MONITORING HEALTH BY DETECTING DRIFTS AND OUTLIERS IN PATTERNS OF AN INHABITANT IN A SMART HOME

by

GAURAV JAIN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2006

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervising professor Dr. Diane Cook for providing me with continuous encouragement, inspiration and confidence. Her support and guidance played a big part in my thesis. Many thanks to the faculty at UTA who gave me knowledge and valuable advice that I hope to carry all my life. I would like to especially thank Dr. Manfred Huber, Dr. Sharma Chakravarthy, and Dr. Alp Aslandogan. I thank my old school teachers for helping me grow interest in computer science. I thank all my friends at UTA and back home. I am also grateful to the many friends and colleague from AI Lab who gave me good advice, especially Darin Brezeale, Michael Youngblood, Edwin Heierman, Abbasali Ginwala, and Nikhil Ketkar. Finally, I thank my parents Ranjana and Vinod Kumar Jain, my sister Ruchika Jain and cousins Gaurav, Aakash, Arnav and Elina for providing a lot of encouragement and support in my life and academic pursuits.

December 5, 2005

ABSTRACT

MONITORING HEALTH BY DETECTING DRIFTS AND OUTLIERS
IN PATTERNS OF AN INHABITANT
IN A SMART HOME

Publication No. _____

Gaurav Jain, MS.

The University of Texas at Arlington, 2005

Supervising Professor: Diane J. Cook

The elderly, along with people with disabilities or chronic illness, are most often dependent on some kind of formal or informal care. They are forced to move to a place where they can be cared for. Automatic health monitoring allows them to maintain their independence and continue living at home longer by continuously providing key health and activity information to caregivers. In this thesis, we present a novel technique, called the Health Monitoring System (HMS), which is a data-driven automated monitoring system for detecting changes in the patterns of activities/inactivity, health data and the living environment. HMS classifies these changes as drifts and outliers. These changes reflect short and long term lifestyle trends as well as any sudden changes

in the living patterns of the inhabitant. HMS uses domain knowledge to determine the importance of a change and reports them to the caregivers in an easy-to-understand format.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Overview

The burden on professional health-care providers has increased considerably as people are growing older. The elderly, along with people with disabilities or chronic illness, are most often dependent on some kind of formal or informal care. However, due to increasing healthcare cost, inadequate healthcare infrastructure and personal preference, they choose to stay at home. One solution is to provide a low-cost health monitoring system at home. Recent advancements in technology have made this possible. Research on intelligent environments offers an excellent platform on which a health monitoring system can be developed. Such a system can be used to monitor changes in lifestyle and health of such people and thus allow them to be independent longer.

This research examines the problem of finding changes in a human inhabitant's health and behavior. Finding changes, trends and patterns in somebody's behavior to understand their lifestyle and determine their wellbeing poses a challenging problem. Firstly, these patterns are difficult to obtain even with sophisticated monitoring systems and machine learning technologies. The reason for this is that the person's behavior and essentially their health could be dependent upon a large number of parameters. These may include his age, weight, body temperature, room temperature, activities, and social

interaction, to name a few. Secondly, as a person grows old he tends to change his habits and do things differently. As a result, the underlying patterns of activity and other measured characteristics changes and analysis becomes difficult. Thirdly, it is difficult to make assumptions on what could be classified as normal behavior and what is not normal or potentially harmful. Something which is normal for one person can be abnormal for another. Finally, people can have a tendency to change the way they do things without necessarily being affected by deterioration in their physical and mental abilities. These changes may indicate changes in their lifestyle but may not be correlated with their health. Adaptive algorithms are necessary to handle such analysis with the particularities of each individual in mind.

Changes are usually undesirable and unidirectional. However, we may also have changes which are bidirectional, cyclic, or they may be long or short in duration. Moreover, there could be sudden or unexpected changes which do match any previously-observed pattern. It is desired that algorithms for health care be able to differentiate between all these changes. Much of the prior work on providing technologies for automated at-home health care rely upon video or audio technologies for monitoring health. However, these technologies are expensive and are usually out of reach of many home dwellers. In contrast, we would like to develop an approach that does not require video or audio data specifically, but will identify changes in any available data that reflects the lifestyle of a smart environment inhabitant.

1.2 Scenario

Consider a scenario of an elderly man, who is a home care patient living alone after hospitalization for some disease. The patient's son, living in a different city, wants to keep a regular check on his father's health. The father does not want his son to call too often to check on his health. In addition, the son may not be able to correctly assess his father's health through telephone calls alone. Such a patient may be required to be regularly monitored for change in direct health parameters such as heart rate, blood pressure and body temperature. However, this data may not give complete information about his health condition and needs to be integrated with information on changes in other parameters from his environment (e.g., room temperature and humidity) and from his activity (e.g. his movement in the house, eating patterns, adherence to his daily routine). A monitoring system is required to collect all of this information and report important changes to the patient or somebody else in an easy-to-understand format. The son installs such a system in his father's home and within a short period of time the system learns the inhabitant behavior and starts reporting accurate and timely information about the changes in his health. A few weeks later the son notices in the report that his father has a sudden decrease in his movements around the house. He calls up and finds out that his father is not feeling well.

This kind of system may not only be beneficial to elderly people but also to vulnerable people such as patients, individuals with disabilities, or even healthy young people living alone.

1.3 Contributions

In this thesis, we introduce a new *Health Monitoring System* (*HMS*) for finding sudden, substantial or gradual changes in health, activities and the environment of an inhabitant. We hypothesize that such changes can be determined by analyzing sensor data collected in an intelligent environment, and that detecting such changes can be a useful component of a health monitoring system. This work is a part of the MavHome Project [2] at the University of Texas at Arlington. Some of the contributions of this research are listed below.

- HMS is an integrated approach for finding changes to health and lifestyle. It uses two algorithms for achieving its purpose. The first is a diurnal algorithm for finding changes in the form of drifts and outliers in the data. The second algorithm finds changes, in the form of outliers, in patterns of an ordered sequence of data (time series).
- Our approach predominantly uses data from simple, cost-effective sensors, common to home security systems.
- The detected drifts are categorized as *increasing*, *decreasing*, *cyclic*, *chaotic* and *no-drift*. The system is robust against noise and can respond to gradual changes in the data stream, and also at the same time respond to sudden changes.
- The outliers detected using the two algorithms, can be caused either by periodic changes in distribution of individual actions or by changes in time-ordered patterns collection of actions.

- HMS uses domain knowledge to weigh the significance of any change in the information being monitored. This knowledge is used to decide which changes to report. Reports are generated in such a fashion that even a non expert in data analysis can easily understand the results.

1.4 Thesis Layout

In this thesis we start by introducing prior work done in intelligent environments and health monitoring in chapters 2 and 3, respectively. Next, we give the design of our approach in chapter 4. In chapter 5, we provide experiments used to validate our approach. Finally, in chapter 6 we conclude.

CHAPTER 2

INTELLIGENT ENVIRONMENTS

Since the mid 1940s, the home automation industry has promised to revolutionize our living environments. The vision of the industry is that household devices - appliances, entertainment centers, temperature and lighting control units - will behave intelligently. Researchers share this vision and hope to provide a non-intrusive technology for assisting people in everyday life. Recent advances in technology have now allowed this vision to become a reality.

Our research uses intelligent environments as a tool for providing health monitoring to an inhabitant. In this chapter we present the definition for intelligent environments, its challenges and the work done by researchers on smart homes and other intelligent environments.

2.1 What is an Intelligent Environment?

Intelligent environments link computers to everyday settings and commonplace tasks. A definition of "smart" or "intelligent" is "the ability to acquire and apply knowledge", while "environment" refers to our surroundings. We therefore define a "smart environment" as one that is able to acquire and apply knowledge about an environment and its inhabitants in order to improve their experience in that environment [10].

Intelligent environments can be of many types such as smart homes, smart offices [3] [11] [12] [13], intelligent classrooms [6] [7] and intelligent hospitals [8]. Our research is conducted in the context of smart homes.

2.2 Challenges for Building an Intelligent Environment

Most of the advanced technologies such as personal computers available today, require a fair amount of intervention to setup and maintain. This resulted in the need for a *calm technology*, which could recede into the background of our lives. Weiser termed this technology as ubiquitous computing [2] [5]. He said that: "Machines that fit the human environment instead of forcing humans to enter theirs will make using a computer as refreshing as a walk in the woods."

For the past decade, researchers have worked toward the implicit goal of assisting everyday life and not overwhelming it. Researchers are investigating the intelligent environment frameworks that could recognize natural human behaviors, interpret and react to these behaviors, and adapt to its inhabitants in a non-intrusive manner. These features of an intelligent environment present difficult challenges to solve.

Another challenge is to seamlessly integrate different fields of study and research such as computer science, digital devices, and wireless and sensor networking to create an intelligent environment.

2.3 Smart Homes

Adding to the definition of smart environments, smart homes acquire knowledge about the home and its inhabitants by observing them over a period of time

and use this knowledge to predict the behavior and needs of the inhabitant in order to achieve the goal of improving their experience in the home. As with other smart environments, smart homes are expected not to be intrusive in the inhabitant's daily life and provide him a natural and easy to use interface to his environment. Moreover, smart homes should benefit different people according to their specific age, interests, and needs. In particular, elderly people may require special attention in order to increase their possibility of maintaining an independent lifestyle for aging in their own home [9].

A smart or intelligent home uses readily-available devices such as passive infra-red sensors, motion detectors, smoke, heat and gas detectors, door entry systems, and powered doors, connected together by a communication network. Enabling devices to communicate efficiently means that one device can then instruct other devices to perform functions if certain other conditions are met. The initiating device could be any device in the house. In this way, a collection of separate devices can be organized and programmed to carry out complex functions.

2.4 Research Projects

Early work done related to smart environments concentrated on specific applications, components and gadgets that could improve the experience of inhabitants in the environment. Some of the early applications were localization, people tracking and activity recognition. These formed the foundations of the intelligent environment.

In 1989, Cambridge University Computer Laboratory conceived the active badge location system [11] (or BATS). It provided a means of locating individuals within a building and was used for to forward phone calls and information to the

individual. Researchers at the University of Florida have instrumented a house with ultrasound localization and displays with the goal of providing timely and relevant information to residents [76]. People tracking is similar to localization and was approached using a variety of sensors, including cameras, laser range finders, wireless networks, RFID (Radio frequency identification) badges, and infrared or ultrasound badges [77] [78] [79] [80] [81] [82]. With these two technologies in place, the next step was simple activity recognition. Researchers at the Medical Automation Research Center (MARC) at the University of Virginia have used an array of motion detectors and contact switches to attempt to detect activities of daily living (ADLs) [21]. Other researchers have used cameras to detect a variety of activities, including sitting, standing and walking behaviors via wearable cameras [83], human gait recognition [84], and sign language recognition [85].

The Aware Home Research Initiative (AHRI) at the Georgia Institute of Technology developed an indoor location service and an activity recognition algorithm [86]. They utilized the location information to develop applications for social communication, memory aid and control of devices using gesture. They also developed an application which painted a digital family portrait which could communicate the general well-being of a person to family members.

A lot of work has also been done by industry and researchers on intelligent gadgets for a smart environments. MIT Media Lab's Consortia on Things That Think (TTT) and Counter Intelligence have worked on developing smart architectural surfaces (for example intelligent spoons), and smart kitchen gadgets (such as context-aware

tables, dishwashers, food oracles, SmartSink, and many more). Microsoft Research has combined the use of intelligent gadgets (such as smart cabinets) and other technologies such as inhabitant tracking, visual gesture recognition and others for their Easy Living project [87]. Another group at MIT, called the Agent-based Intelligent Reactive Environments group (AIRE) conducted research on pervasive computing and people-centric applications to construct intelligent spaces or zones. Their work included an intelligent conference room, intelligent workspaces, kiosks, and oxygenated offices. One of these is the Coen's Intelligent Room [88], which is a HCI platform that uses computer vision and speech recognition systems. One configuration of this room supports disaster relief planning. Inhabitants can examine weather conditions by pointing to a map and asking for weather information of a geographic location. Since a major goal of the Intelligent Room is to make the user interface invisible, the room uses cameras and microphones to interpret the activities of the inhabitants of the room.

The IPSI Research Institute in Germany is working on integrating home automation, consumer electronics, mobile communications, and personal computing for creating an intelligent network environment [89]. They are doing some leading research in creating gadgets and exploring issues related to human-computer interaction. An experimental iRoom [90] has been developed at Stanford University to explore integration issues with multiple devices, multiple user applications, and interaction technologies. It is conceived to be a representative of 'Weiserian' [2] ubiquitous computing spaces. They are studying issues with eye contact in videoconferencing, wall-display interaction, and information fusion issues. Some of the pioneering work in

ubiquitous computing was done by AT&T Laboratories Cambridge. Their work included simulating computer perception in computing systems that detect, interpret, and respond to facets of a user's environment [92]. The University of Illinois at Urbana-Champaign is involved in creating active spaces for ubiquitous computing, which can sense inhabitant actions and assist them with different tasks [91].

Researchers at the University of Florida have deployed a large array of smart devices and sensors to create a complete smart home, called the Gator Tech Smart House, with the goal of creating assistive environments to provide services to the inhabitant using telematics [98].

Several other projects use learning methods for activity recognition, energy conservation, prediction, and automation in an intelligent environment to improve the inhabitant's experience. Mozer's Adaptive Home [82] uses neural network and reinforcement learning to control heating, lighting, ventilation, and water temperature to the behavior and the behavior and the desires of the inhabitant. This is done to lower the cost of operation and increase comfort. Another project called the iDorm [88] uses a fuzzy logic based expert system to learn rules that replicate inhabitant interactions with devices. It is used to predict inhabitant needs and regulates energy consumption.

Finally, we have the MavHome (Managing an Adaptive Versatile Home) project [94], located at the University of Arlington at Texas, which views the home as an intelligent agent. Its goals include maximizing comfort of the inhabitants, minimizing the consumption of resources, and maintaining safety of the environment and its inhabitants. The home perceives the state of the environment using sensors and

acts upon the environment using device controllers in a way that can optimize its goals. The agent relies upon tools from artificial intelligence such as data mining, prediction and automated decision making.

In the MavHome project, prediction is used to determine the inhabitant's next action as well as to predict behavior of devices in the home, such as how much time is needed to warm the house to a specified temperature, and how much energy will be utilized in doing this. Specifically, MavHome identifies repetitive tasks performed by inhabitants that warrant potential automation by the home, and will need to be able to predict the inhabitant's next action in order to automate such selected repetitive tasks for the inhabitant. The home will need to make this prediction based solely on previously-seen inhabitant interactions with various devices and the current state of the inhabitant and the house. Prediction must then be handed over to a decision-making algorithm that selects actions for the house to meet its goals.

2.5 Summary

An intelligent environment can be defined as one that is able to acquire and apply knowledge about the environment and its inhabitants in order to improve their experience in that environment. Smart homes are intelligent environments used to reduce the burden on the inhabitant and provide health care and monitoring. The burden is reduced by automating the devices and controlling the environment. In the next chapter we will present related work on health monitoring and give an introduction to our work, which is called the Health Monitoring System (HMS).

CHAPTER 3

HEALTH MONITORING

Lanspery [20] notes that older adults and people with disabilities want to remain in their homes even when their conditions worsen and the home cannot sustain their safety. We all hope to maintain our independence and be a part of active society for as long as possible. However, when faced with old age, or chronic health conditions, or disabilities, we may be forced to move to a different environment where formal or informal health care is provided. This can be delayed or prevented if we can provide health monitoring at home.

In this chapter we introduce the topic of health monitoring, its different facets and the work done by researchers and industry related to health monitoring.

3.1 Introduction

Health monitoring can be used to monitor the general wellbeing of a person, emergency health situations and disease-specific symptoms. This can be done by using intelligent systems to monitor the inhabitant's activities, his health and his environment. Thus it brings together the fields of intelligent environments and health care. The need for health monitoring is not limited to the elderly and individuals with disabilities, but also can be useful to the healthy and young.

Some additional benefits of this technology include management of risk and reduced costs. Risk management supports people with dementia, people at risk of

falling or at risk of violence, and prevents hospital admission. This technology usually costs much less compared to the high costs of nursing homes. Remote monitoring reduces the need for physicians and other health care providers to perform home visits. Also, health monitoring at health care facilities will reduce the financial burden on the individual and the community.

An important form of health monitoring is telecare. Telecare offers remote care of elderly and vulnerable people. It is normally preventive, performing a monitoring role rather than providing primary care. A telecare system allows collection of health related information and sending it to a formal or informal health care provider located at a distance. Health monitoring forms a more general view of this system which allows even the inhabitant himself to monitor his health.

3.2 Different Facets of Home Health Monitoring

There are many facets of Home Health Monitoring. Most health monitoring systems combine two or more of these facets to comprehensively monitor and report changes in an inhabitant's health. Some of these facets are listed below.

3.2.1 Monitoring Inhabitant's Activities

A traditional assessment of everyday functioning provides a good starting point for looking at a person's health. According to Lawton[21], people living independently must be capable of performing basic Activities of Daily Living (ADLs) such as bathing, toileting, and eating and Instrumental Activities of Daily Living (IADLs) such as managing a medication regimen, maintaining the household, and preparing meals of adequate nutrition. Glacock and Kutzik [22] suggest eight human activities that can be

assessed to effectively monitor a person's health. These include eating, overnight toileting, stove use, medication adherence, wake-up time, general up and around, bedtime and overnight sleep disturbances, and bathtub use. It is important that the inhabitant should be able to adapt to a changing environment. Each of these classes of activities may generate patterns which could be detected. Status and trend analysis can thus be performed on these patterns to understand the inhabitant's behavior. Dewing et al.[49], Najafi et al. [54], Ogawa et al.[55], Noury [23] and Togawa [56] have researched techniques for learning and predicting inhabitant's activities. Research on specific activities such as hand washing [58], meal preparation [57], and movements around town [59] has also been pursued. Philipose, et al. [60] designed a PROACT system, which employs a Dynamic Bayesian Network (DBN) that represents daily activities such as making tea, washing, brushing teeth, etc.

3.2.2 Inactivity Detection

Patterns of inactivity can be used to make inferences about health and to help detect falls. A fall is a crucial problem of public health for the elderly people. More than one third of adults ages 65 and older fall each year. A fall can be detected using either a fall detector or using inactivity detection [23]. Several authors have used inhabitant tracking over extended periods of time to learn patterns of human activity in a scene from extracted motion data and subsequently to detect unusual activities in that scene. Nait-Charif and McKenna [24] use a visual tracking system to detect inactivity outside the usual zones of inactivity.

3.2.3 Health Data Monitoring

These include vital signs such as pulse rate, temperature, and respiratory rate and other physiological parameters such as weight, blood pulse pressure, and oxygen saturation. These measures provide a direct indication of inhabitant health. The collection of this data uses invasive or semi-invasive technologies. Devices which monitor these readings are either worn as a wrist watch or belt, or have electrodes which can be attached to an individual's body. Research has made these devices small and light weight so that they are minimally invasive to the person wearing them. The information collected can be then sent for analysis by a doctor or be processed through some software to detect anomalies. Monitoring of these parameters is called bio-monitoring. Budinger [25] provides the method and devices used for wireless bio-monitoring which can be used in a smart home environment.

3.2.4 Environment Safety

The physical environment in which a person lives plays an important part in shaping his health and safety. Factors such as room temperature, humidity and water temperature can affect the wellbeing of the inhabitant. The information collected here could be used in conjunction with health or activity data to study this effect.

3.2.5 Medication Adherence

Monitoring medical adherence is not just a requirement for elderly people but also for younger or middle-aged people. According to a study [26] conducted on a community-dwelling rheumatoid arthritis (RA) patients between the age 34 to 84, perfect adherence was more common in adults older than 55 years (47%) than in adults

aged 34 to 54 years (28%). Today, many commercially-available gadgets are available which cater to promoting and monitoring medication adherence in people. However, a determined person can still fool most of these systems. Researchers are trying to come up with an intelligent system using technologies such as video monitoring and others which can conclusively determine medicine intakes.

3.2.6 Social Interaction

Social interaction plays an important role in our daily lives. It is one of the most important indicators of physical or mental changes in aging patients. Social interaction is considered important for elderly people living alone or people suffering from dementia. A healthy communication between the elderly inhabitant and their extended families is desired for maintaining inhabitant's good health. According to a study conducted by Chen [27], a person sleeps well if he has a sufficient amount of social interaction during the day. For patients suffering from dementia, social interaction allows emotional release that helps to delay negative symptoms and promote positive aging. Technology has started to alleviate the social isolation among elderly. Generations on Line [47], a nonprofit organization, has developed simplified email and other technologies to promote communication for elderly people.

3.3 Health Monitoring in MavHome and HMS

MavHome, with its low cost monitoring technology and ability to process gathered information provides an excellent platform for many health monitoring systems. For HMS we use MavHome as our platform.

HMS is a data-driven automated Health Monitoring System. It combines different facets of health monitoring to provide an integrated approach for monitoring changes in the patterns of activities/inactivity, health data and environment. Changes in these data are classified as drifts and outliers. These changes indicate short and long term lifestyle trends and any sudden change in the state of the system. The system first learns the frequency patterns of individual actions, and then uses these patterns to detect lifestyle trends such as increasing, decreasing, constant, cyclic or chaotic frequencies. Any such change is considered to be a drifting pattern. These drifts are monitored and reported with higher priority given to drifts on health-critical actions. Outlier detection is performed to detect any anomalies in the patterns of the inhabitant, which may be caused by a sudden change in his health. This is done on the learned frequency patterns and data provided by the Active LeZi [14] method.

3.4 Other Health Monitoring Projects

Over the last ten years, numerous efforts have been made to create monitoring systems and to automate caregiver services for people living at home. In the previous sections we have listed some of the work done related to health monitoring and its supporting technologies. Here we present some more projects related to this field.

Much of the early work in this area used the store-and-forward method for monitoring a person's health. This involved acquiring data, images and/or video and transmitting this material to a health care provider at a convenient time for assessment offline. Togawa, et al. [28] [29] and Celler, et al. [30] were among some who used this method. Togawa was one of the first to use passive sensing of everyday activities to

monitor subjects. They monitored physiological parameters and also other parameters such as sleep hours, toileting habits, body weight and computer use. Celler collected data for measuring the behavior and functional health status of the elderly, and assessing changes in that status. Data analysis in both of these projects was done off-line. Celler generated reports for participants who have demonstrated a consistent change in functional health status. This work lacked in reporting the nature or the cause of the change.

In 2003, an experimental research project was done under the name Everyday Living Monitoring System (ELMS) [31] to study the usefulness of a low cost, non-invasive home health care system, which uses existing off-the-self technologies. This system collected data points representing specific human activities such as medical adherence, sleep disturbance, stove use, eating, and so forth. Once the data was collected and processed, status and trend analysis rules were used to detect particular human events (such as a fall) and detect changes in long term activity trends. All of this information was then compiled and provided to the caregiver using a website. The results indicated that the system was highly reliable and produced valid results.

By comparing HMS with these systems we conclude that HMS uses a more general approach by monitoring the environment without mapping observations to specific human activities. Status and trend analysis is done on the data to detect any anomalies and drifts in the inhabitant's behavior and health. Unlike other approaches, HMS allows the possible use of semi-invasive gadgets such as heart rate and blood pressure monitors.

The store-and-forward method was not useful in emergency situations such as sudden changes in the inhabitant's condition or his environment. This is called a live-monitoring method. It was used by Inada, et al. [32] to contact emergency personnel whenever there is a sudden change in the patient's condition. The system collects biological information, physical activity, and subjective information such as complaints. This system still required the patient to initiate the call. Chan, et al. [27] used machine learning to control the environment and automatically raise alarms. They used a neural network to learn the habits (locations) of the inhabitants [33] and raise alarms if they found any behavioral changes [34]. Sixsmith [35] described a similar system which raised alarms when current activity patterns are outside average patterns of activity.

HMS balances both of these methods, i.e., store-and-forward and live-monitoring, to give a more comprehensive analysis of the person's health. We will present the design for HMS in the next chapter.

The work at the Gator Tech Smart House [46] focuses on application of smart environments to elder care. They are planning to achieve similar health monitoring goals to the MavHome project. They are building an assistive environment to support independent living for older people and individuals with disabilities. Their ultimate goal is to create a "smart house in a box": off-the-shelf assistive technology for the home that the average user can buy, install, and monitor without the aid of engineers.

The Adaptable Home System (AHS) by Richardson and Poulson [36, 37] and the Georgia Tech Aware Home [19] provide gadgets for monitoring the inhabitant's health and assisting them if needed. Although AHS focuses on making devices more

supportive and easier to use, they also use medical monitoring devices that raise appropriate alarms. The Aware home provided applications and gadgets for social-communication, memory aids, and home assistants. They reported on the general well-being of the inhabitants rather than emergency situations. Memory aids are particularly useful for the elderly and patients suffering from various types of dementia [38] [39] [40] [41]. These aids usually exist in the form of reminders. Leikas, et al. [42] and Vigil [43] have provided security systems for monitoring the activities of demented people at home. Most of these systems focused on only a small subset of the problem.

Haigh, et al.'s Independent LifeStyle Assistant (I.L.S.A.) [44] is a well defined agent-oriented system for health monitoring and reporting. They are able to integrate the information, assess the situation and communicate it in an appropriate fashion. They use separate agents for sensing, situation assessment, response planning, and response execution and actuation. The aim for this project is to evaluate the feasibility of integrating multiple disparate AI technologies into one cohesive system.

A recent work [45] by Martin, et al. uses fuzzy association analysis in a telecare system to monitor long term trends, significant patterns and association among patterns. It addresses the issue of the imperfect data source of a sensor network. Their focus is on reporting the general well-being rather than emergency situations.

Finally, work on many specialized technologies has been done towards supporting health monitoring in intelligent environments. These include computer vision techniques for inhabitant tracking [52] [53], and fall detection and other crisis detection techniques [23] [48] [50] [51].

We conclude that HMS is a balanced approach and is more general compared to the previous work done in health monitoring.

CHAPTER 4

HEALTH MONITORING SYSTEM (HMS)

In this chapter, we introduce our Health Monitoring System (HMS) algorithm that uses statistical and prediction-based techniques to detect and report changes in an inhabitant's lifestyle and occurrences of possible emergency situations. The lifestyle changes are referred to as pattern *drifts* and the emergency situations as *outliers*.

4.1 Health Monitoring System (HMS)

Changes are seen in our life all so very often. These changes could be seen in the environment in which we are living, or could be in our health, or our social activities, or our habits and behaviors. These changes could be due to old age, bad health, unexpected incidents, or could be just done intentionally to deviate from our monotonous daily routines. They may affect our long term and immediate health. Knowledge of all the significant and gradual changes can give an individual's caregiver insight on the health and general well well-being of a person. This will help him assist the individual in a better way. Moreover, personal knowledge of these changes for a smart environment inhabitant can allow him improve his lifestyle and avoid potentially dangerous situations.

The Health Monitoring System (or HMS) is a data-driven automated algorithm designed to detect these changes. HMS classified these changes as drifts and outliers. Drifts are further classified as *cyclic*, *increasing*, *decreasing*, *chaotic* and *no-drift*. Drift

detection may help us find general changes in lifestyle and outliers may help us in detect emergency situations. This system is an integrated approach to find changes based on both small and long trends and to report the most significant changes. This system is designed to be robust to noise and at the same time detect any sudden unexpected changes.

HMS is a highly flexible system, which can monitor many kinds of data. This data may consist of individual actions such as turning the television on or opening the door, a collection of actions such as taking a bath followed by preparing a meal, health data collected at regular intervals such as pulse rate and body temperature, or parameters such as temperature and humidity which define our environment state.

HMS is an integrated system for monitoring both long term trends and sudden changes in activity patterns, as well as the environment and the health data of an inhabitant. Knowledge of these trends and changes helps us to understand the lifestyle of the inhabitant. This information can be use to ensure the health and well being of an individual. Moreover, information on sudden and significant changes can help them take evasive actions. Throughout this thesis we will be using the terms *drift*, *trend*, and *outlier* to represent types of changes that may be automatically detected in collected data.

HMS is designed to work in a dynamic environment where data is being continuously received and the underlying model may change over time. HMS is designed to incorporate all detected changes into the learned model, allowing it to quickly adapt to the drifting concepts.

For the purpose of monitoring the health of a smart environment inhabitant, some types of change are more important than others. For example, a big variation in blood pressure readings, a decrease in general in home activity levels, and a sudden increase in use of potentially dangerous gadgets such as stove or water heater, could indicate changes in health and thus should be reported with high priority. Other changes, such as seasonal changes in habits, increased use of a home computer, and increased movement within the house, may not be significant enough and thus are reported with a lesser priority. Moreover, the priority of a reportable change can be affected by the size or type of the change. Big changes may affect an inhabitant's lifestyle and thus should be reported with a greater priority. For this purpose we use our domain knowledge to associate a criticality level with each data type. This criticality level, along with the size and type of the change, indicates the importance of a change in that data. These values are used to calculate the urgency of reporting any change or long term trend in that data. In this thesis we use the term *action* to refer to each data type on which change is being monitored. This is because we are initially monitoring smart environment inhabitant actions for change detection.

HMS consists of two algorithms. The first algorithm is used to find drifts and outliers in action data collected daily. The second component is an online algorithm that is used to find outliers and report them as soon they occur.

4.2 Terminology

The following sections present the terminology that will be used when discussing the components of the HMS algorithm.

1. ACTION: This term is used to represent the change in state or measurement of an entity in the environment caused by the inhabitant. This could be a change in the state of a device or object (e.g. stove_on, light_off, door_close etc.) or it could be measurements taken at regular interval (e.g. heart rate and blood pressure). We can categorize numeric measurements such as heart rate into discrete actions such as low_heart_rate, regular_heart_rate, and high_heart_rate. In addition, specific activities such as eating and cooking can be considered as actions. An action is represented by the tuple:

<*action_name action_criticality*>,

where *action_criticality* is an integer between 1 to 5 (inclusive), which governs how important is it to report a drift or an outlier based on this action. Here five represents the most critical actions and one represents the least critical actions.

2. EVENT: An event is represented as <*event_time action_name*>, where event_time is the time when action_name was observed.

3. FREQUENCY SET: Each action is associated with a frequency. This is the number of times this action was observed in a specified interval of time. A frequency set is a collection of frequencies of all the actions in the environment in a specified interval of time. Note that the frequency of a particular action could be zero. A frequency set is represented as

<*start_time duration* x_1 x_2 x_3 ... x_n>

where *start_time* is the time at the start of the interval and *duration* is its length. The sequence ($x_1...x_n$) represents the frequencies of each possible action ($a_1..a_n$) during the given interval.

4. SAMPLE SIZE: Each collected frequency set is called a sample and the duration or interval it represents is called the sample size. Examples of sample size could be 6 hours, daily, and weekly, to name a few. In our algorithm we use frequency sets with many different sample sizes.

5. EXECUTION INTERVAL: This is the regular interval at which our algorithm is executed to detect drifts and outliers in the data.

6. HISTORY: A history represents all previous and current frequency sets. It maintains separate queues of frequency sets, for each sample size.

7. WINDOW: A window represents a subset of one of the queues in the history, containing the most recent frequency sets; see figure 4.1. The size of this window is less than or equal to the size of the corresponding queue in history.

4.3 Drift Detection Algorithm

This algorithm has an execution interval of one day. This means that at the end of every day the drift detection algorithm is executed to find the drifts in our data. The history is maintained through different executions. The input to this algorithm is the history collected until yesterday and today's frequency sets. The first step is to update the history with this data. Next, we run our algorithm and classify each action with the most prominent drift calculated for the action. Finally, a report is generated listing all the significant drifts and outliers in the data.

We classify each action as one of five different kinds of drifts. These are *no-drift*, *increasing*, *decreasing*, *cyclic*, and *chaotic*. Each of these drifts could be mapped onto a certain activity or action in our daily lives. For example, frequencies for brushing your teeth and taking a bath usually remain constant and thus can be classified as *no-drift*. Other activities such as cleaning the house and leaving for school could be *cyclic*. Moreover we can have actions which have steady an increase or decrease in frequency over some period of time. In the rest the frequency distribution may be too skewed to determine any change. This drift can be caused when the type or nature of the drift is going through a transition from one to another. These will be classified as *chaotic*.

The components of the algorithm are explained in the following sections. Reporting of drifts will be discussed in section 5 of this chapter.

4.3.1 Updating the History

In our algorithm we maintain three history queues. These are the six-hour queue, the daily queue, and the weekly queue. The algorithm execution interval is taken as daily. Thus at the end of each day HMS adds four six-hour frequency sets, one daily frequency set and one weekly frequency set to the history. The weekly frequency set is constructed using the last seven daily frequency sets, including the current one. The history is allowed to grow up to a maximum size. The oldest frequency sets are deleted when the size exceeds the maximum size threshold.

For each sample size in our history we use different window sizes. The size of the window used could determine what kind of trend we are looking for. A large window provides the learner with much training data, allowing it to generalize, given

that the concept did not change. A large window is more tolerant to random noise than a smaller one, thus they are used for detecting long term trends. On the other hand, a large window can contain old data that is no longer relevant (or even confusing) for the current target concept. In these cases smaller windows are useful. They can detect any sudden changes of concept.

For each sample size, we use three fixed-size windows. The size of these windows is chosen in such a way that it is approximately four times the length of one cycle in a *cyclic* trend and is twice the length of other trends, which we would like to find. This is required since we need at least four complete cycles to confirm a presence of a continuing *cyclic* behavior. For the daily sample size, we consider three window sizes:

- a two-week window for detecting cycles of two to four days in length,
- a five-week window for detecting cycles of five to nine days in length, and

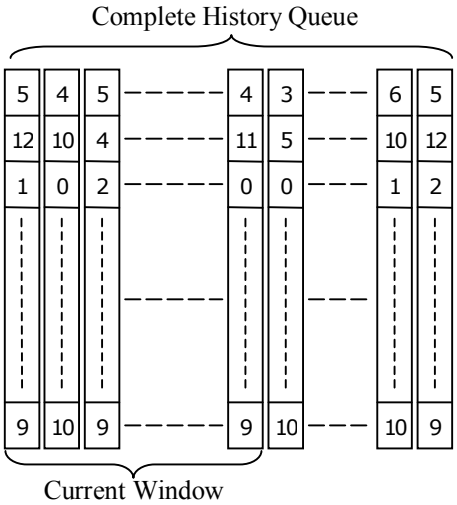


Figure 4.1: One of the history queues with the current active window.

- a five-month window for detecting cycles of nine to forty days in length.

In figure 4.1 we see a visual representation of a queue of frequency sets in the collected action history with pointers to the current active window. At any moment of the execution of the algorithm, the history keeps the pointers to the current active queue and the current active window inside it. The active window is the window in which the algorithm searches for drifts and outliers.

4.3.2 The Algorithm

Here we describe the functions that comprise the HMS drift detection algorithm. We start by first giving the main algorithm loop.

4.3.2.1 Main Algorithm Loop

Given the history *h*, the new frequency sets, the action list and the criticalities of each action, the main loop of the HMS algorithm for drift detection can be summarized using the pseudo-code in figure 4.2. This algorithm first adds the new frequency sets to the history *h*. Next, it finds and reports drifts for each action in the history. Finally, the

Input: history *h*, frequency sets, action list and their criticalities
OutFile: report file

- update *h* with the frequency sets
- for each action *a* loop
  - find the drift type *d* in action *a*'s history
  - send the drift *d* for action *a* to the report manager
- the report manager generates the final report based on the criticality of each action, the current drift parameters and previous drift parameters.

Figure 4.2: Pseudo code for drift detection main loop

report manager generates the final report, which is based on the criticality of each action, the current drift parameters and previous drift parameters.

4.3.2.2 Find drifts in each action

Different tests are preformed on the data for each action to detect different types of drifts. The test for *no-drift* is performed first. This has the highest priority. If this test succeeds, then the remaining tests will not be performed and the action will be classified as *no drift*. Following the *no drift* test are the tests for *cyclic* and the *sloping* (*increasing* and *decreasing*) drifts. *Cyclic* drifts are given only a slightly higher priority than *sloping* drifts. The priority between these two is based on the confidence of a drift. The confidence of a drift is defined later in this chapter. Finally, if no classification was successful, the examined data is classified as *chaotic*. The result of these tests, which is the drift classification along with its parameters (if any) such as confidence and length of the drift, is sent to the report manager.

The tests for drifts on an action are executed using frequency data for one or more sample sizes and one or more corresponding window sizes. Graphical and statistical analysis is then performed on the corresponding temporal autocorrelation plots for each window to find the drifts. Temporal autocorrelation refers to the correlation between time-shifted values of a time series. It reflects the fact that the value at a given time point is not completely independent of its past signal values; indeed, there is a high dependence. A temporal autocorrelation plot [2] is one which shows the autocorrelation values of a time series with respect to increasing difference in time. The time difference is called the lag of the autocorrelation values and is plotted on the plot's

x-axis. These plots are a commonly-used tool for checking randomness in a data set. This randomness is ascertained by computing autocorrelations for data values at varying time lags. If the relationship between these time-offset data sequences is random, such autocorrelations should be near zero for any and all time-lag separations. If non-random, then one or more of the autocorrelations will be far from non-zero. This means that for drifts such as cyclic, increasing and decreasing we will see non-zero autocorrelation values. Refer to Appendix A for a complete definition of autocorrelation and autocorrelation plot. In our experiments we have used a slightly modified version for calculating the autocorrelation values. For values ($x_1 ... x_n$) with mean μ and standard deviation σ the autocorrelation value with lag k is defined as:

$$r_k = \frac{\sum_{i=1}^{n-k} (x_i - \mu) (x_{i+k} - \mu)}{(n-k) * \sigma^2}$$

Given action *a*, the algorithm for finding and reporting the drifts is summarized in figure 4.3.

4.3.2.3 Test for no-drift

Any data which remains constant for a significant period of time is classified as *no-drift*. We extended this definition to cover the data which is nearly constant (i.e., it may have a small amount of acceptable random noise). A threshold is set for the amount of allowable random noise in the data. Through experimental tests, visual analysis of the data, and generation of autocorrelation plots we have laid down some rules for detecting data with no drift.

Input: action a, history *h*, reporter r
OutFile: drift type *d* and its parameters *p*

- check if action a has drift type *d* == no drift
- if yes then
  - send the drift type and its parameters to the reporter
  - return to the calling function
- check if action a has drift type *d* == cyclic or increasing
- if yes then
  - send the drift type and its parameters to the reporter
  - return to the calling function
- send the drift type as chaotic to the reporter

Figure 4.3: Algorithm for finding and reporting the drifts

1. Only the top half of the autocorrelation plot corresponding to the given data is taken to determine the *no-drift* in the data. This is because using autocorrelation values with lags larger than half the size of the window may result in loss of data points positioned in the middle of the window. These values are then analyzed using rules 2 and 3 to determine if there is no-drift or not.

2. For *no-drift* actions, the autocorrelation plot values should not exceed a given maximum value. In our experiments we have set this threshold to 0.4. This test eliminates the possibility of higher-autocorrelation trends such as *cyclic*, *increasing*, *decreasing* and *chaotic*.

3. If more than 10% of these values lie outside the *(m – 2s, m + 2s)* range, where *m* is the mean and *s* is the standard deviation of the values in the autocorrelation plot, then the variation or noise is considered too non-random to be classified as *no-drift* data.

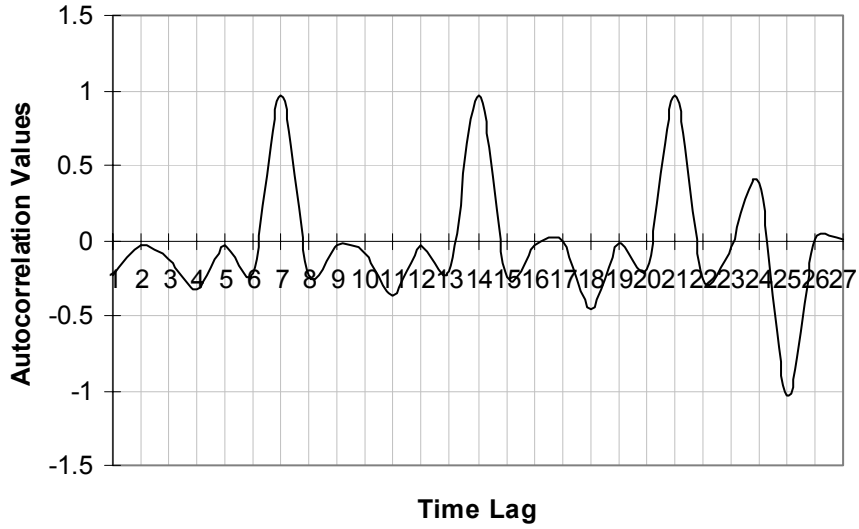
An autocorrelation plot is used instead of the actual data points since auto-correlation reduces the amount of random noise in the data, making the analysis easier. An auto-correlation of zero at a given lag means that the data distribution remains constant with respect to some time in the past or contains complete random noise.

The *no-drift* test is done only for the daily sample size using the largest window size first. Smaller windows are used only if the test fails for the larger windows. This means that even if the frequencies may be drifting in a large window, they may turn out to represent *no-drift* in the smaller window. For example, an action may show drift in a six-month-long window but may not show drift in a month-long window contained inside the larger sample of data.

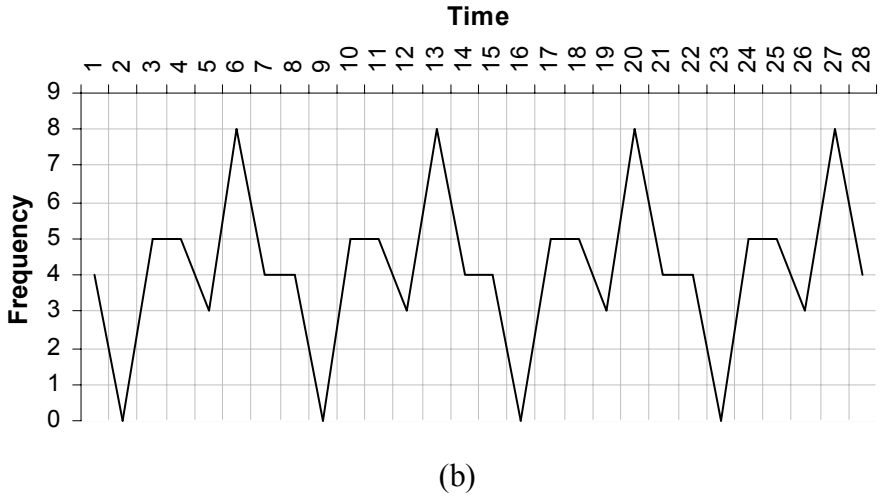
If the test is positive then the *NoDriftLength* is returned. This represents the maximum length in time for which the data remained fairly constant, without any drifts. This is usually equal to half the length of the window for which the test returned positive. *No-drift* tests hold the highest priority. If the result is positive then tests for the other drifts are not conducted.

4.3.2.4 Test for cyclic drifts

A *cyclic* trend shows high upward peaks in the autocorrelation graph. This is because correlation between cyclic values is high and drops when we deviate even slightly from the cycle. In figure 4.4(b), we have frequencies of an action which are *cyclic* with a cycle length of seven days. In the corresponding autocorrelation plot in figure 4.4(a), we see upward-facing high peaks at intervals of seven. This indicates that the length of the cycle is seven. Also from the figure we observe that the last quarter of



(a)



(b)

Figure 4.4: An example of a cyclic drift in frequency data in (b) and its corresponding autocorrelation plot in (a).

the autocorrelation values are a little skewed and do not represent the cycles in the data. We thus will look at only the first three-fourths of the plot to find *cyclic* peaks.

Our intention is to find the best long-term cycle, and for this purpose we try to find a set of three peaks positioned at regular intervals starting from the first value.

These peaks should always be greater than a minimum peak value. This minimum-value threshold is set to 0.3 in our experiments. Looking at three peaks allows us to ascertain that the cycle exists for at least four periods. If we need to find smaller *cyclic* behavior then we may use only the first or the first two peaks positioned at regular intervals. However, using only one or two peaks may not give the best long term cycles in a noisy data. Also going further than the third cyclic peak is not required as we get a good enough picture of the cycle from the first three peaks. In our experiment all upward-facing peaks with height greater than the minimum peak value are called *high peaks*. For every such set, all other peaks should be either smaller than the smallest peak in the set or should be positioned at a multiple of the same regular interval, at which the three peaks are positioned. This means that any higher peak not in the set should be caused because of the same *cyclic* behavior which the set represents. Now for every valid set of three peaks that is found, we calculate the *confidence* measure of the set. This is a real value between [0, 1). Calculation of confidence depends on the following factors:

- Average height of the first two peaks
- Average variation between the peaks

Thus we calculate confidence as

$$confidence = height - \alpha * variation$$

where α is some constant between [0.0, 0.5). If more than one such set is found we pick the one which yields the greatest confidence.

Input: minimum cycle length min_len, autocorrelation plot values array *v*, plot size *n*
Output: is_cyclic_in_this_window, best_confidence and best_cycle

- best_cycle ← 0
- confidence ← 0.0
- best_confidence ← 0.0
- construct a list *t* with pairs of value pairs *(i, j)*. Each value pair represents a high upwards peak in *v*, where *i* is the autocorrelation value and *j* is the lag of the peak.
- sort list *t* in descending order of their height (i.e., their *i* value)
- maximum cycle length, max_len ← ceil (total number of points in plot / 4.0)
- for each cycle length *y* between min_len to max_len
  - is_cyclic_in_this_window ← false
  - confidence ← 0.0
  - min_confidence ← 0.5
  - peak1 ← v [y]
  - peak2 ← v [2*y]
  - peak3 ← v [3*y]
  - if (peak1, peak2, peak3) belongs to *t*
    - unexpected_peak ← false
    - for each peak *p* in *t*, which is higher than min(peak1, peak2, peak3)
      - if the *j* value of peak *p* is not a multiple of *y* (i.e., *p* is not inside the cycle)
        - unexpected_peak ← true
    - if unexpected_peak is false
      - calculate the 'confidence' in the cycle *y*
      - if confidence > best_confidence
        - best_confidence ← confidence
        - best_cycle ← *y*
- if best_confidence > min_confidence
  - is_cyclic_in_this_window ← true
- return is_cyclic_in_this_window, best_confidence and best_cycle

Figure 4.5: Algorithm for the test for cycles in a given window

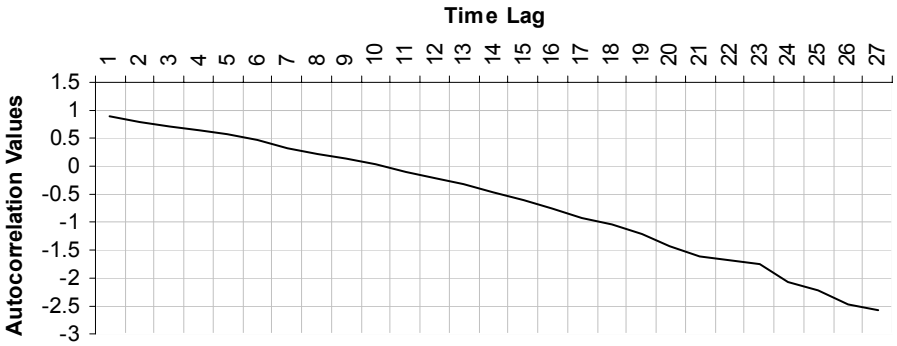
A summary of the algorithm to test for cycles in a given window is given in figure 4.5. Note that in this algorithm, we use only the first three-fourths of the autocorrelation plot values. Thus the autocorrelation plot values array, *v*, in the

algorithm will only have three-fourths of the values. Also, plot size *n* in the algorithm represents the total number of points in the plot.

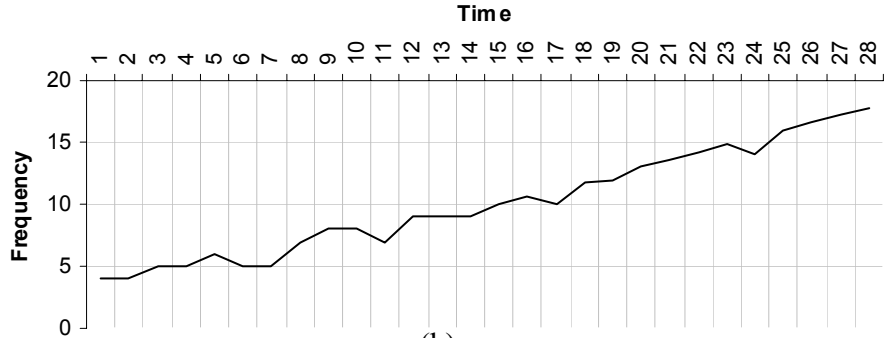
This experiment is repeated for all window sizes with a sample size of daily. Since we are collecting the data at the end of every day, we are not interested in finding cycles of length less than a day, which could be detected using the six-hourly sample size. Also we do not need to use the weekly sample size as large cycles can be detected even with the daily sample size. If more than one cycle is found, the cycle with the highest confidence is picked for further analysis. If no cycles were found or the best confidence is less than a minimum confidence measure (0.5), we simply move to test for sloping.

To make sure that this is the best classification we use two measures: a high confidence measure (0.7) and a minimum confidence measure (0.5). If the confidence of the cycle is greater than the high confidence measure, then it is sent to the report manager. If the confidence measure is in between the two (i.e., 0.5 to 0.7), then a test for sloping is performed to find slopes of length greater than the best cycle length. If this test fails, then the data is reported as *cyclic*, otherwise the confidence of a sloping drift is compared to the confidence that the drift is *cyclic*. The classification with the higher confidence is sent to the report manager. Finally, if both these tests fail, we conclude that the drift is *chaotic*.

For cyclic drifts the confidence and the cycle length are also sent to the report manager.



(a)



(b)

Figure 4.6: Example of an increasing sloping curve
with some noise in (b) and its corresponding autocorrelation plot in (a)

4.3.2.5 Test for Sloping (Increasing and Decreasing) Drifts

For *increasing* or *decreasing* trends, a high degree of autocorrelation is seen between adjacent and near-adjacent observations. The autocorrelation plot for this kind of data will show very high autocorrelation at lag one and will decrease as the lag increases, as can be seen in figure 4.6. We have used this property of autocorrelation plot for detecting *sloping* drifts. Again, the use of autocorrelation helps in suppressing some random noise in the data, thereby making long-term trend detection possible.

For sloping drifts, tests are first conducted on data with a sample size of daily. If this test fails then tests are done on data with sample size as weekly. For each sample we test all available windows for the longest slope. Different window sizes are used since it is easier to detect smaller slopes in small windows. This is because in large windows an inconsistent older distribution might change the autocorrelation values, making it difficult to find the slopes of smaller length.

This test is simple. Given a data window, if the autocorrelation at lag one is greater than a minimum sloping correlation measure (0.5), then it is considered to be sloping. For sloping data, the autocorrelation values keep on decreasing as the lag increases. Thus the length of the slope is the smallest lag at which the autocorrelation stops decreasing. The drift is considered sloping only if its length is greater than a minimum sloping length and greater than the best cycle length (if available).

The direction of the slope can be determined by calculating the sum of the deviation in the adjacent data points which are within the slope length. If this value is positive than the slope is *increasing*, otherwise it is *decreasing*. The confidence in this drift is the autocorrelation value at lag one.

For drifts classified as sloping, the length of the drift, the confidence and the direction is sent to the report manger.

4.3.2.6 Chaotic

Anything which is not classified as any of the other drifts is classified as *chaotic*. This drift is caused by a large number of irregular changes or heavy random or non-random noise in data. It may also be caused by a few sudden large changes in the

distribution of the data. In synthetic data this type of drift is often seen for a short period of time when there is a sudden change in the type of drift. This means if the data suddenly changes from being purely cyclic to purely increasing then this drift is seen at the point of change. It can also be seen when the period of a cyclic drift changes. In real world, we can see this drift when we change our daily patterns as a result of some outside influence such as leaving for a trip, a short illness, finals, etc. There are no additional tests for this drift.

4.4 Outlier Detection Algorithm

We use two different methods to detect two different types of outliers. The first method extends the algorithm presented for drift detection for finding outliers at regular intervals of time. It uses the conventional method of z-scores for detecting outliers. The second method uses prediction techniques (Active LeZi [14]) to continuously detect outliers as time progresses and is important for reporting emergency situation's which may require immediate action. The inhabitant can be provided with a warning when ever this change occurs.

4.4.1 Frequency-based Outlier Detection

For this method, we define an outlier as an extremely high or low value when compared to the rest of the values. We have used the z-score method to detect outliers in the frequency values of the actions. The Z-score or Standard Score is the value obtained by subtracting the mean and dividing by the standard deviation. This method is based on Chebyshev's theorem [98]. It states that the proportion of the values that fall within k standard deviations of the mean is at least $1 - (1/k^2)$, where $k > 1$. Chebyshev's

theorem can be applied to any distribution regardless of its shape. For a normal distribution - that is, a distribution that is bell-shaped - the empirical rule [99] is that approximately 68% data points fall within one standard deviation of the mean; 95% data points fall within two standard deviations; and 99.7% within 3 standard deviations of the mean. This means that almost all the observations in a data set will have a z-score less than 3 in absolute value. That is, they will fall into the interval ($m - 3s$, $m + 3s$), where m is the mean and s is the standard deviation of the periodic frequency values. Therefore, the observations with z-scores greater than 3 will be considered outliers.

This method is used in conjunction with the drift detection algorithm. Even before we look for no drift trends we check if the most recent data is an outlier with respect to the rest of the window. If it is an outlier, then tests for drifts are not performed. The outlier is sent to the report manager. If it is not an outlier then we continue testing for drifts. In this method we also look for outliers in windows of different sizes belonging to the daily sample size.

One question that arises here is what one should do once an outlier is detected. We may feel that the outliers are so extreme that they could not reasonably have arisen by chance from the current model and are a mistake, thus should be rejected or corrected before continuing our analysis. Alternatively, these outliers may indicate the adoption of a new model, with important implication for the further analysis of the data and for this reason they should be included in our data. In our algorithm we assume that the outliers are a part of a new pattern learned by the system and thus should be kept in

the data. We note that by retaining the outliers in the data can affect the detection of the future drifts and outliers.

Another point to consider is a situation such as one in which a valid drift may be detected as an outlier. We take an example of a weekly cyclic drift on an action with frequency of the action being extremely high on one particular day of every week. This may repeatedly get detected as an outlier despite being a regular cyclic drift. To prevent such situations, we recommend as future work to use the drifts detected in the data to aid the detection of outliers. This may be done by first doing drift detection and then doing outlier detection based on the drift detection results.

4.4.2 Prediction-based Outlier Detection

The previous method for drift and outliers detection used the store-and-forward monitoring technique. For detecting emergency situations we need to use a live-monitoring technique. For this purpose we define an outlier to be the occurrence of an unexpected action in an ordered sequence of actions. The unexpected actions are considered irregularities that do not follow the expected pattern or routine followed by the inhabitant. We use the Active LeZi [14] algorithm to find the expected pattern in the data. Active LeZi (ALZ) employs the power of Markov models to optimally predict the next symbol in any sequence of symbols that can be modeled as a stochastic process. ALZ is based on the LZ78 data compression algorithm [100] which employs incremental parsing. LZ78 parses an input string x_1 , x_2 , …. x_i into $c(i)$ substrings w_1 , w_2 , …. $w_{c(i)}$ such that for all $j>0$, the prefix of the substring w_j (i.e., all but the last character of w_j) is equal to some w_i for $1<i<j$. Because of this prefix property, parsed

substrings (*LZ phrases*), and their relative frequency counts, can efficiently be maintained in a multiway tree structure called a trie. This trie is termed the *dictionary*. ALZ is a modified LZ78 algorithm, which uses the dictionary to calculate the probability of each state occurring in the sequence, and predicts the one with the highest probability as the most likely next action.

In our experiment, ALZ symbols represent the inhabitant actions we are monitoring, and the order of the sequence of actions is defined by the order these actions take place in time. ALZ is used to determine the probability distribution for each action at any point of time. When an action occurs we use this probability distribution to determine if this action was an outlier or not. For this purpose we calculate the anomaly measure for this action. We use two methods to calculate the anomaly measure. Our first measure assumes that the anomaly measure depend only on the probability of the next action and is not affected the complete probability distribution. It is given as:

$$\text{anomaly measure, } n1(x) = \begin{cases} 1, \text{ if } \rho(x) * 100 < 1 \\ \dfrac{1}{\rho(x) * 100}, \text{ otherwise} \end{cases}$$

where $\rho(x)$ is the probability of action x to be the next action and x is the new action observed. The factor of 100 is carefully chosen and included in the equation to make sure that anomaly measure for data points that are not anomalous lie within [0, 1). This measure may not be useful when probabilities are distributed between a large numbers of actions and thus we need a second measure. In the second method we use

both the current action x and the most probable action (action with the highest probability) y. It is given by:

$$\text{anomaly measure, } n2\ (x) = \begin{cases} 1, & \text{if } \rho\ (x) * 100 \leq \rho\ (y) \\ \dfrac{\rho\ (y)}{\rho\ (x) * 100}, & \text{otherwise} \end{cases}$$

Here $\rho\ (y)$ will always be greater than or equal to $\rho\ (x)$ and thus *n2* could be considered to be more selective than *n1*. This means that outliers labeled using only *n2* will have a greater probability of being an anomaly. Note that all outliers labeled using *n2* could also be detected using *n1*. Thus, we check for outliers based on *n1* only if the test using *n2* comes out to be negative. In *n2* we have used the assumption that anomaly measure increases with increase in difference between $\rho\ (x)$ and $\rho\ (y)$. Thus, *n2* takes into account the underlying distribution in probability values while calculating the anomaly measure.

For the purpose of reporting we calculate another value called the urgency factor. It is given as:

urgency factor, $u\ (x) = n(x) * c(x)$

where n is the anomaly measure and c is the ratio of the action criticality to the total number of criticality levels. Thus, $c(x)$ = criticality of action x / 5.0. We report an action as an outlier if $u\ (x) >= 0.1$. See Appendix B for example tables for calculating anomaly measures and urgency factors. In the first table the anomalies are marked in

Input: input_stream, actions and their criticalities
Output: outlier detection report

- *dictionary* ← null
- *window* ← null (window is a sequence of actions, used by ALZ)
- phase, *w* ← null (phrase is also sequence of actions, used by ALZ)
- *max_lz_length* ← 0
- probability distribution array, *p* ← null
- loop
  - wait on the input stream for next action symbol, *a*.
  - update *p* using the existing dictionary, and window
  - determine if *a* is an outlier based on information in *p* and the action criticality.
  - if *a* is an outlier then
    - report action *a* as an outlier.
  - use Active LeZi Algorithm to add *a* to dictionary and update the frequencies of all possible pattern sequences. The phrase *w*, *window* and *max_lz_length* are used and updated in this step.
- forever

Figure 4.7: Algorithm for outlier detection using prediction-based method.

bold font. An alternative to reporting outlier with $u(x) >= 0.1$, we may rank the outliers according to their urgency and report them all to the caregiver.

A summary of the algorithm is given in figure 4.7. The process of adding action 'a' to the dictionary using ALZ is summarized in figure 4.8.

4.5 Generation of Report by Report Manager

For drifts and outliers based on frequency-based method, a report file is generated at the end of each day. The report manager uses the current classification, the previous classification, the criticality of the action, and other parameters like confidence and length of drifts to decide what to report. A report can be generated at three levels.

Input: action a, dictionary, window, phase w and max_lz_length

- if ((w.a) in dictionary)
  - w ← w.a
- else
  - add (w.a) to dictionary
  - update max_lz_length if necessary
  - w ← null
- add a to window
- if (length(window) > max_lz_length)
  - delete window[0]
- Update frequencies of all possible contexts within window that includes a

Figure 4.8: Algorithm for adding action 'a' to dictionary using ALZ [14]

4.5.1 Level One Report

This is a report listing all critical drifters or outliers. The general rule for outliers is that if the criticality for the action is above 3 (i.e., the criticality is high) then it is added to this list. The general rule for drifts is that if the criticality for the action is above 3 and the classification type has changed from the previous one, then it is added to this list. In addition, for cyclic drifts if the length of the cycle for the action is above 3, it is added to this list.

4.5.2 Level Two Report

This report includes the level one report and adds to it a list of all important drifts and outliers, along with other parameters such as length, criticality and change from previous classification. At this level separate lists are made for the outlier and each drift type. All outliers irrespective of the criticality are listed here. For drifts the general rule is that it reported in this list if the classification changes, the criticality is above 3,

the length of the cycle changes, or the confidence changes by some amount. In our experiments we consider the change to be important if the confidence changes by an amount more than 0.05. An example of such a drift from the report in figure E.6 is: NoisyIncreasing_On (3) is increasing for a length of 6 Days with confidence 0.788725, with change of 0.0880085 in confidence.

4.5.3 Level Three Report

This report includes the level one report and adds to it a table, which lists the drift type or outlier along with all their parameter for every action in the environment.

4.6 Other Methods and Related Work

Over the past 20 years, a lot of work has been done in machine learning and statistics to understand the patterns and thus the changes in a data stream. Most of these works assume that the environment is stationary and require a large number of training examples in advance. The concentration in these works is usually on detecting either small term (local) drifts or long term drifts. Small term drifts were largely ignored as noise. In contrast, humans learn in changing environments and may cause drifts which could vary in size from a few days to several weeks or months.

Jin et al. [74], describes a fast and efficient algorithm for detecting local outliers. He uses a 'micro-cluster' based data compression method for calculating the degree of outlying, dependent on the density of the local neighbors. This required analyzing neighboring data on either side of the outlier. This is not possible in a time series, where we do not have data representing days in the future.

Some of the earlier systems used for handling concept drift are STAGGER [66], IB3 [67], and FLORA [68]. More recently, many systems have been proposed to handle specific problems such as vowel recognition [69], data on flight simulators [70], web page access [71], credit card fraud [72], and spam mail filtering [73].

For outlier detection many different techniques have been used. Helman and Bhangoo [15] present a statistical method to determine sequences which occur more frequently in intrusion data as opposed to normal data. Lee, et al. [16] [17] use a prediction model trained by a decision tree applied over the normal data. Ghosh and Schwartzbard [61] use neural networks to model normal data. Lane and Brodley [62] [63] [64] examine unlabeled data for anomaly detection by looking at user profiles and comparing the activity during an intrusion to the activity under normal use. HMS fulfills the need for developing a drift and outlier detection system which is specific to health monitoring.

Most of the work done in outlier and concept drift detection used one of the two general approaches. The first approach used large amount of normal (or clean) data to build a concept of normal data. Deviation from this concept was considered as change, which could be classified as drift or outlier. The second approach used a feedback mechanism or labeled training examples to build a notion of an outlier or drift. In health monitoring, it is difficult to provide example training data which could encapsulate normal behavior of all individuals. This is because a pattern which is normal for one person may not be normal for another. Also, in a real scenario of a home it is difficult to collect labeled data for training purpose. For HMS, we have assumed that no feedback

mechanism is available and thus could not use either of the two approaches. To detect drifts and outliers in the data we have thus used fixed notions of what a drift is and what an outlier is.

Barnet and Lewis [65] have described many methods using the second approach. One of this was used by Eskins [66], who used 'mixed models' [65] for outlier detection. In the mixture model, each element falls into one of two cases: with (small) probability λ, the element is an anomalous element and with probability $(1-\lambda)$ the element is a majority element or a normal element. This model used two description sets to build the final concept description or hypothesis.

Similar methods in which the hypothesis was represented as two or more description sets have also been used in detecting concept drifts. One example is a two-description-set approach in which one represents only the positive examples and the other keeps both positive and negative examples. Widmer, et al. [68] uses three description sets to represent the hypothesis. These are only positive examples; only negative examples; and all positive examples and some negative examples. Examples were then moved from one set to the other depending on the feedback from the system.

Till now, there has been little work done towards an integrated approach for finding both drift and outliers. We found out that temporal-autocorrelation gives an excellent picture of the drifting concepts and at the same time does not require any feedback mechanism to learn intelligently. Also, autocorrelation allows easy differentiation of different kinds of drifts. Compared to all these techniques autocorrelation method provides a general approach to drift and outlier detection by

allowing both sudden and gradual changes to be detected. This method, complimented with the prediction technique for outlier detection, gives us a complete system for monitoring changes in an inhabitant's health.

4.7 Conclusion

In this chapter we presented the HMS algorithm for finding drifts and outliers in a time series data. HMS algorithm uses two approaches. The first method uses statistical techniques such as autocorrelation plots and others to determine diurnal outliers and drifts (such as *increasing*, *decreasing*, *cyclic*, *chaotic* and *no-drifts*). The second method uses prediction techniques to continuously report outliers that may be caused due to unexpected change in the routine followed by the inhabitant.

CHAPTER 5

EXPERIMENTS

In this chapter, we verify the capabilities of HMS by the means of some experiments. These experiments are designed to be focused on different aspects of the algorithm. Experiments are conducted on synthetic and real data to verify the existence of different kinds of drifts and outliers using the two algorithms described in chapter 4. We start by explaining the nature of the data used for these experiments.

5.1 Nature of Data Used

We have tested HMS using both simulated data and real data. The real data was collected from a smart home environment called MavPad, which is an on-campus student apartment located at University Village on The University of Texas at Arlington's campus. MavPad is a part of the MavHome project also at The University of Texas at Arlington.

5.1.1 Synthetic Data

We have implemented a synthetic data generator that produces a time-ordered sequence of frequency or event sets of actions provided to it. The frequency sets are used by the autocorrelation and frequency-based algorithm and event sets are used by the prediction-based algorithm. The generator can create perfect or noisy drifts in the inhabitant patterns. It can generate increasing, decreasing, cyclic, constant and random patterns in the data. It can also be used to introduce outliers in the data.

We have chosen to generate a data set of hundred days with nine distinct actions. Let's call this *synthetic set one*. Five actions were designed with only one kind of drift, while the other four had one or transition from one kind to drift to the other. Noise was introduced in some of these actions to study the effect of this noise during classification. This was done by adding or removing a value of either one or two from the current frequency with a probability of 0.09. Note that the amount of noise is limited in a way that it does not destroy the underlying drift. The action name and the nature of drift in all the actions are listed in table 5.1. Note that the *increasing* or *decreasing* drifts in actions are long term drifts with gentle slopes. Outliers were introduced in the Changing_On action on the 71st, 72nd, and 88th day of the set. The criticalities for all the actions were chosen at random. The set contains a total of 10639 data points, with the number of data points for each action listed in table 5.1. This data is processed using the autocorrelation-based algorithm.

Table 5.1: Actions in the *Synthetic Set One*

Action name	**criticality**	**Description**
PerfectCyclic3_On	Medium(3)	A perfect *cyclic* with a period of three days. No noise.
DailyConstant_On	Low(2)	A daily constant (or *no-drift*) . No noise.
Cyclic3ToNoisyCylic7_On	High(4)	Change of cycle period from three days to seven on the 41st day of monitoring.
PerfectIncreasing_On	Medium(3)	Perfectly *Increasing* drift. No noise.
NoisyCyclic7_On	Low(2)	Noisy *cyclic* with period of a week
NoisyIncrToNoisyDesc_On	High(4)	Change from noisy *increasing* to noisy *decreasing* drift at the 41st day of monitoring
NoisyIncresing_On	Medium(3)	A simple long term increasing with noise.
Changing_On	Lowest(1)	A drift constantly changing between constant, *increasing* and *decreasing* with *outliers* in the data.
NoisyDecrToCyclic3_On	Low(2)	Changing from noisy *decreasing* drift to a *cyclic* with period three days.

Another set called *synthetic set two* is created for the testing of the prediction-based algorithm. This is a small set with four actions and 100 data points (see table 5.2). It is created as an ordered sequence of events with regular patterns in the data. We have introduced sudden changes in the pattern to simulate outliers. These changes mimic emergency situations which should be successfully detected as outliers by the algorithm. In the initial of the data we see changes in the form that new actions are observed for the first time. Later we form a set of patterns between these actions. For example Lamp_On is usually seen in the following sequence of actions: Motion_On, Motion_On, Cabinet_Open, Cabinet_Open, Computer_On, Lamp_On, Lamp_On. A sudden change to this pattern is seen when Lamp_On occurs right after the sequence Motion_On, Motion_On on January 7, 2005 at 2 AM. More of similar changes are introduced in the data. The criticality for each action is again chosen at random.

Table 5.2: Actions in the *Synthetic Set Two*

Action Name	Criticality	Number of data points
Motion _On	5	39
Cabinet _Open	3	21
Lamp _On	3	28
Computer _On	4	10

Two more sets called the *synthetic set three* and *synthetic set four* are defined which have the same data as synthetic set two but with criticalities kept constant at one and five respectively. These sets are used for the prediction based algorithm to understand the effect of using criticality for reporting.

5.1.2 Real Data

For this we used two types of data for our experiments. The first is inhabitant activity data from the MavPad. The second consists of health data collected by the same inhabitant.

5.1.2.1 Activity data from MavPad

The MavPad consists of a living/dining room combination, a kitchen, a bathroom, a large bedroom, and a walk-in closet. A number of networks are set up in the house to monitor various activities throughout the house. The Argus Sensor Network is the main perception system in MavPad. It is build around a core Argus Master Board. The Master in combination with Superslaves (separate component boards that relay sensor information to a Master board) and Dongles (separate component boards that host up to four sensors and connect to Superslaves) form the Argus Master-Slave (ArgusMS) network. The master is also extended into Argus and Argus Motor (ArgusM) networks. Argus D allows only pure digital I/O form the network and ArgusM allows control of stepper motors. The ArgusD network deployed at MavPad, supports forty sensors and is used to monitor motion as well as couch and chair seat switches. The motion sensors (see table C.1) are labeled as (V1...V38), the couch seat switch as V39 and the chair seat switch as V40. An X-10 powerlined/based controller is used to monitor electrical outlet usage, light usage and the overhead fan. X-10 is divided into three zones: a, c, and i, supporting sixteen sensors each; (a1..a16, c1..c16, i1..i16).

For our experiments we will use data[1] [95] [96] from only the ArgusD and the X-10 networks. Refer to Appendix D for the layout of these two networks in the MavPad. Each sensor in these two networks has two possible states: 0 and 1. Thus, we represent our actions as sensorname_state (for example, a1_0, a1_1, V12_0, V12_1 etc.). The input to HMS is taken as the list of events represented by the tuple:

<date_n_time sensor_name state >

Here *sensor_name_state* is an action which has occurred at time *date_n_time*. For drift detection and frequency-based outlier detection, HMS parses all the events belonging to every quarter of a day and partitions each into frequency sets. These frequency sets are then used as input to the algorithm. For the outlier detection algorithm based on prediction, the event list is directly fed as input to the algorithm.

We use around seven weeks of data collected from the MavPad. This is divided into two sets. Let's call these sets *real set one* and *real set two*. *Real set one* uses data from only X-10 where as *real set two* uses data from both ArgusD and X-10. *Real set one* is used only to test the prediction-based outlier detection. It contains 2163 data points with 79 distinct actions. The *real set two* contains 334935 data points with 157 distinct actions. The criticality of all the actions is kept constant at medium (or 3).

5.1.2.2 Health data collected by the inhabitant

A wrist wearable device was used by the inhabitant to record health data such as the systolic[2], diastolic[3] and heart rate[4]. The data was collected on an average once a day

[1] This data is (C) Copyright 2005 G. Michael Youngblood. All Rights Reserved.
[2] Systolic refers to the highest, peak, pressure within the blood stream occurring during each heart beat.
[3] Diastolic refers to the lowest, nadir, pressure within the blood stream which occurs between heart beats.

during a period of three months. We refer to this data as the *health set*. Due to inconsistency in data collection the data for some of the days is missing and some days have multiple recordings. We manually filled in the missing values using an average recording of the previous two days and the next two next days. In addition, multiple recordings for a single day are substituted by the average of these recordings. Since this data is collected only once a day we associate each action (i.e. systolic, diastolic and heart rate) with their actual reading instead of the frequency of reading. Thus, for drift detection we use the frequency set:

$<start\_time\ duration\ x_{systolic}\ x_{diastolic}\ x_{heart\_rate}>$

where $x_{systolic}$, $x_{diastolic}$ and x_{heart_rate} are the values of systolic, diastolic and heart rate, respectively, for *duration* of a particular day starting from the *start_time*. To allow this to work with our current algorithm we substitute this daily tuple with four six-hourly tuples:

$<start\_time_1\ duration\ x_{systolic}\ x_{diastolic}\ x_{heart\_rate}>$

$<start\_time_2\ duration\ 0\ 0\ 0>$

$<start\_time_3\ duration\ 0\ 0\ 0>$

$<start\_time_4\ duration\ 0\ 0\ 0>$

Here the *duration* is six-hours and $start\_time_i$ is the start times for each of the quarters. In essence, the values of these measurements play the role of frequency values in the HMS drift detection algorithm. As a result, we will detect drifts and outliers in the health values the same way frequency changes will be detected for action data.

[4] The heart rate is the number of contractions of the heart in one minute. It is measured in beats per

Since the changes in this data are of big importance we have given them the highest criticality level of 5. To indicate that the inhabitant was not able to record this data on a particular day, we put use zero for the missing values on the 31st day of recording. We observe that averaging due to missing values has created some short length increasing and decreasing patterns in the first half of the data set and some constant patterns in the later half. Since the order of these measurements does not hold any importance we do not use the prediction method for detecting outliers on this data.

5.2 Experiments Using the Autocorrelation and Frequency-based method

We ran three experiments to test the ability of HMS to detect diurnal drifts and outliers using the autocorrelation and frequency-based method. The first experiment uses the *synthetic set one* to verify the suitability of HMS for detecting these changes. For this purpose we follow a three-step verification process. First we examine the results for actions which have only one drift with no noise to determine if the classification is correct. Next, we try to determine if drifts can be detected in actions containing a noisy distribution. Finally, we look at data, where there is a change in the type of drift, to see if the algorithm is successfully able to detect the transition. Verification for the presence of outliers is much easier. For this we simply run the algorithm on an action known to contain outliers. In the second and the third experiments we will observe how this algorithm works when applied to activity and health related data. These experiments are run on the real set two and the health set. All

---

minute (bpm).

of these experiments were executed with report level set to 3. We use window sizes of two weeks, two months and six months for all of the experiments.

5.2.1 Experiment #1 Observations

This experiment generates 109 report files. From this set, one hundred files contain the report for each of the hundred days and nine files list the changes in the individual outliers. Since the minimum window size is fourteen, drift and outlier reporting starts with the fourteenth report.

In the first of the three-step verification process we observe actions which did not have any noise and are embedded with only one type of drift. These are DailyConstant_On, PerfectCyclic3_On, and PerfectIncreasing_On. DailyConstant_On was classified as *no-drift*, PerfectIncreasing_On as *increasing* and PerfectCyclic3_On as *cyclic* for all the days. This is a positive result for this step.

In this step we observed that the confidence for all these drifts were very high and the length of DailyConstant_On and PerfectIncreasing_On drift increased as the number of days with that drift increased. For PerfectCyclic3_On, the correct cycle period of three was correctly detected. It was also observed that the confidence of increasing drift in PerfectIncreasing_On decreased on days when the slope of the drift became very gentle or nearly constant. With these observations, we conclude that confidence is affected by the intensity and the length of the drift. This is good news as it gives us additional information about the drift.

In the second step, we look at the actions which have some noise embedded in them. First we consider the NoisyIncreasing_On action. We observe that it was

successfully classified as increasing on most of the days. However, on some days, when the amount of random noise increased and the rate of increase slowed down to nearly constant, the data was classified as *no-drift*. This is consistent with the definition of *no-drift* which allows a certain amount of random noise in nearly-constant data. On one particular day we see that this action is classified as chaotic, as the distribution changes from *no-drift* to *increasing*. At this point the algorithm found that the distribution was too skewed to detect the *increasing* drift. A section of the report on NoisyIncreasing_On can be seen in figure E.1. The second action with noise, which is NoisyCyclic7, gets classified correctly as cyclic every time, but exhibited lower confidence then the drifts containing no noise. Thus, we conclude that our algorithm can be highly successful in detecting drifts in noisy distribution and that high noise can cause confidence to reduce and sometimes even result in *no-drifts*. Also we note that resistance to noise increases as the length of the drift increases.

Actions which depict a shift in lifestyle, in particular the ones for which either the type of drift changes or the period of a previous cycle changes, gave some interesting results for the verification of the third step. The general rule seen here was that the confidence of the current classification decreases sharply for a few days after a shift in lifestyle. The algorithm then classifies the action as *no-drift* or *chaotic* for a period of time until the confidence in the second drift increases enough for it to be correctly classified. A few outliers were observed on days when the change was big and abrupt. This process is clear when looking at figures E.3, E.4 and E.5. In figure E.5 we see that it takes 28 days for HMS to detect a cycle of period seven. This is consistent

with our design discussed in chapter 4 that a cycle needs to be present for at least four periods for it to be detected. Thus we can say that this algorithm still works, even though it returns *chaotic* or *no-drift* classifications for some number of days. It is able to bounce back quickly and return the second drift correctly. Occurrence of *chaotic* drifts can thus be attributed to a change in distribution either due to excess noise in the data or due to a sudden change of drift type.

For the Changing_On action it was observed that classification varied between *increasing* and *decreasing*, with an occasional *no-drift* in between. The confidence generally remained low because of the changing nature of the distribution. Outliers inserted in this action came out correctly. The outliers on the 71st and 72nd day were detected in a window on 35 days. In contrast, the outliers on the 88th day required a seven-day window to detect the outlier. This outlier used a smaller window of seven instead of larger window of 35 since the outliers on the 71st and 72nd days did not allow these outliers to be detected. See figure E.2 for a section of this report.

In an additional observation we found out that the drifts in critical-actions got correctly elevated to the important and critical levels, when the drift type changed or a big change was seen in the length or confidence of the current drift. Figure E.6 shows a report for one of the days.

To summarize our observations, these experimental results validated that HMS can provide a good classification for the changes in the data and thus, can be used to gain good knowledge about the changes in our lifestyle. More specifically, we found that the confidence decreases with noise and increases with length of the drift. Sudden

transitions between *sloping* and *cyclic* drifts and between *cyclic* drifts of different periods may result in *outliers*, *chaotic* drifts and *no-drifts* for a number of days. Also our reporting mechanism helps to easily sort out important and critical drifts.

5.2.2 Experiment #2 Observations

Now that we have proved that our algorithm works, we would like to test HMS on the real data. We run the algorithm on the real set two and make the following interesting observations:

- On most days, actions were largely classified as *no-drift* or *chaotic*. *No-drift* can be explained as the inhabitant's behavior does not change much in just seven weeks. *No-drift* was especially dominant in rarely-used devices such as electrical outlets and some lamps. In most others *no-drift* and *chaotic* drifts were seen in equal proportions.

- *Increasing and decreasing* trends were seen on some of the days in motion detector data. This tells us that the inhabitant is spending increased or decrease amounts of time in the house. *Increasing and decreasing* trends were also seen on some of the days in the use of the floor lamp in the living room. The closet light in the bed room was observed to have an i*ncreasing* drift.

- *Cyclic* drifts were the rarest of all. Only two were observed in the data. The first one was a ceiling light in the living room and the second was the counter light in the kitchen. Interestingly both had a cycle of three days with low confidence.

5.2.3 Experiment #3 Observations

The monitoring of health data was particularly useful as it provided an excellent insight into the inhabitant's changing health. We observed every type of pattern including *increasing*, *decreasing*, *chaotic*, *no-drift* and *outliers* in the data. Outliers were successfully detected on the 31st day (2005, 04 14) of the set for all three actions. Increasing and decreasing drifts of length as small as three days were found, allowing us to detect even the short term changes. Lets try to visualize the result for systolic. For this we define a graph confidence as follows:

$$\text{graph confidence} = \begin{cases} c, & \text{if } \textit{increasing} \\ -c, & \text{if } \textit{decreasing} \\ 0, & \text{if } \textit{chaotic} \\ 1, & \text{if } \textit{no-drift} \\ 1.5, & \text{if } \textit{outlier} \\ -1.5, & \text{if } \textit{cyclic} \end{cases}$$

where c is the confidence of a decreasing or increasing drift. As discussed in chapter 4, we know that c lies between (0..1). Thus a negative graph confidence means that the classification is *decreasing*. The plot for systolic and graph confidence against the number of days is given in figure 5.1. For the first thirteen days confidence is not calculated as the minimum window size that we used for this experiment was fourteen. We see that on day 31 the systolic is zero and the corresponding graph correlation is 1.5, which means it is an outlier. We also see in the graph that systolic slowly decreases between days 10 and 23. Correspondingly, the graph confidence shows decreasing drift during this period. The report gives the length of the *decreasing* drift on day 21 as eleven days. This can be verified from this graph.

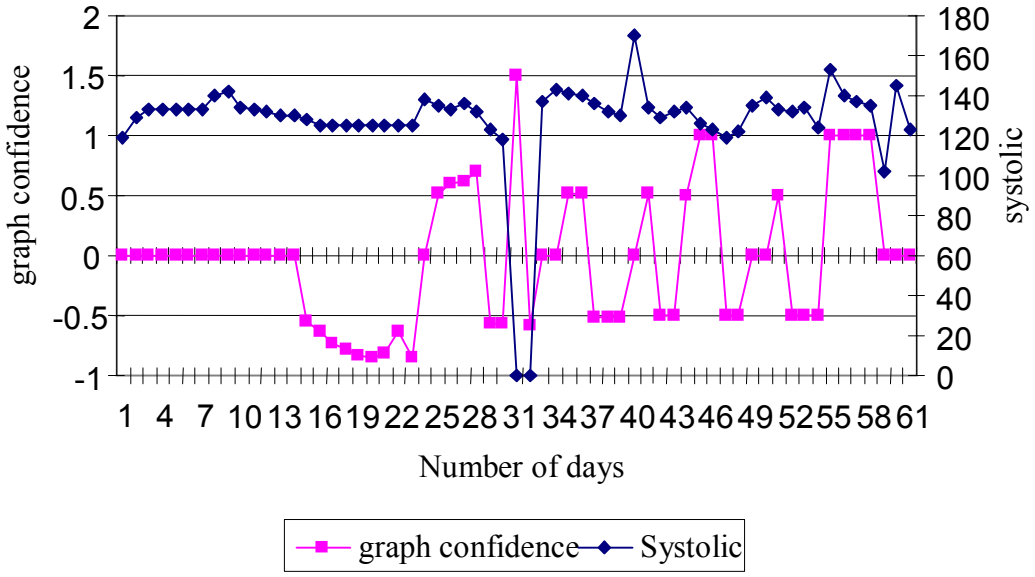


Figure 5.1: Line graph for graph confidence & systolic vs. number of days for health set

Between the 25th and 28th days an *increasing* drift is reported despite the slight decrease in systolic on days 25 and 26. This is because in the larger window of a few weeks these values are still higher than before. This observation tells us that our algorithm can account for slight noise in the system. Also as observed in the previous experiments, we see that the action often gets classified as *chaotic* when there is a transition from *increasing* to *decreasing*. Also we observe that sudden changes which are not outliers such as those on days 23, 33, 40, 49 and 59 gets classified as *chaotic*. This is because for a short time the distribution is too skewed for the algorithm to be able to find a regular drift. Thus, a *chaotic* drift following another type of drift could mean that there is a change in the distribution. Towards the later days we see that there is some random noise in the systolic measure and thus gets repeatedly classified as *no-drift*. We note that the high value on day 40 could have been an outlier, but went

undetected because the outlier at 31 may have increased the standard deviation. Also this outlier could have caused a change in autocorrelation values in the in the windows containing this data and result in a different drift classification. In a separate experiment, we substituted values at 31st and 32nd days as 125. This resulted in the detection of the outlier at 40 and *no-drift* for all the points between days 41 to 59.

The large amount of changes in the drift in systolic of this person does not necessarily means that the person is unhealthy but could be an expected pattern in this measure. Similar graphs for diastolic and heart rate are given in Appendix D, figures D.9 and D.10.

Thus we conclude that the algorithm is sensitive to sudden large changes and at the same time could detect drifts due to long term trends even with small amounts of noise. This data could prove to be very useful to a professional healthcare provider to monitor changes in vital signs and other vital health data of a chronically ill patient.

5.3 Experiments Using the Prediction-based Method

We first execute our algorithm using *synthetic set two* to see if the prediction-based analysis method is able to detect the intended outliers. Next we use a much larger set of real data (real set one) to analyze how this algorithm works. Finally, we execute the algorithm on *synthetic set three* and *synthetic set four*, and compare the result with that of *synthetic set two*. The reports for outlier detection in *synthetic set two* is given in figure D.7, a part of the report for *real set one* is given in figure D.8, and reports for *synthetic set three* and *synthetic set four* is given in figure D.11 and figure D.12 respectively.

From the first two experiments, we observe that the algorithms may take some time to learn the patterns of the inhabitant. During the initial period, which could be from a couple of days to a few weeks, we see a high number of outliers in the data. This is because all the patterns that the algorithm finds are new to it and it needs time to learn them. The length of the learning period usually depends on the number of actions in the data sets and the complexity of the patterns formed by these actions. We see that most outliers during the learning period are dramatic enough to detect them using the anomaly measure *n2* (defined in chapter 4). For the synthetic set we observe that the learning period is about three days. For real set one this is harder to determine and it may be between two to four weeks. As time progresses, we see that fewer data points are labeled as outliers. This is because now more and more events fall in the learned pattern. We also see that in the post-learning period the number of outliers detected using anomaly measure *n2* reduces and the one detected using anomaly measure *n1* increases with time. In synthetic data, which had a large number of predictable patterns, we found four outliers in the post learning period. These correctly pinpoint the big changes in the underlying pattern.

Finally, we found that the criticality of outliers can affect the changes of an anomaly to be reported as an outlier. For example Motion_On, which was assigned the highest criticality (a critical of 5), was detected as an outlier with anomaly measure of 0.102281. This means that the urgency factor for this action is also 0.102281 (as the criticality is 5 and urgency factor = anomaly measure * criticality/max_criticality, see chapter 4). This event will be classified as an outlier because the anomaly value is

greater than 0.1, which is the maximum value set for non-outliers. This would not been an outlier if the criticality was less than five.

To understand the role of criticality in reporting we execute the algorithm on *synthetic set three* and *synthetic set four*. With *synthetic set three* we see that no outliers are detected after the initial learning period. With *synthetic set four*, two addition outliers were found compared to *synthetic sets two*. Both of these were found using only anomaly measure *n1*, indicating that these were of lesser intensity and were important only because the action associated with it had the highest criticality.

5.4 Conclusion

We conclude that the results from HMS can be used to gain information about different types of drifts and outliers that are part of the inhabitant's lifestyle. Healthcare providers, family members and even the inhabitant himself can benefit from this information, as it can give a long-term view of the inhabitant's changing health and habits. In addition, this information is useful during emergency situations caused by some unexpected sudden changes. Furthermore, the choice of the criticality factors can have a significant effect on the reported outliers, so these factors need to be chosen with care based on the priorities in the domain. More intelligent systems can be used in conjunction with this system to provide the inhabitant with reminders or calls for help, when an anomaly is detected.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Interest in the research in the field of health monitoring is growing despite its challenges. With our work we hope to contribute to this growing field and help it grow further. The focus of this thesis has been on the area of automated health monitoring, where sensors and other smart devices are used in a home to automatically collect data about the inhabitant, and algorithms are used to process this data into useful information about the inhabitant's lifestyle for the purpose of gaining knowledge about his health. We now conclude with a brief summary of our system, our results, and our recommended directions for future work.

6.1 Conclusions

HMS is an amalgamation of two algorithms for detecting drifts and outliers in the inhabitant's behavior in a smart home environment. In the first algorithm, we find diurnal frequency drifts and outliers in the action data for an inhabitant. It uses autocorrelation statistical techniques to differentiate between different classes of drifts (*increasing*, *deceasing*, *cyclic*, *chaotic*, and *no-drift*) and other statistical measures to find outliers based on frequency. These drifts and outliers are found from the pattern of occurrence of each individual action, taken independently from the others. In the second approach, we use a prediction-based algorithm (ALZ) to detect outliers at the time of their occurrence. This approach complements the frequency-based approach by finding

outliers in the time-ordered patterns formed by a collection of actions. We noted a number of interesting observations from our experiments on synthetic and real data.

HMS successfully detected all intended drifts and outliers. The measures used such as the urgency factor, anomaly measure and confidence gave us a good insight on the intensity of the change. We found out that confidence increases with an increase in the size of the drift and decreases with an increase in the amount of noise in the data. In addition, we noted that a sudden transition between *sloping* and *cyclic* drifts or between *cyclic* drifts of different periods may result in *outliers*, *chaotic* drifts and *no-drifts* for some days, indicating that a big change has occurred. We observed that the prediction-based algorithm has an initial learning period during which most data points will be classified as outliers. The number of detected outliers decreases after this period as the patterns of the inhabitant are learned. We found that the importance of reporting a drift or outlier increases with the criticality of the action as well as the size or nature of the change.

Thus, we conclude that HMS is capable of finding all the substantial or gradual changes in an inhabitant's lifestyle and providing a report on the important ones.

6.2 Future Work

In our approach we found changes in the patterns of individual actions. It would be interesting to extend our approach to examine commonly occurring sets of actions (also called interesting patterns). Interesting patterns can be mined from the history and their frequency distribution can be used to find drifts and outliers. Methods such as Episode Discovery[75] [18] and ALZ can be used to find these patterns.

It remains to be investigated, how the system will behave on a very long term real data (one that spans several months). We were not able to perform these tests as consistent long term real data was not available. The verification of such a system is possible only if this system is installed in a real environment and the inhabitant provides feedback on the changes the system detects.

The prediction-based method for detecting outliers can be extended to cover situations in which no action is observed for a long time. Gopalratnam and Cook [14] suggest a way to find the relative time between events using a Gaussian measure. They provide a distribution in time to predict the presence of future events. A reminder could be provided to the inhabitant if the expected events are not observed at the expected time.

It still remains to be examined whether HMS could be used to provide solutions for specific health monitoring problems such as fall detection and social interaction. Also, in order to minimize generalizing error in the system work can be done to provide a feedback mechanism which could help in discarding irrelevant data from the model and help the system learn better.

Finally, this system can be combined with other health care systems to form a hybrid system, which can exploit the complimentary strengths of different learning models.

APPENDIX A

AUTOCORRELATION

Autocorrelation is a mathematical tool used for analyzing a series of values, such as time or space domain signals. It is the cross-correlation of a signal with itself. Autocorrelation is useful for finding repeating patterns in a signal, such as determining the presence of a periodic signal which has been buried under noise. It is used for the following two purposes:

- To detect non-randomness in data.
- To identify an appropriate time series model if the data are not random.

Given measurements, x_1, x_2, ..., x_N at time t_1, t_2, ..., t_N, the lag k autocorrelation function is defined as

$$r_k = \frac{\sum_{i=1}^{n-k} (x_i - \mu) (x_{i+k} - \mu)}{\sum_{i=1}^{n} (x_i - \mu)^2}$$

where μ is the mean of the x_i values.

This function has the attractive property of being in the range [−1, 1] with 1 indicating perfect correlation (the signals exactly overlap when time shifted by k) and −1 indicating perfect anti-correlation.

When the autocorrelation is used to detect non-randomness, it is usually only the first (lag 1) autocorrelation that is of interest. When the autocorrelation is used to identify an appropriate time series model, the autocorrelations are usually plotted for many lags. This is called an autocorrelation plot.

APPENDIX B

ANOMALY MEASURE AND URGENCY FACTOR TABLES

Table B.1 shows a table of ugency factors using the anomaly measure *n1*. The values highlighted in bold font represent outliers.

Table B.1: Urgency Factors Calculated Using *n1* over a Range of Values.

p(action)	anomaly measure, *n*	action criticality *c* (action)	1.00000 0.20000	2.00000 0.40000	3.00000 0.60000	4.00000 0.80000	5.00000 1.00000
			urgency factor, *u*				
0.00001	1.00000		**0.20000**	**0.40000**	**0.60000**	**0.80000**	**1.00000**
0.00002	1.00000		**0.20000**	**0.40000**	**0.60000**	**0.80000**	**1.00000**
0.00003	1.00000		**0.20000**	**0.40000**	**0.60000**	**0.80000**	**1.00000**
0.00004	1.00000		**0.20000**	**0.40000**	**0.60000**	**0.80000**	**1.00000**
0.00005	1.00000		**0.20000**	**0.40000**	**0.60000**	**0.80000**	**1.00000**
0.00100	1.00000		**0.20000**	**0.40000**	**0.60000**	**0.80000**	**1.00000**
0.01000	1.00000		**0.20000**	**0.40000**	**0.60000**	**0.80000**	**1.00000**
0.02000	0.50000		**0.10000**	**0.20000**	**0.30000**	**0.40000**	**0.50000**
0.03000	0.33333		0.06667	**0.13333**	**0.20000**	**0.26667**	**0.33333**
0.04000	0.25000		0.05000	**0.10000**	**0.15000**	**0.20000**	**0.25000**
0.05000	0.20000		0.04000	0.08000	**0.12000**	**0.16000**	**0.20000**
0.06000	0.16667		0.03333	0.06667	**0.10000**	**0.13333**	**0.16667**
0.07000	0.14286		0.02857	0.05714	0.08571	**0.11429**	**0.14286**
0.08000	0.12500		0.02500	0.05000	0.07500	**0.10000**	**0.12500**
0.09000	0.11111		0.02222	0.04444	0.06667	0.08889	**0.11111**
0.10000	0.10000		0.02000	0.04000	0.06000	0.08000	**0.10000**
0.50000	0.02000		0.00400	0.00800	0.01200	0.01600	0.02000
0.60000	0.01667		0.00333	0.00667	0.01000	0.01333	0.01667
0.70000	0.01429		0.00286	0.00571	0.00857	0.01143	0.01429
0.80000	0.01250		0.00250	0.00500	0.00750	0.01000	0.01250
0.90000	0.01111		0.00222	0.00444	0.00667	0.00889	0.01111
1.00000	0.01000		0.00200	0.00400	0.00600	0.00800	0.01000

Table B.2 shows the anomaly measure *n2,* calculated over a range of different values for the probability of the current action *x* against a range of probabilities of the most probable action *y*.

Table B.2: Anomaly Measures Using *n2* over a Range of Values.

p (*y*) →	0.6000	0.5000	0.4000	0.3000	0.2000	0.1000
p (*x*) ↓	anomaly measure *n*					
0.0100	0.6000	0.5000	0.4000	0.3000	0.2000	0.1000
0.0150	0.4000	0.3333	0.2667	0.2000	0.1333	0.0667
0.0200	0.3000	0.2500	0.2000	0.1500	0.1000	0.0500
0.0250	0.2400	0.2000	0.1600	0.1200	0.0800	0.0400
0.0300	0.2000	0.1667	0.1333	0.1000	0.0667	0.0333
0.0350	0.1714	0.1429	0.1143	0.0857	0.0571	0.0286
0.0400	0.1500	0.1250	0.1000	0.0750	0.0500	0.0250
0.0450	0.1333	0.1111	0.0889	0.0667	0.0444	0.0222
0.0500	0.1200	0.1000	0.0800	0.0600	0.0400	0.0200
0.0550	0.1091	0.0909	0.0727	0.0545	0.0364	0.0182
0.0600	0.1000	0.0833	0.0667	0.0500	0.0333	0.0167
0.0650	0.0923	0.0769	0.0615	0.0462	0.0308	0.0154

APPENDIX C

MAVPAD

In figure C.1 we see a picture of MavPad with motion detectors installed on the ceiling connected to the ArgusD network.

Figure C.1: ArgusD ceiling mounted PIR motion sensors.

Figure C.2 shows the layout of the X-10 and ArgusM actuators inside MavPad. ArgusMS and ArgusD networks can be seen in figure C.3.

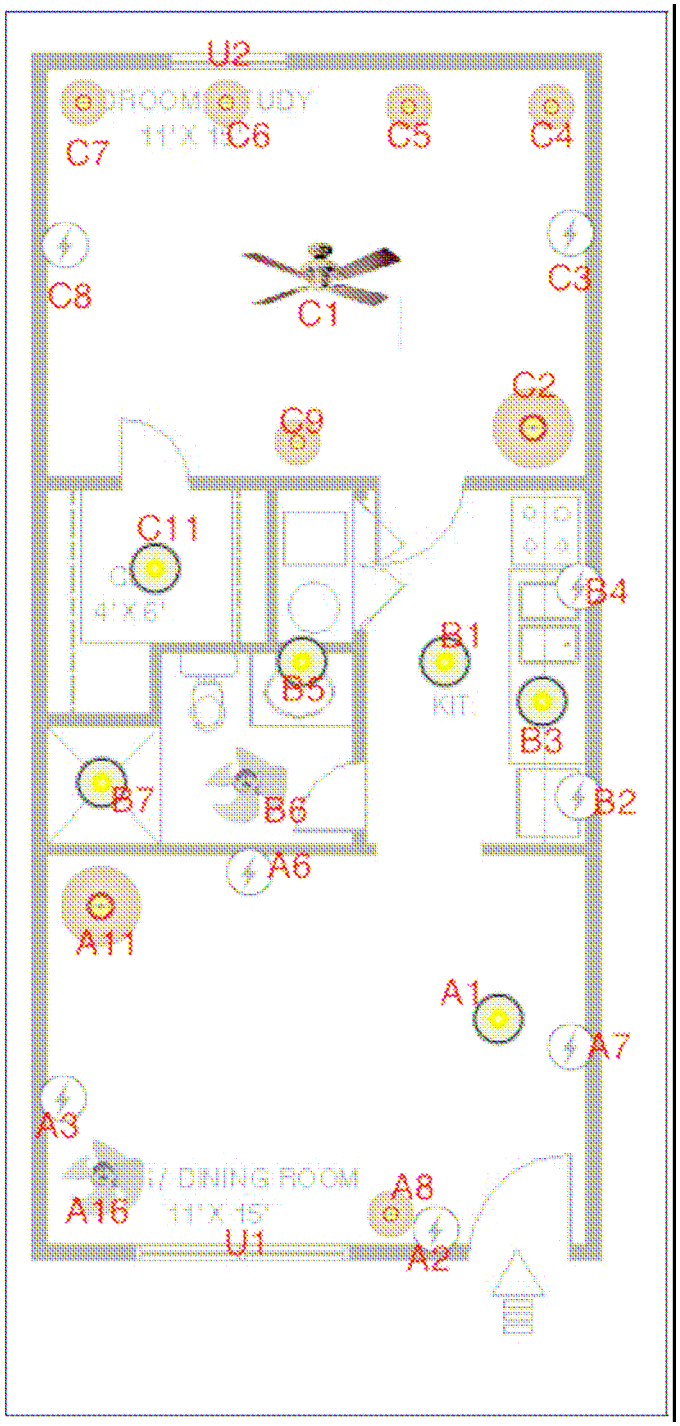


Figure C.2: MapPad X-10 and ArgusM actuators.

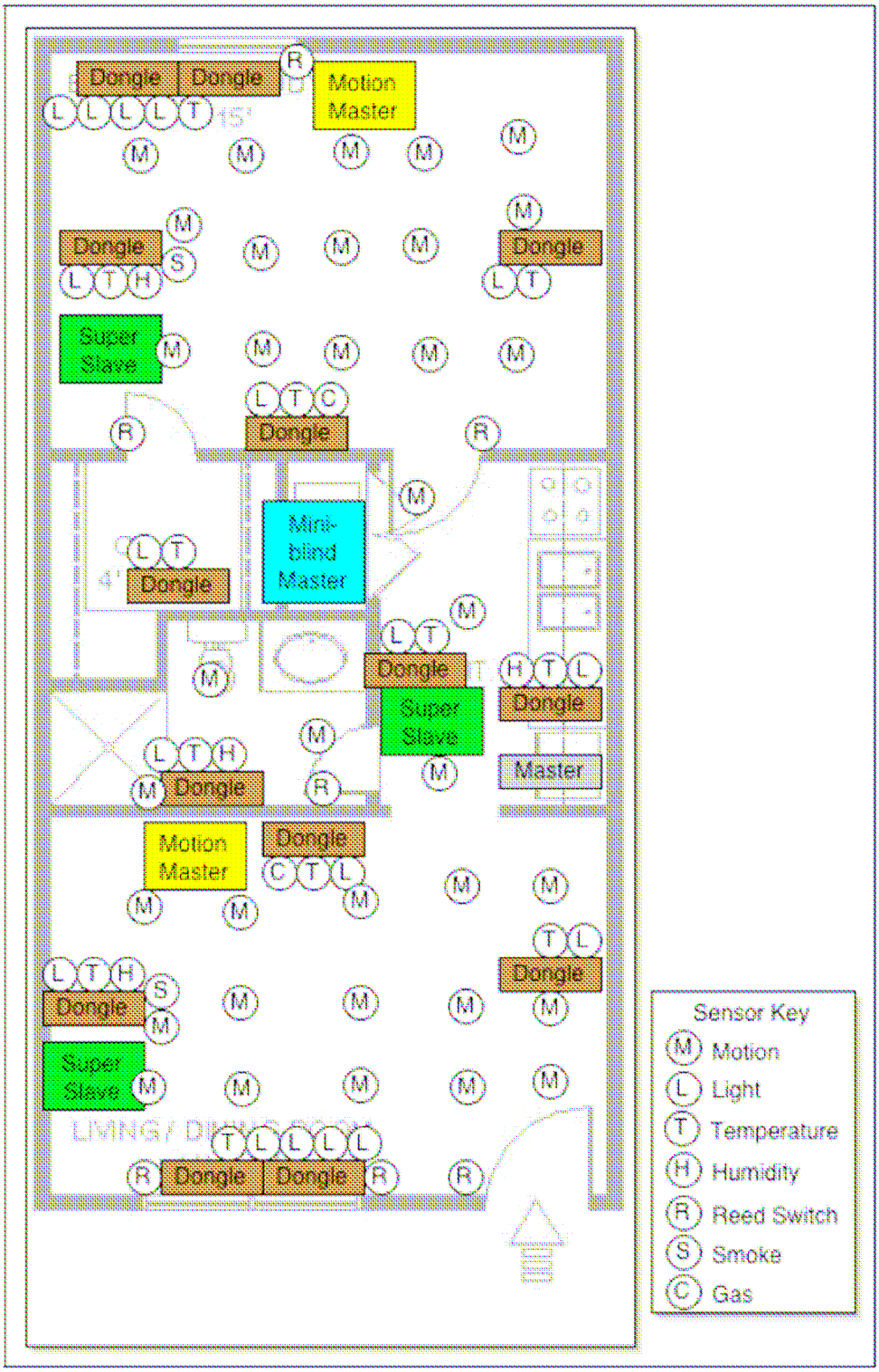


Figure C.3: MavPad ArgusMS and ArgusD sensors

Table C.1 gives a list of all ArgusD and X-10 sensors and actuator. Note that zone B in this table is the same as zone i in our data set.

Table C.1: ArgusD and X-10 Sensors and Actuators [97]

Zone	ID	Type	Source	Room and Location
A	1	Ceiling light	X10	Living Room. Overhead light near north wall
A	2	Electrical outlet	X10	Living Room. East wall near door, top outlet
A	3	Electrical outlet	X10	Living Room. Southeast corner of room, top outlet
A	6	Electrical outlet	X10	Living Room. West wall behind entertainment center, top outlet
A	7	Electrical outlet	X10	Living Room. North wall near table, top outlet
A	8	Table lamp	X10	Living Room. Near door on side table
A	9	Electrical outlet	X10	Kitchen. North wall over counter, top outlet
A	11	Floor Lamp	X10	Living Room. Southwest corner
A	16	Floor Fan	X10	Living Room. Southeast corner of room, top outlet
B	1	Ceiling light	X10	Kitchen. Overhead in kitchen
B	2	Electrical outlet	X10	Kitchen. Behind refrigerator
B	3	Counter lights	X10	Kitchen. Over counter, under cabinets
B	4	Electrical outlet	X10	Kitchen. Over counter, on left near oven, top outlet
B	5	Bathroom light	X10	Bathroom. Over mirror
B	6	Bathroom fan	X10	Bathroom. On ceiling. Will not turn off with remote.
B	7	Shower light	X10	Bathroom. Over shower
B	13	Argus Motor	X10	Kitchen. On top of fridge
C	1	Overhead fan	X10	Bedroom. On ceiling
C	2	Room light	X10	Bedroom. Northeast corner
C	3	Luxo lamp	X10	Bedroom. Attached to north bed
C	4	Luxo lamp	X10	Bedroom. On north computer desk
C	5	Luxo lamp	X10	Bedroom. On south computer desk
C	6	Luxo lamp	X10	Bedroom. Attached to south bed
C	7	Electrical outlet	X10	Bedroom. South wall behind bed, top outlet
C	8	Table lamp	X10	Bedroom. East wall, on dresser
C	10	Closet light	X10	Bedroom. In closet, on ceiling
V	1	Motion	ArgusD	Bedroom. Northwest corner of ceiling
V	2	Motion	ArgusD	Bedroom. 1 sensor south of V1
V	3	Motion	ArgusD	Bedroom. 2 sensors south of V1
V	4	Motion	ArgusD	Bedroom. 3 sensors south of V1
V	5	Motion	ArgusD	Bedroom. Southwest corner of ceiling
V	6	Motion	ArgusD	Bedroom. North center of ceiling
V	7	Motion	ArgusD	Bedroom. 1 sensor south of V6
V	8	Motion	ArgusD	Bedroom. 2 sensors south of V6
V	9	Motion	ArgusD	Bedroom. 3 sensors south of V6
V	10	Motion	ArgusD	Bedroom. South center of ceiling

Table C.1 - *Continued*

V	11	Motion	ArgusD	Bedroom. Northeast corner of ceiling
V	12	Motion	ArgusD	Bedroom. 1 sensor south of V11
V	13	Motion	ArgusD	Bedroom. 2 sensors south of V11
V	14	Motion	ArgusD	Bedroom. 3 sensors south of V11
V	15	Motion	ArgusD	Bedroom. Southeast corner of ceiling
V	16	Motion	ArgusD	Bedroom. On ceiling in closet
V	17	Motion	ArgusD	Kitchen. West on ceiling
V	19	Motion	ArgusD	Kitchen. Center on ceiling
V	20	Motion	ArgusD	Kitchen. East on ceiling
V	21	Motion	ArgusD	Bathroom. On ceiling over toilet
V	22	Motion	ArgusD	Bathroom. On ceiling over shower
V	23	Motion	ArgusD	Bathroom. On ceiling over bathroom door
V	24	Motion	ArgusD	Living Room. Northwest corner of ceiling
V	25	Motion	ArgusD	Living Room. 1 sensor south of V24
V	26	Motion	ArgusD	Living Room. 2 sensors south of V24
V	27	Motion	ArgusD	Living Room. 3 sensors south of V24
V	28	Motion	ArgusD	Living Room. Southwest corner of ceiling
V	29	Motion	ArgusD	Living Room. North center of ceiling
V	30	Motion	ArgusD	Living Room. 1 sensor south of V29
V	31	Motion	ArgusD	Living Room. 2 sensors south of V29
V	32	Motion	ArgusD	Living Room. 3 sensors south of V29
V	33	Motion	ArgusD	Living Room. South center of ceiling
V	34	Motion	ArgusD	Living Room. Northeast corner of ceiling
V	35	Motion	ArgusD	Living Room. 1 sensor south of V34
V	36	Motion	ArgusD	Living Room. 2 sensors south of V34
V	37	Motion	ArgusD	Living Room. 3 sensors south of V34
V	38	Motion	ArgusD	Living Room. Southeast corner of ceiling
V	39	Couch Seat Switch	ArgusD	Living Room. Couch - east wall
V	40	Chair Seat Switch	ArgusD	Living Room. Chair - southeast corner

APPENDIX D

EXPERIMENTAL RESULTS

In this section we show some of the reports produced during our experiments. In figure D.1, and figure D.4 we see the occasional *no-drift* and *chaotic* classification which might be seen in a noisy sloping drift. Figure D.2, shows a section of the report from a data set that changes between increasing and decreasing very often. We see sometimes it results in outliers in the transition point.

Time	Action	Type	confidence	Length
2005, 01 23	NoisyIncresing_On	IncreasingSlope	0.816711	11
2005, 01 24	NoisyIncresing_On	IncreasingSlope	0.875239	11
2005, 01 25	NoisyIncresing_On	IncreasingSlope	0.919976	10
2005, 01 26	NoisyIncresing_On	IncreasingSlope	0.917517	9
2005, 01 27	NoisyIncresing_On	IncreasingSlope	0.904406	9
2005, 01 28	NoisyIncresing_On	IncreasingSlope	0.876733	8
2005, 01 29	NoisyIncresing_On	IncreasingSlope	0.788725	6
2005, 01 30	NoisyIncresing_On	IncreasingSlope	0.699196	5
2005, 01 31	NoisyIncresing_On	IncreasingSlope	0.643725	4
2005, 02 01	NoisyIncresing_On	IncreasingSlope	0.592949	3
2005, 02 02	NoisyIncresing_On	NoDrift	1	7
2005, 02 03	NoisyIncresing_On	Chaotic	1	0
2005, 02 04	NoisyIncresing_On	IncreasingSlope	0.915799	27
2005, 02 05	NoisyIncresing_On	IncreasingSlope	0.927797	27
2005, 02 06	NoisyIncresing_On	IncreasingSlope	0.927874	27
2005, 02 07	NoisyIncresing_On	IncreasingSlope	0.929283	27
2005, 02 08	NoisyIncresing_On	IncreasingSlope	0.943109	27
2005, 02 09	NoisyIncresing_On	IncreasingSlope	0.930832	27
2005, 02 10	NoisyIncresing_On	IncreasingSlope	0.913884	27
2005, 02 11	NoisyIncresing_On	IncreasingSlope	0.896902	27

Figure D.1: A section of the report on NoisyIncreasing_On action

Time	Action	Type	confidence	Length
2005, 03 08	Changing_On	IncreasingSlope	0.862559	11
2005, 03 09	Changing_On	IncreasingSlope	0.875239	11
2005, 03 10	Changing_On	IncreasingSlope	0.869942	11
2005, 03 11	Changing_On	DecreasingSlope	0.77917	9
2005, 03 12	Changing_On	PeriodicOutlier	1	35
2005, 03 13	Changing_On	PeriodicOutlier	1	35
2005, 03 14	Changing_On	DecreasingSlope	0.773333	8
2005, 03 15	Changing_On	DecreasingSlope	0.825901	9
2005, 03 16	Changing_On	DecreasingSlope	0.864164	9
2005, 03 17	Changing_On	DecreasingSlope	0.876202	8
2005, 03 18	Changing_On	DecreasingSlope	0.854804	8
2005, 03 19	Changing_On	DecreasingSlope	0.810986	9
2005, 03 20	Changing_On	DecreasingSlope	0.852284	10
2005, 03 21	Changing_On	DecreasingSlope	0.8929	14
2005, 03 22	Changing_On	DecreasingSlope	0.894728	14
2005, 03 23	Changing_On	NoDrift	1	7
2005, 03 24	Changing_On	DecreasingSlope	0.909805	14
2005, 03 25	Changing_On	DecreasingSlope	0.91357	20
2005, 03 26	Changing_On	DecreasingSlope	0.888249	20
2005, 03 27	Changing_On	DecreasingSlope	0.908394	19
2005, 03 28	Changing_On	DecreasingSlope	0.914272	18
2005, 03 29	Changing_On	PeriodicOutlier	1	14
2005, 03 30	Changing_On	IncreasingSlope	0.762319	17
2005, 03 31	Changing_On	IncreasingSlope	0.798297	17

Figure D.2: A section of the report on Changing_On action

Transition between sloping and cyclic outliers and between cycles of different periods can cause some of the days to be classified as *chaotic* or *no-drift* or even outliers. This is evident from figure E.3 and figure E.5.

Time	Action	Type	confidence	Length/period
2005, 02 06	NoisyDecrToCyclic3_On	DecreasingSlope	0.926206	27
2005, 02 07	NoisyDecrToCyclic3_On	DecreasingSlope	0.936626	27
2005, 02 08	NoisyDecrToCyclic3_On	DecreasingSlope	0.943739	27
2005, 02 09	NoisyDecrToCyclic3_On	DecreasingSlope	0.948423	27
2005, 02 10	NoisyDecrToCyclic3_On	PeriodicOutlier	1	35
2005, 02 11	NoisyDecrToCyclic3_On	DecreasingSlope	0.880497	27
2005, 02 12	NoisyDecrToCyclic3_On	DecreasingSlope	0.856947	27
2005, 02 13	NoisyDecrToCyclic3_On	DecreasingSlope	0.850697	27
2005, 02 14	NoisyDecrToCyclic3_On	DecreasingSlope	0.898385	27
2005, 02 15	NoisyDecrToCyclic3_On	DecreasingSlope	0.880649	27
2005, 02 16	NoisyDecrToCyclic3_On	DecreasingSlope	0.869798	27
2005, 02 17	NoisyDecrToCyclic3_On	DecreasingSlope	0.896784	27
2005, 02 18	NoisyDecrToCyclic3_On	DecreasingSlope	0.873809	27
2005, 02 19	NoisyDecrToCyclic3_On	Chaotic	1	0
2005, 02 20	NoisyDecrToCyclic3_On	NoDrift	1	7
2005, 02 21	NoisyDecrToCyclic3_On	DecreasingSlope	0.858938	23
2005, 02 22	NoisyDecrToCyclic3_On	Cyclic	0.618328	3
2005, 02 23	NoisyDecrToCyclic3_On	Cyclic	1.02064	3
2005, 02 24	NoisyDecrToCyclic3_On	Cyclic	0.975368	3
2005, 02 25	NoisyDecrToCyclic3_On	Cyclic	0.959791	3

Figure D.3: A section of the report on NoisyDecrToCyclic3_On action

Time	Action	Type	confidence	Length
2005, 01 15	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.901951	11
2005, 01 16	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.90682	11
2005, 01 17	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.895152	11
2005, 01 18	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.861632	11
2005, 01 19	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.891333	11
2005, 01 20	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.837982	11
2005, 01 21	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.716884	11
2005, 01 22	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.632107	11
2005, 01 23	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.520362	11
2005, 01 24	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.525641	11
2005, 01 25	NoisyIncrToNoisyDesc_On	NoDrift	1	7
2005, 01 26	NoisyIncrToNoisyDesc_On	NoDrift	1	14
2005, 01 27	NoisyIncrToNoisyDesc_On	NoDrift	1	14
2005, 01 28	NoisyIncrToNoisyDesc_On	NoDrift	1	14
2005, 01 29	NoisyIncrToNoisyDesc_On	PeriodicOutlier	1	14
2005, 01 30	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.525641	11
2005, 01 31	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.69697	11
2005, 02 01	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.776923	11
2005, 02 02	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.818803	11
2005, 02 03	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.839744	11
2005, 02 04	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.895027	27
2005, 02 05	NoisyIncrToNoisyDesc_On	IncreasingSlope	0.877574	27

Figure D.4: A section of the report on NoisyIncrToNoisyDesc_On

Time	Action	Type	confidence	Length/period
2005, 02 06	Cyclic3ToNoisyCylic7_On	Cyclic	0.975368	3
2005, 02 07	Cyclic3ToNoisyCylic7_On	Cyclic	0.959791	3
2005, 02 08	Cyclic3ToNoisyCylic7_On	Cyclic	1.02064	3
2005, 02 09	Cyclic3ToNoisyCylic7_On	Cyclic	0.975368	3
2005, 02 10	Cyclic3ToNoisyCylic7_On	Cyclic	0.859892	3
2005, 02 11	Cyclic3ToNoisyCylic7_On	Cyclic	0.681689	3
2005, 02 12	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 13	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 14	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 15	Cyclic3ToNoisyCylic7_On	NoDrift	1	7
2005, 02 16	Cyclic3ToNoisyCylic7_On	NoDrift	1	7
2005, 02 17	Cyclic3ToNoisyCylic7_On	NoDrift	1	7
2005, 02 18	Cyclic3ToNoisyCylic7_On	NoDrift	1	7
2005, 02 19	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 20	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 21	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 22	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 23	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 24	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 25	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 26	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 27	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 02 28	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 03 01	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 03 02	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 03 03	Cyclic3ToNoisyCylic7_On	NoDrift	1	17
2005, 03 04	Cyclic3ToNoisyCylic7_On	NoDrift	1	17
2005, 03 05	Cyclic3ToNoisyCylic7_On	NoDrift	1	17
2005, 03 06	Cyclic3ToNoisyCylic7_On	NoDrift	1	17
2005, 03 07	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 03 08	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 03 09	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 03 10	Cyclic3ToNoisyCylic7_On	Chaotic	1	0
2005, 03 11	Cyclic3ToNoisyCylic7_On	Cyclic	0.501138	7
2005, 03 12	Cyclic3ToNoisyCylic7_On	Cyclic	0.657839	7
2005, 03 13	Cyclic3ToNoisyCylic7_On	Cyclic	0.774344	7
2005, 03 14	Cyclic3ToNoisyCylic7_On	Cyclic	0.795383	7
2005, 03 15	Cyclic3ToNoisyCylic7_On	Cyclic	0.913025	7
2005, 03 16	Cyclic3ToNoisyCylic7_On	Cyclic	0.955818	7

Figure D.5: A section of the report on NoisyIncreasing_On action

Figure D.6 shows a report for a particular day generated by HMS. This is a three-level report with drifts and outliers are reported as critical, important or all.

```
---------------------------------------------------------------
              This is a report is generated by HMS
---------------------------------------------------------------
REPORT DESCRIPTION:
       This file was made for this date: 2005, 01 29
NOTE:
       Criticality of the actions is kept in parenthesis.
       (1-lowest, 2-low, 3-medium, 4-high, 5-highest)

===============================================================
LEVEL 1 REPORT: Critical Drifters and Outliers
-----------------------------------------------
       NoisyIncrToNoisyDesc_On (criticality = 4) is an outlier in a window of
       length = 14 Days
===============================================================
LEVEL 2 REPORT: Important Drifters and Outliers
-----------------------------------------------
Outliers in data:
       NoisyIncrToNoisyDesc_On (criticality = 4)in a window of length = 14 Days

NoDrift data:
Cyclic data:
       Cyclic3ToNoisyCylic7_On (4) is cyclic for a length of 3 Days with
       confidence 0.959791

Increasing data:
       NoisyIncresing_On (3) is increasing for a length of 6 Days with
       confidence 0.788725, with change of 0.0880085 in confidence

Decreasing data:
       Changing_On (1) is decreasing for a length of 11 Days with confidence
       0.796766, with change of 0.0600622 in confidence.

       NoisyDecrToCyclic3_On (2) is decreasing for a length of 11 Days with
       confidence 0.796766, with change of 0.0600622 in confidence.
Chaotic data:
Other data to report:
===============================================================
LEVEL 3 REPORT: All Drifters and Outliers
-----------------------------------------
Action Name              Type            Confidence   Length
PerfectCyclic3_On        Cyclic            0.959791        3
DailyConstant_On         NoDrift                  1       14
Cyclic3ToNoisyCylic7_On  Cyclic            0.959791        3
PerfectIncreasing_On     IncreasingSlope   0.636364       11
NoisyCyclic7_On          Chaotic                  1        0
NoisyIncrToNoisyDesc_On  PeriodicOutlier          1       14
NoisyIncresing_On        IncreasingSlope   0.788725        6
Changing_On              DecreasingSlope   0.796766       11
NoisyDecrToCyclic3_On    DecreasingSlope   0.796766       11
```

Figure D.6: Levels one, two and three of a daily report. This report has been reformatted to fit this document

Reports on *synthetic set two* and the initial part of the report on the *real set one* are given in figures D.7 and D.8, respectively.

```
REPORT DESCRIPTION:
        Outlier Detection using Active Lezi

NOTE:
        Criticality of the actions is kept in parenthesis.
        (1-lowest, 2-low, 3-medium, 4-high, 5-highest)

REPORTING TIME:
        November 01, 2005 at 15:00
Time                type 2       |   type 1     urgency  anomaly measure
2005-01-01T02:00:00 Motion_On               (5) 1        1 (-1.#IND)
2005-01-02T02:00:00 Cabinet_Open            (3) 0.6      1 (1.#INF)
2005-01-03T02:00:00 Cabinet_Open            (3) 0.6      1 (1.#INF)
2005-01-03T08:00:00 Computer_On             (4) 0.8      1 (1.#INF)
2005-01-03T14:00:00 Lamp_On                 (3) 0.6      1 (1.#INF)
2005-01-07T02:00:00 Lamp_On                 (3) 0.109333 0.182222 (0.182222)
2005-01-16T08:00:00 Lamp_On                 (3) 0.144738 0.241231 (0.241231)
2005-01-20T02:00:00               Motion_On (5) 0.102281 0.102281 (0.102281)
2005-01-23T14:00:00 Lamp_On                 (3) 0.156589 0.260982 (0.260982)
```

Figure D.7: Report generated for the *synthetic set two*.

```
Time                type 2|type 1    urgency ..  anomaly measure
2005-01-03T02:47:30 i5_1          (3) 0.6         1 (-1.#IND)
2005-01-03T09:56:17 i5_0          (3) 0.6         1 (1.#INF)
2005-01-03T13:04:45 a1_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:05:37 i3_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:06:11 c4_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:06:22 c4_0          (3) 0.6         1 (1.#INF)
2005-01-03T13:16:44 i3_0          (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 a2_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 a3_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 a4_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 a5_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 a6_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 a7_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 a11_1         (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 a12_1         (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 a13_1         (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 a14_1         (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 a15_1         (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 i4_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 i6_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 i7_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:18:08 i16_1         (3) 0.6         1 (1.#INF)
2005-01-03T13:18:20 i3_0          (3) 0.6         1 (1.#INF)
2005-01-03T13:18:20         i3_0 (3) 0.15        0.25 (0.25)
2005-01-03T13:25:05         c4_1 (3) 0.28        0.466667 (0.466667)
2005-01-03T13:25:08         c4_0 (3) 0.174       0.29 (0.29)
2005-01-03T13:27:33 c5_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:27:36 c5_0          (3) 0.6         1 (1.#INF)
2005-01-03T13:28:09 a1_0          (3) 0.6         1 (1.#INF)
2005-01-03T13:28:25 a8_1          (3) 0.6         1 (1.#INF)
2005-01-03T13:28:28 a8_0          (3) 0.6         1 (1.#INF)
```

Figure D.8: Report generated for the initial part of the learning period for *real set one*. This report has been reformatted to fit this document

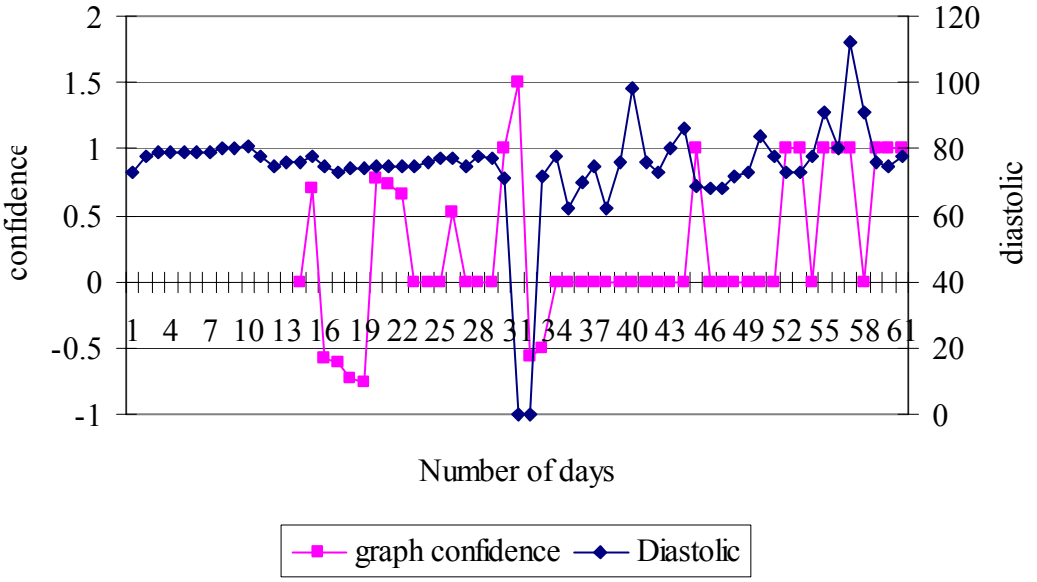


Figure D.9: Line graph for graph confidence & diastolic vs. number of days for *health set.*

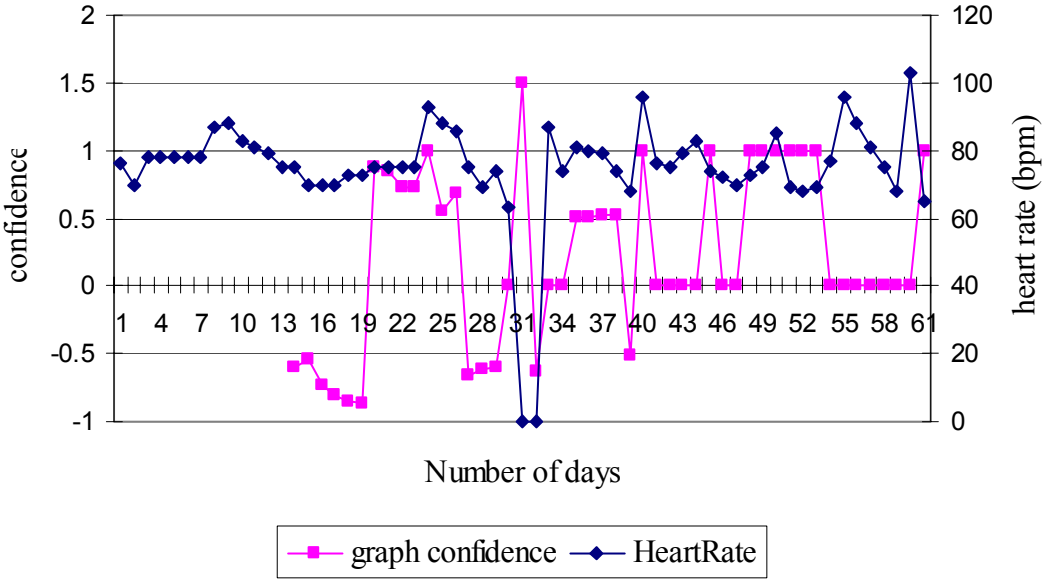


Figure D.10: Line graph for graph confidence & heart rate vs. number of days for *health set.*

Reports on *synthetic set three* and *synthetic set four* are given in figures D.11 and D.12, respectively.

```
REPORT DESCRIPTION:
       Outlier Detection using Active Lezi

NOTE:
       Criticality of the actions is kept in parenthesis.
       (1-lowest, 2-low, 3-medium, 4-high, 5-highest)

REPORTING TIME:
       November 01, 2005 at 15:00
Time                type 2       |   type 1     urgency  anomaly measure
2005-01-01T02:00:00 Motion_On               (1) 1        1 (-1.#IND)
2005-01-02T02:00:00 Cabinet_Open            (1) 0.6      1 (1.#INF)
2005-01-03T02:00:00 Cabinet_Open            (1) 0.6      1 (1.#INF)
2005-01-03T08:00:00 Computer_On             (1) 0.8      1 (1.#INF)
2005-01-03T14:00:00 Lamp_On                 (1) 0.6      1 (1.#INF)
```

Figure D.11: Report generated for the *synthetic set three*.

```
REPORT DESCRIPTION:
       Outlier Detection using Active Lezi

NOTE:
       Criticality of the actions is kept in parenthesis.
       (1-lowest, 2-low, 3-medium, 4-high, 5-highest)

REPORTING TIME:
       November 01, 2005 at 15:00
Time                type 2       |   type 1     urgency  anomaly measure
2005-01-01T02:00:00 Motion_On               (5) 1        1 (-1.#IND)
2005-01-02T02:00:00 Cabinet_Open            (5) 0.6      1 (1.#INF)
2005-01-03T02:00:00 Cabinet_Open            (5) 0.6      1 (1.#INF)
2005-01-03T08:00:00 Computer_On             (5) 0.8      1 (1.#INF)
2005-01-03T14:00:00 Lamp_On                 (5) 0.6      1 (1.#INF)
2005-01-04T02:00:00             Computer_On (5) 0.1      0.1 (0.1)
2005-01-07T02:00:00 Lamp_On                 (5) 0.109333 0.182222 (0.182222)
2005-01-11T20:00:00             Computer_On (5) 0.1025   0.1025 (0.1025)
2005-01-16T08:00:00 Lamp_On                 (5) 0.144738 0.241231 (0.241231)
2005-01-20T02:00:00               Motion_On (5) 0.102281 0.102281 (0.102281)
2005-01-23T14:00:00 Lamp_On                 (5) 0.156589 0.260982 (0.260982)
```

Figure D.12: Report generated for the *synthetic set four*.

REFERENCES

[1] Ronald M. Baecker, Jonathan Grudin, William A. S. Buxton, Saul Greenberg (1995): Readings in human-computer interaction. Toward the Year 2000. 2. ed. Morgan Kaufmann, San Francisco 1995 ISBN 1-558-60246-1

[2] M. Weiser, “The Computer of the 21st Century,” Scientific American, vol. 265, no. 3, Sept. 1991, pp. 66–75 (reprinted in this issue, see pp. 19–25).

[3] Christophe Le Gal, Jer ome Martin, Augustin Lux, and James L. Crowley. Smartoffice: Design of an intelligent environment. IEEE Intelligent Systems, 16(4):60--66, 2001. http://citeseer.ist.psu.edu/legal01smartoffice.html.

[4] M. H. Coen, "Design principals for intelligent environments," in Proceeding American Association for Artificial Intelligence 1998 Spring Symposium on Intelligent Environments, Stanford, CA, USA, Mar. 1998.

[5] Gregory D. Abowd. “Introduction: The Human Experience,” IEEE Pervasive Computing, vol. 02, no. 2, pp. 22-23, April-June, 2003.

[6] Flachsbart, J., Franklin, D. and Hammond, K. (2000). Improving Human Computer Interaction in a Classroom Environment using Computer Vision. In Proceedings of the Conference on Intelligent User Interfaces (IUI-2000).

[7] Franklin, D. (1998). Cooperating with people: The Intelligent Classroom. In Proceedings of the Fifteenth National Conference on Artifical Intelligence (AAAI-98).

[8] J. E. Bardram. “Hospitals of the Future – Ubiquitous Computing support for Medical Work in Hospitals.” In J. E. Bardram, I. Korhonen, A. Mihailidis, and D. Wan, editors, UbiHealth 2003: The 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications. http://www.pervasivehealthcare.dk/ubicomp2003, Seattle, WA, USA, Oct. 2003.

[9] Mynatt E., Essa I. and Rogers W. Increasing the Opportunities for Aging in Place. Proceedings on the2000 Conference on Universal Usability, November 2000.

[10] D. Cook and S. K. Das, Smart Environments: Technology, Protocols and Applications, John Wiley, 2005.

[11] R. Want, A. Hopper, V. Falcao, and J. Gibbons. "The Active Badge location system," ACM Transaction on Information Systems, vol. 10, no. 1, pp. 91-102, January 1992.

[12] B. Johnson, A. Fox, and T. Winograd. "The interactive workspaces project: Experiences with ubiquitous computing rooms," IEEE Pervasive Computing Magazine, vol. 1, no. 2, pp 71-78, April-June 2002.

[13] M. Coen. "Design principles for intelligent environments," AAAI Spring Symposium, Stanford, pp. 36-43, March 1998.

[14] Karthik Gopalratnam, Diane J. Cook: Active Lezi: an Incremental Parsing Algorithm for Sequential Prediction. International Journal on Artificial Intelligence Tools 13(4): 917-930, 2004

[15] Helman, P., & Bhangoo, J., A statistically base system for prioritizing information exploration under uncertainty. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 27, 449-466, 1997

[16] Lee, W., Stolfo, S. J., & Chan, P. K. (1997). Learning patterns from unix processes execution traces for intrusion detection, In Proceedings of the AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management (pp. 50{56). Menlo Park, CA: AAAI Press.

[17] Lee, W., & Stolfo, S. J., "Data mining approaches for intrusion detection," In Proceedings of the Seventh USENIX Security Symposium, 1998

[18] Edwin O. Heierman III, Diane J. Cook: Improving Home Automation by Discovering Regularly Occurring Device Usage Patterns. ICDM 2003: 537-540

[19] Kidd, Cory D., Robert J. Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth Mynatt, Thad E. Starner and Wendy Newstetter. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In Proceedings of the Second International Workshop on Cooperative Buildings . CoBuild '99.

[20] S. Lanspery, J. J. C. Jr, J. R. Miller, and J. Hyde. Introduction: Staying put. In S. Lanspery and J. Hyde, editors, Staying Put, "Adapting the Places Instead of the People," Baywood Publishing Company, pages 1-22, 1997.

[21] Lawton, M.P., "Aging and performance of home tasks," Human Factor, 32, pp. 527-536.

[22] Anthony P. Glascock, David M. Kutzik, Moving Telematics from the Laboratory to a Truly Enabling Technology within the Community, Published in Towards a Human-Friendly Assistive Environment, IOS Press, 2004

[23] Norbert Noury, "A smart sensor for the remote follow up of activity and fall detection of the elderly", 2nd annual international IEEE-EMBS Special Topic Conference on Micro-technologies in Medicine & Biology, 2002.

[24] H. Nait Charif and S. J. McKenna, "Activity Summarisation and Fall Detection in a Supportive Home Environment", IAPR International Conference on Pattern Recognition (ICPR), Cambridge, UK**,** August 2004.

[25] Budinger TF., Biomonitoring with wireless communications, Annu Rev Biomed Eng., vol. 5, 2003, 383-412.

[26] Park DC, Hertzog C, Leventhal H, Morrell RW, Leventhal E, Birchmore D, et al. Medication adherence in rheumatoid arthritis patients: older is wiser. J Am Geriatr Soc 1999; 47:172–83.

[27] Chen, D., Wactlar, H., Yang, J, A Study of Detecting Social Interaction in a Nursing Home Environment., International Conference on Computer Vision workshop on Human-Computer Interaction, In conjunction with the Tenth IEEE International Conference on Computer Vision (ICCV'05), Beijing, China, October 15-16, 2005

[28] T. Tamura, T. Togawa, and M. Murata. A bed temperature monitoring system for assessing body movement during sleep. Clinical Physics and Physiological Measurement, 9:139–145, 1988.

[29] A. Yamaguchi, M. Ogawa, T. Tamura, and T. Togawa. Monitoring behaviour in the home using positioning sensors. In Proceedings of the 20th annual IEEE conference on Engineering in Medicine and Biology, pages 1977–79, 1998.

[30] B. G. Celler, W. Earnshaw, E. D. Ilsar, L. Betbeder Matibet, M. F. Harris, R. Clark, T. Hesketh, and N. H. Lovell. "Remote monitoring of health status of the elderly at home. A multidisciplinary project on aging at the University of New SouthWales," International Journal of Bio-Medical Computing, 40:147–155, 1995.

[31] A. P. Glascock and D. M. Kutzik. Behavioral telemedicine: A new approach to the continuous nonintrusive monitoring of activities of daily living. Telemedicine Journal, 6(1):33–44, 2000.

[32] H. Inada, H. Horio, Y. Sekita, K. Isikawa, and K. Yoshida. A study on a home care support information system. In Proceedings of the Seventh World Congress on Medical Informatics, pages 349–353, 1992.

[33] M. Chan, C. Hariton, P. Ringeard, and E. Campo. Smart house automation system for the elderly and the disabled. In IEEE International Conference on Systems, Man and Cybernetics, pages 1586–1589, 1995.

[34] F. Steenkeste, H. Bocquet, M. Chan, and E. Campo. Lamise en place d'une technologie pour observer le comportement nocturne des personnes agees en institution. Innovation and Technology in Biology and Medicine –Revue of Biomedical Technology (ITBM-RBM), 22:25–30, 2001.

[35] A. J. Sixsmith. An evaluation of and intelligent home monitoring system. Journal of Telemedicine and Telecare, 6:63–72, 2000.

[36] S. J. Richardson, D. F. Poulson, and C. Nicolle. Supporting independent living through Adaptable Smart Home (ASH) technologies. In Human Welfare and Technology: Papers from the Human Service Information Technology Applications (HUSITA) 3 Conference on Information Technology and the Quality of Life and Services, pages 87–95, 1993.

[37] S. J. Richardson, D. F. Poulson, and C. Nicolle. User requirements capture for adaptable smarter home technologies. In Rehabilitation Technology: Proceedings of the 1st TIDE Congress, pages 244–248, 1993.

[38] I.L.S.A. Honeywell independent lifestyle assistant, http://www.htc.honeywell.com/projects/ilsa/.

[39] H. Kautz, L. Arnstein, G. Borriello, O. Etzioni, and D. Fox. An overview of the assisted cognition project. In Proceedings of the AAAI workshop on automation as caregiver, 2002.

[40] Oatfield Estates. Elite care, http://www.elite-care.com/

[41] M. E. Pollack, L. Brown, D. Colbry, C. E. McCarthy, C. Orosz, B. Peintner, S. Ramakrishnan, and I. Tsamardinos. Autoreminder: An intelligent cognitive orthotic system for people with memory impairment. Robotics and Autonomous Systems, 44:273{282, 2003.

[42] J. Leikas, J. Salo, and R. Poramo. Security alarm system supports independent living of demented persons. Proceedings of Gerontechnology Second International Conference, pages 402–405, 1998.

[43] Vigil Health Management http://www.vigil-inc.com.

[44] Haigh, K. Z., Phelps, J., and Geib, C. W. 2002. An open agent architecture for assisting elder independence. In Proceedings of the First international Joint Conference on Autonomous Agents and Multiagent Systems: Part 2 (Bologna, Italy, July 15 - 19, 2002). AAMAS '02. ACM Press, New York, NY, 578-586.

[45] B.-S. Lee, T. P. Martin, N. P. Clarke, B. Majeed, D. Nauck. "Dynamic Daily-Living Patterns and Association Analyses in Tele-Care Systems," icdm, pp. 447-450, Fourth IEEE International Conference on Data Mining (ICDM'04), 2004.

[46] A. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. The gator tech smart house: A programmable pervasive space. IEEE Computer, 38(3):50{60, 2005.

[47] "Generations on line," http://www. generationsonline.com.

[48] H. Maki, Y. Yonezawa, H. Ogawa, I. Ninomiya, K. Sada, and S. Hamada. "A microcomputer-based life-safety monitoring system for elderly people," In Proceedings of the International IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology, pages 3970-3972, 2001.

[49] W. Dewing, S. Metz, and W. Winchester. Does home-care automation for elders change the caregiver experience? Gerontechnology, 2(1), 111, 2002.

[50] N. Noury, T. Herve, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, and T. Porcheron, "Monitoring behavior in home using a smart fall sensor," In Proceedings of the International IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology, pages 607-610, 2000

[51] A. Sixsmith and N. Johnson. SIMBAD: "smart inactivity monitor using array-based detector." Gerontechnology, 2(1):110, 2002.

[52] N. Atsushi, K. Hirokazu, H. Shinsaku, and I. Seiji. Tracking multiple people using distributed vision systems. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 2974-2981, 2002.

[53] D. Focken and R. Stiefelhagen. Towards vision-based 3-d people tracking in a smart room. In Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces, 2002.

[54] B. Najafi, K. Aminian, A. Paraschiv-Ionescu, F. Loew, C. Bula, and P. Robert. Ambulatory system for human motion analysis using a kinematic sensor: Monitoring of daily physical activity in the elderly. IEEE Transactions on Biomedical Engineering, 50(6), 711-723, 2003.

[55] M. Ogawa, R. Suzuki, S. Otake, T. Izutsu, T. Iwaya, and T. Togawa. Long term remote behavioral monitoring of elderly by using sensors installed in ordinary houses. In Proceedings of the International IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology, pages 322{325, 2002.

[56] T. Togawa. Home health monitoring. Journal of Medical and Dental Sciences, 45(3):151-160, 1998.

[57] Barger, T.; Alwan, M.; Kell, S.; Turner, B.; Wood, S.; and Naidu, A. 2002. Objective remote assessment of activities of daily living: Analysis of meal preparation patterns. Poster presentation,
http: //marc.med.virginia.edu/pdfs/library/ADL.pdf.

[58] Mihailidis, A.; Fernie, G. R.; and Barbenel, J. 2001. The use of artificial intelligence in the design of an intelligent cognitive orthosis for people with dementia. Assistive Technology 13:23–39.

[59] Liao, L.; Fox, D.; and Kautz, H. 2004. Learning and inferring transportation routines. In Proceedings of the 19th National Conference on Artificial Intelligence, 348–353.

[60] Philipose, M.; Fishkin, K. P.; Perkowitz, M.; Patterson, D. J.; Hahnel, D.; Fox, D.; and Kautz, H. 2005. Inferring ADLs from interactions with objects. IEEE Pervasive Computing. In press.

[61] Ghosh, A., & Schwartzbard A, A study in using neural networks for anomaly and misuse detection. Proceedings of the Eighth USENIX Security Symposium, 1999

[62] Lane, T., & Brodley, C. E. . Sequence matching and learning in anomaly detection for computer security. Proceedings of the AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management , pp. 43-49, Menlo Park, CA: AAAI Press, 1997

[63] Lane, T., & Brodley, C. E. Temporal sequence learning and data reduction for anomaly detection. In Proceedings of the Fifth ACM Conference on Computer and Communications Security, pp. 150-158, 1998

[64] Lane, T., & Brodley, C. E. (1999). Temporal sequence learning and data reduction for anomaly detection. ACM Transactions on Information and System Security, 2, 295-331.

[65] Barnett, V., & Lewis, T. (1994). Outliers in statistical data. New York: John Wiley and Sons.

[66] J. Schlimmer and R.H. Granger, "Incremental learning from noisy data," Machine Learning, 1(3), pp. 317–354, 1986.

[67] D.W. Aha, D. Kibler, and M.K. Albert, "Instance-based learning algorithms," Machine Learning, 6, pp. 37–66, 1991.

[68] G. Widmer and M. Kubat, "Effective learning in dynamic environments by explicit concept tracking," Proc. of the Sixth European Conference on Machine Learning, pp. 227–243, 1993.

[69] G. Widmer, "Tracking context changes through meta-learning," Machine Learning, 27(3), pp. 259–286.

[70] M.B. Harries, C. Sammut, and K. Horn, "Extracting hidden context," Machine Learning, 32, pp. 101–126, 1998.

[71] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," Proc. of the 7th ACM SIGKDD Int.Conference on Knowledge Discovery and Data Mining, pp. 97–106, 2001.

[72] H. Wang, W. Fan, P.S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," Proc. of the 9th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining, pp. 226–235, 2003.

[73] P. Cunningham, N. Nowlan, S.J. Delany, M. Haahr, "A case-based approach to spam filtering that can track concept drift," ICCBR'03 Workshop on Long-Lived CBR Systems, Trondheim, Norway, Jun. 2003.

[74] Jin, W., Tung, A. K., and Han, J. 2001. Mining top-n local outliers in large databases. In Proceedings of the Seventh ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (San Francisco, California, August 26 - 29, 2001). KDD '01. ACM Press, New York, NY, 293-298.

[75] E. Heierman, M. Youngblood, and D. J. Cook, Mining Temporal Sequences to Discover Interesting Patterns, KDD Workshop on Mining Temporal and Sequential Data, 2004

[76] S. Helal, B. Winkler, C. Lee, Y. Kaddoura, L. Ran, C. Giraldo, S. Kuchibhotla, and W. Mann. Enabling location-aware pervasive computing applications for the elderly. In First IEEE International Conference on Pervasive Computing and Communications, page 531, 2003.

[77] G. Abowd, C. Atkeson, A. Bobick, I. Essa, B. Mynatt, and T. Starner. Living laboratories: The Future Computing Environments group at the Georgia Institute of Technology. In Proceedings of the 2000 Conference on Human Factors in Computing Systems (CHI), 2000.

[78] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. IEEE Computer, 34(8):50–56, August 2001.

[79] M. Bennewitz, W. Burgard, and S. Thrun. Learning motion patterns of persons for mobile service robots. In Proceedings of ICRA, 2002.

[80] B. Clarkson, N. Sawhney, and A. Pentland. Auditory context awareness via wearable computing. In Proceedings of the Perceptual User Interfaces Workshop, 1998.

[81] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson. Advances in cooperative multi-sensor video surveillance. Proceedings of DARPA Image Understanding Workshop, 1:3–24, 1998.

[82] M. Mozer. The neural network house: An environment that adapts to its inhabitants. In Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments, pages 110–114, 1998.

[83] S. Lee and K. Mase. Activity and location recognition using wearable sensors. In First IEEE International Conference on Pervasive Computing and Communications, pages 24–32, 2002.

[84] D. Meyer. Human gait classification based on hidden Markov models. In 3D Image Analysis and Synthesis, pages 139–146, 1997.

[85] T. Starner. Visual recognition of American sign language using hidden Markov models. Master's thesis, MIT Media Lab, 1994.

[86] Gregory D. Abowd, Agathe Battestini, and Thomas O'Connell. The Location Service: A Framework for Handling Multiple Location Sensing Technologies, 2002. Website: www.cc.gatech.edu/fce/ahri/publications/location service.pdf.

[87] Microsoft Research Vision Group. Easy Living, Oct 2003. Website: research.microsoft.com/easyliving.

[88] M. Coen. "Design principles for intelligent environments," AAAI Spring Symposium, Stanford, pp. 36-43, March 1998.

[89] Ambient Intelligence research and Development Consortium. Ambient Intelligence for the Networked Home Environment, 2005. Website: www.amigo-project.org.

[90] Stanford Interactivity Lab. Interactive Workspaces, Oct 2003. Website: iwork.stanford.edu.

[91] UIUC Software Research Group. Gaia homepage, 2005. Website: gaia.cs.uiuc.edu.

[92] D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. Machine Learning, 34(1-3):177–210, 1999.

[93] IIEG. Welcome to IIEG, 2005. Website: cswww.essex.ac.uk/intelligent-buildings.

[94] D. J. Cook and S. Das, MavHome: Work in Progress IEEE Pervasive Computing, 2004.

[95] Michael Youngblood. "MavPad Inhabitant 2 Trial 2 Data Set." The MavHome Project. Computer Science and Engineering Department. The University of Texas at Arlington. 2005. <http://mavhome.uta.edu>.

[96] Michael Youngblood, Diane J. Cook, and Lawrence B. Holder. "Managing Adaptive Versatile Environments." Pervasive and Mobile Computing 20 (2005).

[97] Michael Youngblood. Automating Inhabitant Interactions in Home and Workplace Environments through Data-driven Generation of Hierarchical Partially-observable Markov Decision Processes. Doctoral Dissertation. The University of Texas at Arlington. August 2005.

[98] Chebyshev , "http://en.wikipedia.org/wiki/Chebyshev's_theorem", 2003

[99] Frederick Coolidge, "Statistics: A Gentle Introduction", pp 61-63, 2000

[100] Ziv, J. and Lempel, A. Compression of Individual Sequences via Variable Rate Coding. IEEE Transactions on Information Theory, IT-24:530-536, 1978.

BIOGRAPHICAL INFORMATION

Gaurav Jain was born in Dehradun (in Uttaranchal), India. He lived in New Delhi all his life before moving to the United States of America in 2003. When in India he got a degree in Bachelors in Engineering in Computer Science and Engineering from Institute of Technology and Management, Haryana in 2003. In the US he continued his studies in pursuit of a Masters of Science in Computer Science and Engineering from The University of Texas at Arlington. He received his Master of Computer Science in 2005 from The University of Texas at Arlington. During his Masters degree he worked as an symbian programmer for a project at The University of Texas at Arlington for a period of one year. His academic interests include artificial intelligence, data mining and embedded systems. Some of his previous work and major projects include: An approach to text classification using dimensionality reduction and combination of classifiers; a PDA-based reminder system for people with disabilities; and a study of cryptography techniques.