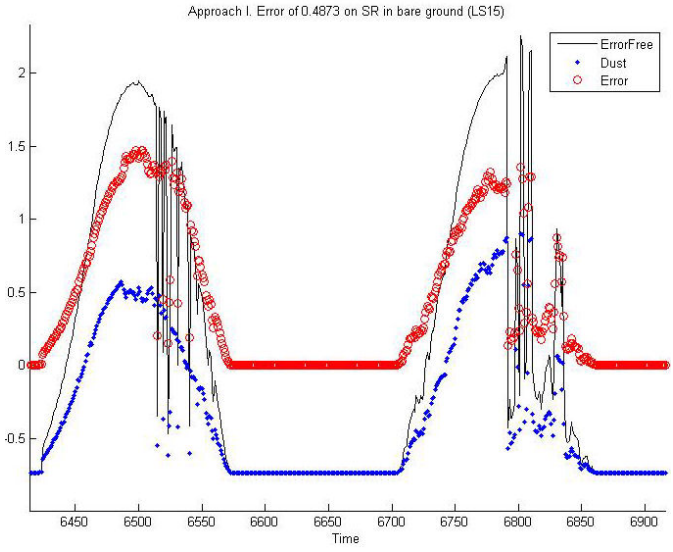

Figure 3.7: Approach I, there is no replaced value.

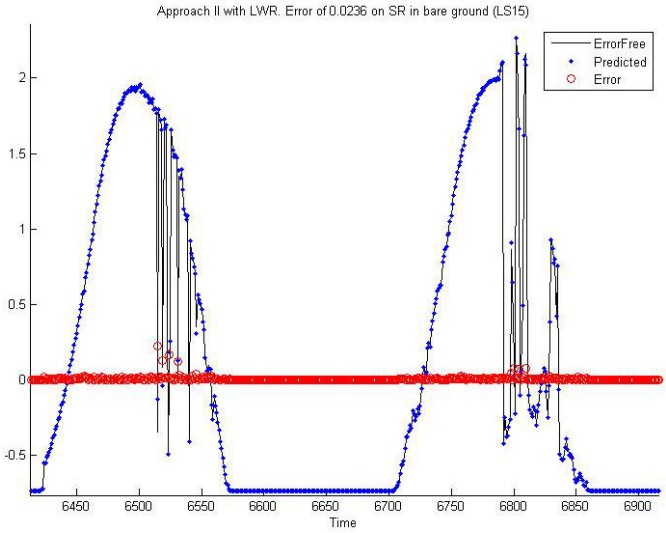

Figure 3.8: Approach II, the value replaced is the prediction of LWR.

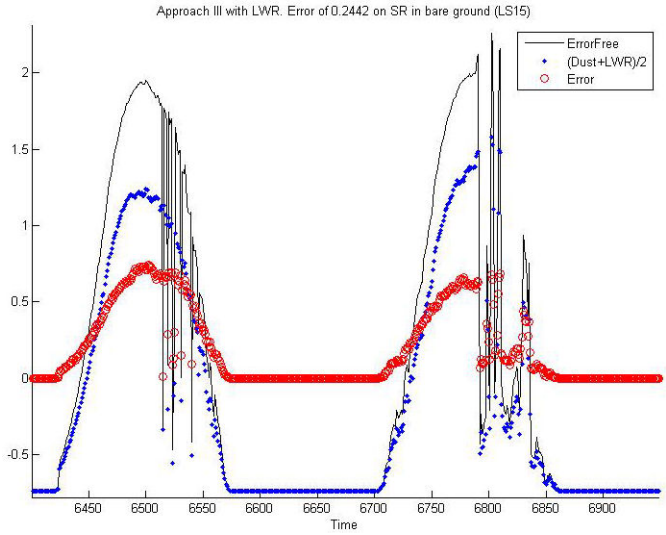

Figure 3.9: Approach III, the value replaced is the mean between the LWR prediction and the incorrect value.

Table 3.3: Identifying One Noisy Sensor

First Step				**Second Step**		
Noisy Sensor	**Mean**	**Std**	**AT**	**NS Candidate**	**Mean**	**Std**
6	0.0409	0.0332	0.0740	**6**	**0.0287**	**0.0320**
				27	0.0126	0.0061
				25	0.0121	0.0040
				4	0.0120	0.0037
				5	0.0120	0.0036
				1	0.0120	0.0034
15	0.0587	0.0462	0.1049	**15**	**0.0345**	**0.0248**
				27	0.0127	0.0064
				25	0.0118	0.0036
				16	0.0115	0.0031
24	0.0390	0.0360	0.0751	**24**	**0.0277**	**0.0313**
				27	0.0128	0.0060
				20	0.0119	0.0041
25	0.0212	0.0297	0.0508	**25**	**0.0196**	**0.0288**
				27	0.0123	0.0057
27	0.0190	0.0309	0.0499	**27**	**0.0191**	**0.0309**
32	0.0332	0.0341	0.0673	**32**	**0.0271**	**0.0319**
				27	0.0131	0.0070
				31	0.0125	0.0052
				30	0.0116	0.0037

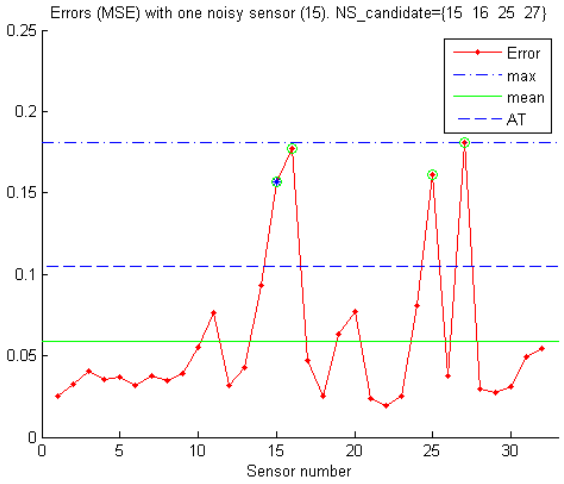

Figure 3.10: First step in algorithm identifying one noisy sensor. Noisy sensor 15.

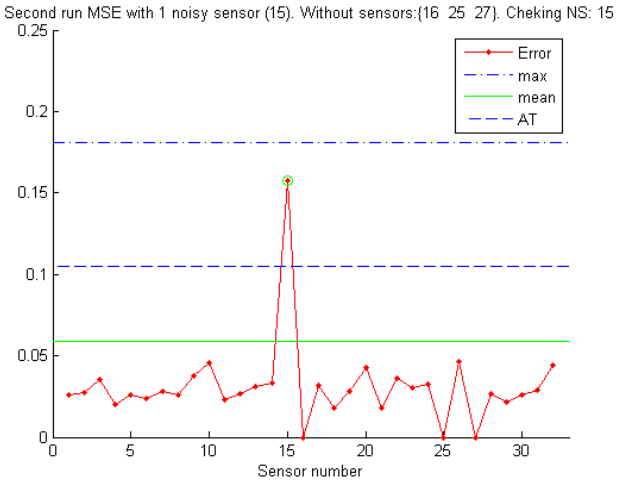

Figure 3.11: Second step in algorithm identifying one noisy sensor. Noisy sensor 15.

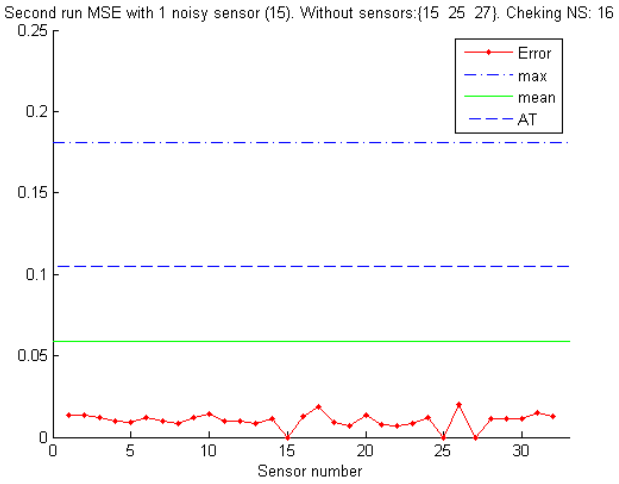

Figure 3.12: Second step in algorithm identifying one noisy sensor. Noisy sensor 15 checking noisy sensor candidate 16.

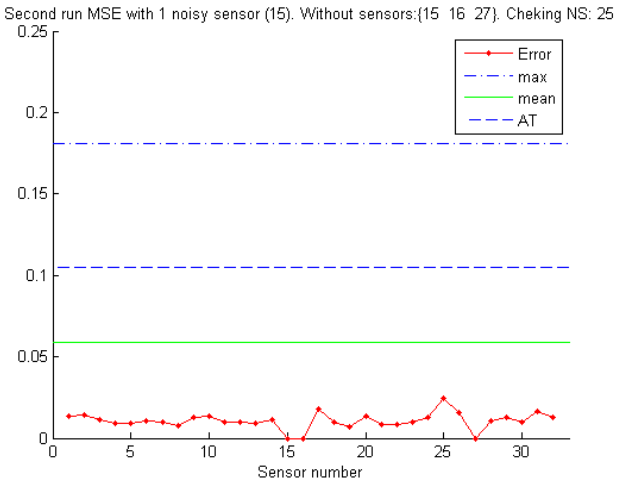

Figure 3.13: Second step in algorithm identifying one noisy sensor. Noisy sensor 15 checking noisy sensor candidate 25.

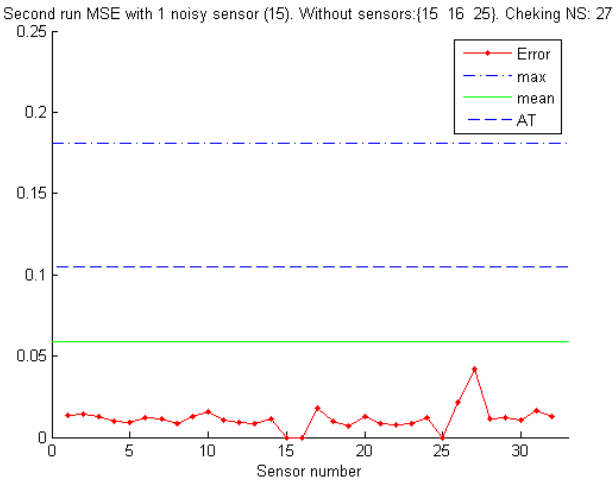

Figure 3.14: Second step in algorithm identifying one noisy sensor. Noisy sensor 15 checking noisy sensor candidate 27.

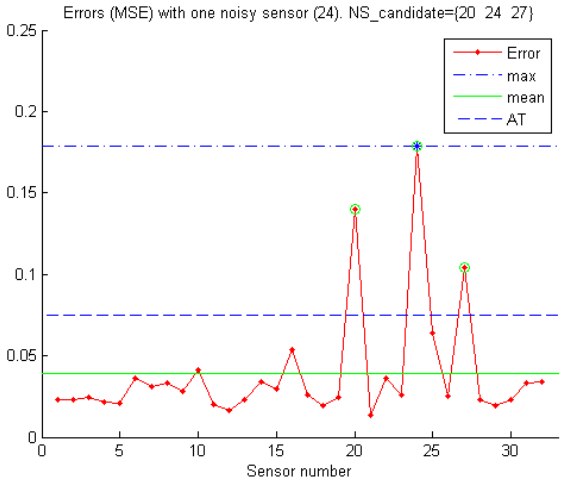

Figure 3.15: First step in algorithm identifying one noisy sensor. Noisy sensor 24.

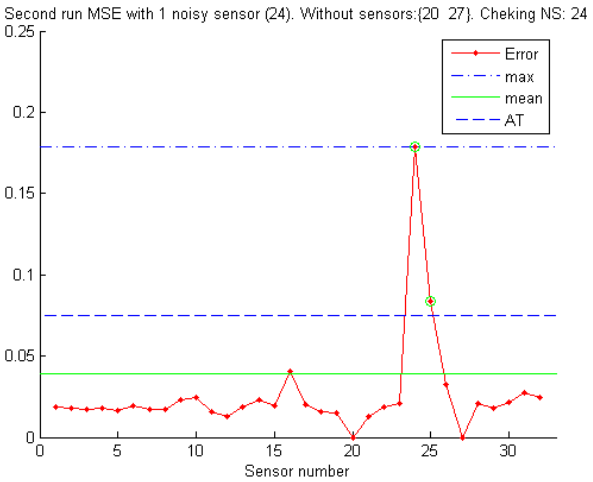

Figure 3.16: Second step in algorithm identifying one noisy sensor. Noisy sensor 24 checking noisy sensor candidate 24.

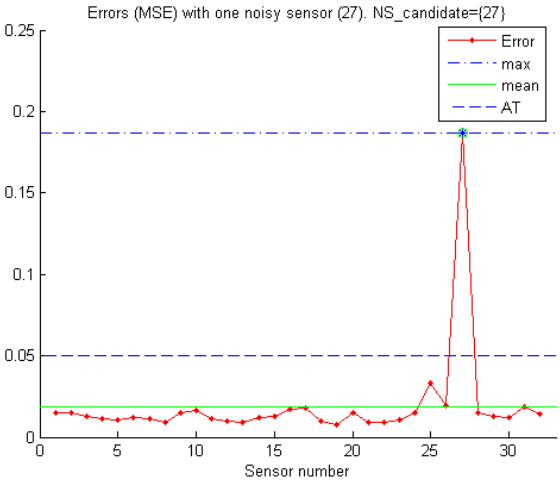

Figure 3.17: First step in algorithm identifying one noisy sensor. Noisy sensor 27.

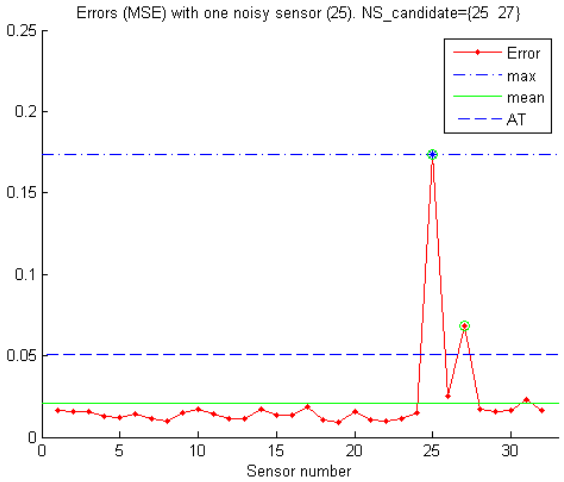

Figure 3.18: First step in algorithm identifying one noisy sensor. Noisy sensor 25.

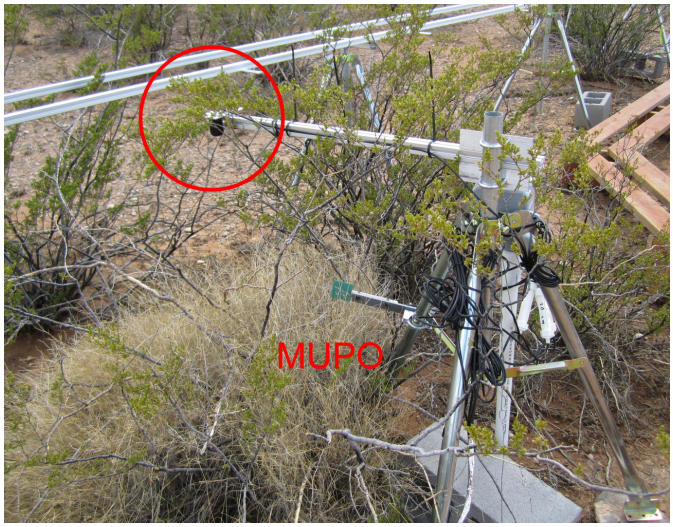

Figure 3.19: Abnormality in installation of sensors 25 and 27. In the circle, the pair of sensors and the branches that cause the abnormality.

Chapter 4

Related Work

The hardware limitations in a Wireless Sensor Network (WSN) make the development of node self-monitoring software a challenge. Sensor nodes usually lack the processing capacity that is necessary to run complex data quality assessment programs that detect when a sensed value is likely to be incorrect, which can occur due to external noise, broken sensors, or other reasons. Error detection is usually not an easy task and can be approached in various ways, which will be discussed in the following paragraphs.

4.1 Manually Cleaning

Manual data-cleaning consists of an expert checking every data point for any kind of error or missed data. This person can guess at a missing data values if he or she determines that it is necessary based on his or her expertise.

Because manually cleaning is tedious and time consuming it is desirable to have an automatic process to clean the sensor network data, as well as the basis of the problem that this work (which its discussed in the section titled 2) and other researchers have created a solution for.

4.2 On-Site Data Cleaning

Some approaches used in other researchers' work involve the detection of incorrect values, which occur as soon as the sensing process arise in each of the WSN's nodes. During this process, the sensor node uses its' processor, memory, and transceiver to analyze each data

value from any sensor connected to both the sensor node and some neighboring sensors to process a new data value. This is the reason why the sensor node uses more energy to process, store, and transmit information between other sensor nodes and the main sensor node. Some examples of on-site data processing are listed below.

Heinzelman et al. [16] present adaptive protocols, called Sensor Protocols, for Information via Negotiation, which efficiently disseminated information among sensors in an energy-constrained WSN. Those protocols use meta-data to eliminate the transmission of redundant data throughout the network.

Shuai et al. [29] present a work that uses sensor readings' spatial and temporal dependencies. To detect an outlier in sensor networks, they use Kalman filters because they achieve an optimal estimation of the state and they do not require so much computational power neither space to store the data. The method presents consists of two components, the state transition module and the measuring module. The first module uses auto-regression models to predict the next state. The second module measures the local environment by using the state of a neighbor sensor to verify that the state of the local sensor is correct.

4.3 Online Data Streams

A data stream is a sequence of digitally encoded signals used to transmit or receive information that is in the process of being transmitted [1]. Usually, this approach utilizes the temporal and/or spatial aspects of data [21], which must be either clean or must be cleaned. Some examples of sensor network data stream-cleaning are listed below.

Madden et al. [23] designed an acquisitional query processor for data collection in sensor networks. The authors focus on the locations and costs of acquiring data to significantly

reduce power consumption over traditional passive systems. They use SQL TinyDB to evaluate their acquisitional techniques which provide significant reductions in power consumption on TinyOS sensor devices.

Deshpande et al. [7] propose an architecture for integrating a database system with a correlation-aware probabilistic model that is based on time-varying multivariate Gaussian distributions, called BBQ. Deshpande's work solves the challenging optimization problem of selecting which are the best sensor readings to acquire from all of the readings by using the combination of BBQ and live-data acquisition to narrow them down.

The final work taken into consideration presents an extensible framework for cleaning the data streams. This framework is a declarative query-processing tool with a pipeline design. The authors introduce the concepts of temporal and spatial granules, which capture application-level notions of time and space. The framework utilizes those concepts in a pipeline of programmable-processing stages designed to clean receptor data as it streams through the system. For more details, refer to Jeffery et al. [20].

4.4 Post-Processing

Approaches in which sensor nodes are limited to sensing and transmitting data in which the data-cleaning occurs off-site as a post-processing stage are beginning to be more commonly used. This post-processing stage refers to a traditional process of cleaning data from a sensor network and is usually deployed from a database. This post-processing stage uses traditional data-mining methods to deal with missing, duplicate, and incomplete or corrupted data values in the databases [14, 4]. Below are other approaches that deal not only with the aforementioned problems, but also add some other advanced approaches that are similar to our work.

Jiang and Chen [21] present an integrated model for data-cleaning on signal-processing systems in a WSN to deal with missing data, duplicate data, incomplete or corrupted data, and outlier or noisy data. The authors in this work take advantage of the temporal and spatial aspects of the data.

Another work presented by Elnahrawy and Nath [8] discussed a framework for cleaning and querying noisy sensors using a Bayesian approach to reduce the uncertainty associated with the data caused by random noise. They use prior knowledge of the true sensor reading and the observed noise-reading to obtain a more accurate estimate of the reading.

The last work regarding processing that is referred to proposes a method that used long-term (multi-year) historical records of a single sensor's readings in order to derive a probabilistic model of its behavior over time. Afterwards, the quality of future readings can be assessed by computing their likelihood given the model. For more information, refer to Deresynski and Dietterich [6].

Chapter 5

Conclusions and Future Work

5.1 Conclusions

5.1.1 Performance Of Three Learning Algorithms

As shown in Table 3.2, the best algorithm to predict correct values was the locally-weighted regression algorithm, which uses 200 data points to construct a local approximation. The artificial neural network algorithm is the second best; its only drawback is that the training time is long. However, once they are trained, a prediction of a new value or dataset can be done quickly. The problem with the locally-weighted regression algorithm is that when a new instance is processed, it needs to process the whole training dataset to predict the correct value.

5.1.2 Replacement Strategies

The best technique to clean a bad sensor reading is to replace it with the value predicted by the locally-weighted regression algorithm.

5.1.3 Identifying One Noisy Sensor

The results in Table 3.3 show that in each experiment, the algorithm always found the noisy sensor. These noisy sensors were simulated with an obstruction of 25%. By using the

algorithm that searched for the noisy sensor, we were able to find those sensor measures that presented abnormalities due to the location conditions. Thus, the algorithm was able to discriminate the true abnormalities to find the noisy sensor. The results produced helped us by giving us an idea of how to find other abnormalities in sensor measures that we were not considering.

5.2 Future Work

Our next goal is to complete the development of a fully-automated system for data cleansing. We plan to take the following steps:

- Test the system with datasets with more than one failing sensor.
- Add new sensors modalities.
  - Solar panel voltage.
  - Rain gage.
  - Pressure.
  - Leaf wetness.
  - Other different sensors.
- Extended experiments with different size of datasets.
  - Large dataset- have at least one year of data.
  - Small dataset- chosen by selecting significative days of each moth of the year.
- Test alternative methods to assess the data quality, particularly, methods that allow multi-modal distributions to model expected sensor values.

References

[1] Federal Standard 1037C. Data streams, August 1996.

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

[3] Campbellsci, August 2009.

[4] Angela L. Cool. A review of methods for dealing with missing data. Annual Meeting of the Southwest Educational Research Association (Dallas, TX, January 27-29, 2000), January 2000. Paper presented at the Annual Meeting of the Southwest Educational Research Association (Dallas, TX, January 27-29, 2000).

[5] Erik Dahlman and et al. *Communications Engineering Desk Reference*. Academic Press, 2009.

[6] E. Dereszynski and T. Dietterich. Probabilistic models for anomaly detection in remote sensor data streams. *23rd Conference on Uncertainty in Artificial Intelligence (UAI-2007)*, 2007.

[7] Amol Deshpande, Carlos Guestrin, Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 588–599. VLDB Endowment, 2004.

[8] Eiman Elnahrawy and Badri Nath. Cleaning and querying noisy sensors. In *in Proceedings of ACM WSNA03*, pages 78–87, 2003.

[9] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor networks. In *MobiCom '99: Proceed-*

ings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pages 263–270, New York, NY, USA, 1999. ACM.

[10] Joo Gama and Mohamed Medhat Gaber, editors. *Learning from Data Streams*. Springer, 2007.

[11] John A. Gamon, Yufu Cheng, Helen Claudio, Loren MacKinney, and Daniel A. Sims. A mobile tram system for systematic sampling of ecosystem optical properties. *Remote Sensing of Environment*, 103(3):246 – 254, 2006. Spectral Network.

[12] Libia Gonzalez, Craig E. Tweedie, Jose Herrera, Aline Jaimes, Gesuri Ramirez, and Geovany Ramirez. The use of low-cost webcams for monitoring canopy development for a better understanding of key phenological events. The 95th ESA Annual Meeting (August 1 – 6, 2010), August 2010.

[13] J. Green, Bhattacharyya, and B. Panja. Real-time logic verification of a wireless sensor network. In *Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering (CSIE '09)*, volume 3, pages 269 –273, Los Angeles, CA, March 2009.

[14] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[15] K.M. Havstad, L.F. Huenneke, and W.H. Schlesinger. *Structure and function of a Chihuahuan Desert ecosystem: the Jornada Basin long-term ecological research site*. Long-Term Ecological Research Network series. Oxford University Press, 2006.

[16] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, pages 174–185, New York, NY, USA, 1999. ACM.

[17] Jose Herrera, Aline Jaimes, Jerald Brady, Gesuri Ramirez, Craig E. Tweedie, and Debra P.C. Peters. Utilizing novel technologies and cyberinfrastructure to understand global change impacts and feedbacks in an arid ecosystem. The 94th ESA Annual Meeting (August 2 – 7, 2009), August 2009.

[18] Jose Herrera, Aline Jaimes, Gesuri Ramirez, Libia Gonzalez, and Craig E. Tweedie. A robotic tram system used for understanding the controls of carbon, water and energy land and atmosphere exchange at the jornada basin experimental range. The 95th ESA Annual Meeting (August 1 – 6, 2010), August 2010.

[19] Aline Jaimes. The 95th ESA Annual Meeting (August 1 – 6, 2010), August 2010.

[20] Shawn R. Jeffery, Gustavo Alonso, Michael J. Franklin, Wei Hong, and Jennifer Widom. A pipelined framework for online cleaning of sensor data streams. Technical Report UCB/CSD-05-1413, EECS Department, University of California, Berkeley, Sep 2005.

[21] Nan Jiang and Zhiqiang Chen. Model-driven data cleaning for signal processing system in sensor networks. In *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, volume 1, pages V1–237 –V1–242, july 2010.

[22] T. Lennvall, S. Svensson, and F. Hekland. A comparison of wirelesshart and zigbee for industrial applications. In *Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on*, pages 85 –88, may 2008.

[23] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 491–502, New York, NY, USA, 2003. ACM.

[24] K. Martinez, J.K. Hart, and R. Ong. Environmental sensor networks. *Computer*, 37(8):50 – 56, aug. 2004.

[25] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[26] G.J. Pottie. Wireless sensor networks. In *Information Theory Workshop, 1998*, pages 139 –140, jun 1998.

[27] G. Ramirez, O. Fuentes, and C.E. Tweedie. Assessing data quality in a sensor network for environmental monitoring. In *Fuzzy Information Processing Society (NAFIPS), 2011 Annual Meeting of the North American*, pages 1 –6, march 2011.

[28] K. Romer and F. Mattern. The design space of wireless sensor networks. *Wireless Communications, IEEE*, 11(6):54 – 61, dec. 2004.

[29] Meng Shuai, Kunqing Xie, Guanhua Chen, Xiuli Ma, and Guojie Song. A Kalman filter based approach for outlier detection in sensor networks. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 4, pages 154 –157, December 2008.

[30] J.A. Stankovic. Wireless sensor networks. *Computer*, 41(10):92 –95, oct. 2008.

[31] A. Wheeler. Commercial applications of wireless sensor networks using zigbee. *Communications Magazine, IEEE*, 45(4):70 –77, april 2007.

Appendix A

Table Of Results Identifying One Noisy Sensor

Table A.1: Results of Experiments of Identifying One Noisy Sensor. The first column is the real noisy sensor that was inserted, the next two columns are error statistics that were calculated in first step, the column "AT" is the automatic threshold, the column "NS candidate" has the noisy sensor candidate that was checked in second step, and the last two columns are the statistics of each sensor of the previous column. The bold number in column "NS candidate" is the real noisy sensor that the identifying one noisy sensor algorithm found. The last column (Std) shows the standard deviation of each sensor, in which we can see the difference in the real noisy sensor and the other clean sensors.

Test in LS	**Mean**	**Std**	**AT**	**NS candidate**	**Mean**	**Std**
1	0.0526	0.0419	0.0945	**1**	**0.0361**	**0.0287**
				27	0.0123	0.0059
				25	0.0117	0.0037
				6	0.0116	0.0032
2	0.0497	0.0374	0.0871	**2**	**0.0398**	**0.0286**
				27	0.0126	0.0062
				25	0.0119	0.0036
				4	0.0115	0.0032
				9	0.0116	0.0031

(Continue)

Table A.1 – continuation of the previous page

Test in LS	Mean	Std	AT	NS Candidate	Mean	Std
3	0.0434	0.0336	0.0770	**3**	**0.0348**	**0.0292**
				27	0.0129	0.0060
				25	0.0124	0.0038
				5	0.0119	0.0032
4	0.0374	0.0322	0.0696	**4**	**0.0357**	**0.0312**
				27	0.0126	0.0058
				25	0.0121	0.0038
				6	0.0117	0.0031
5	0.0374	0.0331	0.0705	**5**	**0.0339**	**0.0323**
				27	0.0133	0.0063
				6	0.0123	0.0039
				3	0.0122	0.0038
6	0.0409	0.0332	0.0740	**6**	**0.0287**	**0.0320**
				27	0.0126	0.0061
				25	0.0121	0.0040
				4	0.0120	0.0037
				5	0.0120	0.0036
				1	0.0120	0.0034
7	0.0497	0.0405	0.0902	**7**	**0.0393**	**0.0275**
				27	0.0122	0.0061
				8	0.0122	0.0043
				10	0.0116	0.0038
8	0.0470	0.0355	0.0825	**8**	**0.0383**	**0.0275**
				27	0.0122	0.0063
				7	0.0124	0.0046

(Continue)

Table A.1 – continuation of the previous page

Test in LS	Mean	Std	AT	NS Candidate	Mean	Std
				9	0.0115	0.0037
9	0.0320	0.0320	0.0641	**9**	**0.0291**	**0.0291**
				27	0.0123	0.0059
				10	0.0118	0.0036
10	0.0319	0.0332	0.0651	**10**	**0.0287**	**0.0302**
				27	0.0124	0.0059
				9	0.0118	0.0034
11	0.0584	0.0413	0.0998	**11**	**0.0330**	**0.0278**
				27	0.0130	0.0063
				14	0.0125	0.0043
				25	0.0124	0.0043
12	0.0607	0.0397	0.1004	**12**	**0.0478**	**0.0276**
				27	0.0126	0.0063
				11	0.0125	0.0045
				25	0.0121	0.0044
				13	0.0119	0.0040
13	0.0512	0.0377	0.0889	**13**	**0.0434**	**0.0340**
				27	0.0129	0.0057
				25	0.0125	0.0040
				14	0.0120	0.0030
14	0.0576	0.0429	0.1004	**14**	**0.0329**	**0.0305**
				27	0.0131	0.0062
				25	0.0124	0.0042
				11	0.0120	0.0032
				15	**0.0345**	**0.0248**

(Continue)

Table A.1 – continuation of the previous page

Test in LS	**Mean**	**Std**	**AT**	**NS Candidate**	**Mean**	**Std**
15	0.0587	0.0462	0.1049	27	0.0127	0.0064
				25	0.0118	0.0036
				16	0.0115	0.0031
16	0.0466	0.0420	0.0886	**16**	**0.0412**	**0.0320**
				27	0.0125	0.0062
				25	0.0117	0.0037
17	0.0554	0.0417	0.0971	**17**	**0.0471**	**0.0244**
				27	0.0119	0.0057
				25	0.0113	0.0034
18	0.0407	0.0311	0.0718	**18**	**0.0387**	**0.0289**
				27	0.0124	0.0058
				25	0.0117	0.0037
19	0.0562	0.0363	0.0926	**19**	**0.0458**	**0.0257**
				27	0.0129	0.0057
				25	0.0122	0.0034
				22	0.0120	0.0029
20	0.0352	0.0314	0.0666	**20**	**0.0246**	**0.0271**
				27	0.0127	0.0058
				24	0.0119	0.0038
21	0.0536	0.0326	0.0861	**21**	**0.0496**	**0.0249**
				27	0.0125	0.0056
				25	0.0119	0.0036
22	0.0478	0.0360	0.0838	**22**	**0.0439**	**0.0321**
				27	0.0127	0.0057
				25	0.0120	0.0035

(Continue)

Table A.1 – continuation of the previous page

Test in LS	Mean	Std	AT	NS Candidate	Mean	Std
23	0.0476	0.0474	0.0951	**23**	**0.0408**	**0.0315**
				27	0.0125	0.0059
				25	0.0117	0.0035
24	0.0390	0.0360	0.0751	**24**	**0.0277**	**0.0313**
				27	0.0128	0.0060
				20	0.0119	0.0041
25	0.0212	0.0297	0.0508	**25**	**0.0196**	**0.0288**
				27	0.0123	0.0057
26	0.0472	0.0480	0.0952	**26**	**0.0385**	**0.0307**
				27	0.0120	0.0057
				25	0.0117	0.0041
27	0.0190	0.0309	0.0499	**27**	**0.0191**	**0.0309**
28	0.0394	0.0361	0.0755	**28**	**0.0343**	**0.0303**
				27	0.0124	0.0060
				25	0.0118	0.0035
29	0.0498	0.0433	0.0931	**29**	**0.0372**	**0.0275**
				27	0.0122	0.0058
				30	0.0122	0.0048
				25	0.0115	0.0038
				31	0.0116	0.0035
30	0.0394	0.0350	0.0744	**30**	**0.0309**	**0.0284**
				27	0.0132	0.0069
				25	0.0125	0.0052
				32	0.0123	0.0046
				31	**0.0299**	**0.0310**

(Continue)

Table A.1 – continuation of the previous page

Test in LS	**Mean**	**Std**	**AT**	**NS Candidate**	**Mean**	**Std**
31	0.0349	0.0340	0.0689	27	0.0125	0.0058
				32	0.0120	0.0039
32	0.0332	0.0341	0.0673	**32**	**0.0271**	**0.0319**
				27	0.0131	0.0070
				31	0.0125	0.0052
				30	0.0116	0.0037

Curriculum Vitae

Gesuri Ramírez García was born on January 18, 1980 in Puebla, México. The third son of Abisai Ramírez Valdivia and Gloria García Luna, and younger brother of Libna and Geovany Abisai. He graduated from Universidad Popular Autónoma del Estado de Puebla (UPAEP) as a computer engineer in fall 2006. In spring of 2008, he entered Graduate School of The University of Texas at El Paso (UTEP). While pursuing a master's degree in Computer Science, he worked as a teaching assistant for one semester in the Department of Computer Science. Then, he worked as a research assistant for Dr. Craig E. Tweedie in the Systems Ecology Lab (SEL) until December 2011. In SEL, he worked with the Arctic Research Mapping Application (ARMAP) from summer 2008 to fall 2008. In spring 2009 with collaboration of Sharing Resources to Advance Research and Education through Cyberinfrastructure (CyberShARE) Center of Excellence, he had the great opportunity to work with the "Desert Team" where he helped to establishment a cyberinfrastructure, located at Jornada Experimental Range. Mr. Ramírez develop the power source system, long-range communication system, robotic tram-cart for a robotic tram system, and the wireless sensor network. Moreover, Mr. Ramírez used his experience in engineering to help with the robotic tramline, phenocam system, and Eddy Covariance System installation. He worked at the University of California in San Diego (UCSD), the summer of 2011, under Dr. Tony Fountain' s supervision in the Division of the California Institute for Telecommunications and Information Technology (Calit2).

Permanent address: 2045 Coordillera del Aconcahua
Fracc. Maravillas
Puebla, Puebla, México. 72220.

E-mail: gramirez12@miners.utep.edu
gesuri@gmail.com