Smartphone Based Activity Recognition System

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Sen Zhang

Graduate Program in Computer Science and Engineering

The Ohio State University

2012

Master's Examination Committee:

Professor Dong Xuan, Advisor

Professor Feng Qin
Abstract

Nowadays, obesity and overweight are epidemic worldwide. Obese people are at risk for diseases such as heart disease, type 2 diabetes, stroke, and certain types of cancer. Finding out an effective treatment for obesity becomes a major challenge for researchers. Physical activity plays a vital role in treating obesity, as the energy consumed during such activity helps “burn off” excess body fat. With the rapid development of smartphone technology, we realized that prolific mobile phones can potentially monitor physical activity thanks to their motion sensors, e.g., accelerometers, orientation sensors, and gyroscopes. In this paper, we design and implement an activity recognition system using mobile phones. Unlike prior work, our system uses acceleration, orientation, and gyroscope data for on-line activity recognition. Our system samples phone sensor data at 5 Hz and uploads them to a central server for processing. Our preliminary experiments indicate our system’s potential for activity recognition.
Dedication

This document is dedicated to my parents
Acknowledgments

I would like to thank my advisor Dr. Dong Xuan for the guidance and the intellectual support he has provided me throughout my M.S. study. I am also thankful for all the inputs and efforts he put for my thesis. I would like to thank for Dr. Feng Qin for agreeing to be on my Master’s Examination Committee and for all the valuable suggestions he has made.

I am very thankful to Adam Champion, Qiang Zhai, Xinfeng Li, Jing Teng, Gang Li and Boying Zhang for being great mentors and close friends, and for all the guidance and discussions we had in our cubicle.

I would like to especially thank Zhe Yuan for helping me with adapting the new environment and for providing me with a lot of study, work and life experience. I would also like to thank everyone who made my stay at the Ohio State memorable.

I would like to express my deepest gratitude to my parents who have shown unconditional love and support at every step in my life. I am especially thankful to my mother for her moral support and for making some of my harder times a lot easier.
Vita

July 27, 1987 .................................................. Born – Shenyang, China

2010 .............................................................. B.S. Information Security, Northeastern University

2011 to present .............................................. Graduate Research Assistant, Department of Computer Science and Engineering, The Ohio State University

Fields of Study

Major Field: Computer Science and Engineering

Studies in:

Computer Networking .................................. Professor Dong Xuan
# Table of Contents

Abstract ........................................................................................................................................ ii

Dedication .................................................................................................................................. iii

Acknowledgments ................................................................................................................ iv

Table of Contents ................................................................................................................ vi

List of Tables ........................................................................................................................ ix

List of Figures ......................................................................................................................... x

CHAPTER 1 .................................................................................................................................. 1

1.1 Problem Overview and Motivation ................................................................................ 1

1.2 Contributions ................................................................................................................... 3

1.3 Outline ............................................................................................................................... 4

CHAPTER 2 ................................................................................................................................ 5

CHAPTER 3 ............................................................................................................................... 11

3.1 Mobile Phone Client ..................................................................................................... 12

3.1.1 Data Collection Module .......................................................................................... 13

3.1.2 Data Communication Module ................................................................................ 15

3.2 Server Design ................................................................................................................ 16
5.2.7 Using Separate Test Set ................................................................. 45

CHAPTER 6 ......................................................................................................... 48

6.1 Conclusion .............................................................................................. 48

6.2 Future Work .......................................................................................... 49

BIBLIOGRAPHY ............................................................................................. 51
List of Tables

Table 1. Average CPU and Memory Usage of the Application on the Mobile Phone .... 30
Table 2. Accuracies of Activity Recognition................................................................. 33
Table 3. Accuracies of Activity Recognition of Another Work[6] ............................... 34
Table 4. Accuracies of Activity Recognition with Sampling Rate of 10Hz................. 36
Table 5. Accuracies of Activity Recognition with Sampling Rate of 20Hz............... 36
Table 6. Accuracies of Activity Recognition with Different Phone Client ............... 38
Table 7. Accuracies of Activity Recognition with One Accelerometer Alone.............. 39
Table 8. Accuracies of Activity Recognition with Accelerometer and Rotation Sensor.. 41
Table 9. Accuracies of Activity Recognition with Accelerometer and Gyroscope Sensor .................................................................................................................. 42
Table 10. Accuracies of Activity Recognition with Fast Speed Running ..................... 44
Table 11. Accuracies of Activity Recognition with Different Test Set .......................... 46
List of Figures

Figure 1. Activity Recognition System Architecture ....................................................... 11
Figure 2. Mobile Phone Client Structure ......................................................................... 12
Figure 3. Coordinate System Used by Sensors .................................................................. 13
Figure 4. Coordinate of the Earth ..................................................................................... 14
Figure 5. Rotation Direction ............................................................................................. 14
Figure 6. Server Workflow .............................................................................................. 17
Figure 7. Server-Database Architecture ........................................................................... 18
Figure 8. DBMS Handler Design ..................................................................................... 19
Figure 9. Data Model ....................................................................................................... 21
Figure 10. Daily Data Generation by Subjects during One Week Experiment ............... 31
Figure 11. Smartphone Battery Life for Subjects during Experiment ............................ 32
CHAPTER 1

INTRODUCTION

1.1 Problem Overview and Motivation

Obesity and overweight are a worldwide epidemic. The World Health Organization (WHO) estimates that 500 million people worldwide were obese (i.e., excessively overweight) in 2008, almost twice as many as in 1980 [1]. Obesity and overweight are prominent in the Americas, including the United States. The Centers for Disease Control and Prevention (CDC) estimate that, in 2010, over 78 million United States adults and about 12.5 million United States children and adolescents were obese [2]. Obesity is associated with diseases such as heart disease, type 2 diabetes mellitus, and certain cancers as well as shortened lifespan. As Barness et al. point out obesity poses “one of the most serious public health concerns in the 21st century” [3]. Not only overweight people but also healthy people have already been aware of the seriousness of obesity.

Obesity and overweight are defined in terms of the body mass index (BMI), a measure specified as follows:

\[
\text{BMI} = \frac{\text{weight (kg)}}{\left(\text{height (m)}\right)^2}
\]
Healthy, overweight, and obese people’s BMIs range from 18.5–24.9 kg/m², 25.0–29.9 kg/m², and at least 30 kg/m², respectively [2], [4]. This report follows such convention.

The U.S. health care system is under severe financial stress. As the first wave of baby boomers reaches retirement age by 2010, the situation is expected to deteriorate rapidly [32]. One partial solution to this problem involves the development of systems that shift the emphasis from obesity treatment in hospitals to health promotion and quality of life conservation at home. Keeping people at home and out of the hospital reduces the financial burden on the system. According to gerontologists, identifying changes in everyday behavior is often more valuable than biometric information for the treatment of obesity and overweight problems. A fine grain recognition of activities of daily living will be essential to implement many of the proposed strategies for encouraging healthy behavior related to diet, exercise, and medication adherence. Unfortunately, health researchers currently do not have the means to collect the necessary sensor data to detect activities and patterns of behavior in actual homes. Researchers do not know if it is possible to recognize human activities using a set of simple, easy to install ubiquitous sensors nor do they understand what modifications are necessary in conventional pattern recognition algorithms to achieve this.

Mobile phones may potentially help encourage obese people to exercise and record their physical activity. These devices are highly pervasive in society, with over 480 million sold in 2011 alone [5]. Mobile phones attend their owners everywhere they go and are equipped with powerful motion sensing capabilities. In particular, phone sensors
such as accelerometers, gyroscopes, and orientation sensors can detect linear acceleration, angular velocity, and angular position, respectively. Different sensor readings can indicate different kinds of activity. For examples, unchanging accelerations over time can indicate sedentary activity such as sitting, whereas sharply varying vertical accelerations can indicate running or climbing stairs. Since mobile phones equipped with multiply motions sensors can potentially recognize people’s activity, they may become a more powerful for users to monitor their health conditions. Obese people have to do a certain amount of exercise every day to lose their weight. Mobile phones become the best tools for them to record their daily activities, because people usually carry their phones every day and everywhere. After a whole day’s working, people could check out their daily activity record on the phone, and they will have a clear picture of their physical activity performance during the day. Then they can adjust their later exercise plan based on the physical activity record to achieve the best exercise and losing weight effect.

1.2 Contributions

This work describes a smartphone based activity recognition system. We collect sensor data from the mobile phones, then upload them to a central server, and analyze them to determine the phone user’s current activities.

There has been some related research work using mobile phones [6]–[9] (and references therein). In general, these works gather acceleration data using mobile phone accelerometers and/or ancillary sensors [9], and then upload them to a central server for processing. They gather only accelerometer data and perform off-line activity recognition,
i.e., activities are recognized long after sensor data collection. In contrast, our system aims to exploit the gamut of motion sensors available in mobile phones for performing near real-time activity recognition. For instance, gyroscopes are common in high-end mobile phones and provide more insight into phone motion than just linear acceleration, so we should incorporate their data into activity recognition.

We implement our system on commercial off-the-shelf smartphones and a GNU/Linux server. Preliminary experimental results show that our system can distinguish different activities with a high accuracy.

1.3 Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces some related works. Chapter 3 describes system design in details. Chapter 4 discusses the implementation of the system. Chapter 5 provides some evaluation of the system with different cases. Conclusion and future work are discussed in Chapter 6.
CHAPTER 2

RELATED WORKS

More and more attention has been paid to activity recognition in the field of mobile communication because of the increasing availability of accelerometers in consumer products, like smartphones, and because of its many potential applications. Some of the earliest work in accelerometer-based activity recognition focused on the use of multiple accelerometers placed on several parts of the user’s body. In one of the earliest studies of this topic, Bao & Intille [10] used five biaxial accelerometers worn on the user’s right hip, dominant wrist, non-dominant upper arm, dominant ankle, and non-dominant thigh in order to collect data from 20 users. Using decision tables, instance-based learning, C4.5 and Naïve Bayes classifiers, they created models to recognize twenty daily activities. Their results indicated that the accelerometer placed on the thigh was most powerful for distinguishing between activities. This finding supports our decision to have our test subjects carry the phone in the most convenient location—the pockets of their pants.

Other researchers (e.g., Bao & Intille (2008)) have used multiple accelerometers for activity recognition. Krishnan et al.[11] collected data from three users using two accelerometers to recognize five activities—walking, sitting, standing, running, and lying down. The authors of this paper claimed that data from a thigh accelerometer was
insufficient for classifying activities such as sitting, lying down, walking, and running, and thus multiple accelerometers were necessary for collecting data from different part of human's body (a claim our research contradicts).

In another study, Krishnan et al. [12] examined seven lower body activities using data collected from ten subjects wearing three accelerometers. This method was tested in supervised and semi-naturalistic settings. Tapia et al. [13] collected data from five accelerometers placed on various body locations for twenty-one users and used this data to implement a real-time system to recognize thirty gymnasium activities. A slight increase in performance was made by incorporating data from a heart monitor in addition to the accelerometer data. Mannini and Sabitini [14] used five triaxial accelerometers attached to the hip, wrist, arm, ankle, and thigh in order to recognize twenty activities from thirteen users. Various learning methods were used to recognize three “postures” (lying, sitting, and standing) and five “movements” (walking, stair climbing, running, and cycling). Foerster and Fahrenberg [15] used data from five accelerometers in one set of experiments and from two of those accelerometers in another for activity recognition. Thirty one male subjects participated in the study and a hierarchical classification model was built in order to distinguish between postures such as sitting and lying at specific angles, and motions such as walking and climbing stairs at different speeds.

Researchers have used a combination of accelerometers and other sensors to achieve activity recognition. Parkka et al. [16] created a system using twenty different types of sensors (including an accelerometer worn on the chest and one worn on the wrist) in order to recognize activities such as lying, standing, walking, running,
football, swinging, croquet, playing ball, and using the toilet in specific locations. Lee and Mase [17] created a system to recognize a user’s location and activities, including sitting, standing, walking on level ground, walking upstairs, and walking downstairs using a sensor module that consisted of a biaxial accelerometer and an angular velocity sensor worn in the pocket combined with a digital compass worn at the user’s waist. Subramayana et. al. [18] addressed similar activities by building a model using data from a tri-axial accelerometer, two micro-phones, phototransistors, temperature and barometric pressure sensors, and GPS to distinguish between a stationary state, walking, jogging, driving a vehicle, and climbing up and down stairs.

While these systems using multiple accelerometers or a combination of accelerometers and other sensors were capable of identifying a wide range of activities, they are not very practical because they involve the user wearing multiple sensors distributed across their body. This could work for some short term, small scale, highly specialized applications (e.g., in a hospital setting) but would certainly not work for the applications that we envision.

Some studies have also focused on combining multiple types of sensors in addition to accelerometers for activity recognition. Maurer et al. [19] used “eWatch” devices placed on the belt, shirt pocket, trouser pocket, backpack, and neck to recognize the same six activities that we consider in our study. Each “eWatch” consisted of a biaxial accelerometer and a light sensor. Decision trees, k-Nearest Neighbor, Naïve Bayes, and Bayes Net classifiers with five-fold cross validation were used for learning. Choudhury et. al [20] used a multimodal sensor device consisting of seven different types of
sensors (tri-axial accelerometer, microphone, visible light photo transistor, barometer, visible+IR light sensor, humidity/temperature reader, and compass) to recognize activities such as walking, sitting, standing, ascending stairs, descending stairs, elevator moving up and down, and brushing one’s teeth. Cho et. al. [21] used a single tri-axial accelerometer, along with an embedded image sensor worn at the user’s waist, to identify nine activities. Although these multi-sensor approaches do indicate the great potential of mobile sensor data as more types of sensors are being incorporated into devices, our approach shows that only one type of sensor—an accelerometer—is needed to recognize most daily activities. Thus our method offers a straightforward and easily-implementable approach to accomplish this task.

Other studies, like our own, have focused on the use of a single accelerometer for activity recognition. Long, Yin, and Aarts [22] collected accelerometer data from twenty-four users using a triaxial accelerometer worn without regard for orientation at the user’s waist. Data was collected naturalistically, and decision trees as well as a Bayes classifier combined with a Parzen window estimator were used to recognize walking, jogging, running, cycling, and sports. Lee et. al. [23] used a single accelerometer attached to the left waists of five users. Standing, sitting, walking, lying, and running were all recognized with high accuracies using fuzzy c-means classification. However unlike these studies, which use devices specifically made for research purposes, our method utilizes commercial devices that are widely-available without any additional specialized equipment. This approach enables make practical real-world applications for our models.
Several researchers have considered the use of widely-available mobile devices such as cell phones to address the activity recognition problem. However, the earlier approaches did not take advantage of the sensors incorporated into the mobile devices themselves. For example, György et al. [24] used “MotionBands” attached to the dominant wrist, hip, and ankle of each subject to distinguish between six different motion patterns. Each MotionBand contained a tri-axial accelerometer, magnetometer, and gyroscope. As the MotionBand collected data, the data was then transmitted to a smart phone carried by the user to be stored. Ravi et al. [25] collected data from two users wearing a single accelerometer-based device and then transmitted this data to the HP iPAQ mobile device carried by the user. Using this data for activity recognition, researchers compared the performance of eighteen different classifiers. Lester et al. [26] used accelerometer data, along with audio and barometric sensor data, to recognize eight daily activities from a small set of users. While these studies could have used a cell phone to generate the accelerometer data, they did not do this. Instead, the data was generated using distinct accelerometer-based devices worn by the user and then sent to a cell phone for storage.

A few studies, like ours, did use an actual commercial mobile device to collect data for activity recognition. Such systems offer an advantage over other accelerometer-based systems because they are unobtrusive and do not require any additional equipment for data collection and accurate recognition. Miluzzo et al. [27] explored the use of various sensors (such as a microphone, accelerometer, GPS, and camera) available on commercial smart phones for activity recognition and mobile social networking.
applications. In order to address the activity recognition task, they collected accelerometer data from ten users to build an activity recognition model for walking, running, sitting, and standing using J48. This model had particular difficulty in distinguishing between the sitting and standing activities, a task that our models easily achieve. Yang [28] developed an activity recognition system using the Nokia N95 phone to distinguish between sitting, standing, walking, running, driving, and bicycling. This work also explored the use of an activity recognition model to construct physical activity diaries for the users. Although the study achieved relatively high accuracies of prediction, stair climbing was not considered and the system was trained and tested using data from only four users. Brezmes et. al. [29] also used the Nokia N95 phone to develop a real-time system for recognizing six user activities. In their system, an activity recognition model is trained for each user, meaning that there is no universal model that can be applied to new users, for whom no training data exists. Our models do not have this limitation.
CHAPTER 3
SYSTEM DESIGN

In this chapter, we present the design details of the system. Our system is an on-line real-time activity recognition system, so we use the typical client and server model. As the architecture shown in Figure 1, the system has three main components: the mobile phone client, central server, and database. We describe each of these in turn.

Figure 1. Activity Recognition System Architecture
3.1 Mobile Phone Client

The phone’s functions are collecting sensor data and uploading them to the server. Therefore, the application running on the phone can be divided into two modules: the data collection module and the data communication module. The data collection module is in charge of data collection from sensors and the data communication module is used to transfer collected data to the server via WiFi. Figure 2 illustrates the structure of the mobile phone client.

![Figure 2. Mobile Phone Client Structure](image-url)
3.1.1 Data Collection Module

The Data Collection Module is in charge of collecting data from three major sensors: the accelerometer, the rotation vector sensor, and the gyroscope. The accelerometer measures (linear) acceleration along the x-, y- and z-axes of the phone. Figure 3 shows the directions of these three axes. The rotation vector represents the orientation of the device as a combination of an angle and an axis, where the device is rotated by an angle $\theta$ around an axis ($x; y; z$). The three elements of the rotation vector are $(x \sin(\theta/2); y \sin(\theta/2); z \sin(\theta/2))$ such that the magnitude and direction of the rotation vector are equal to $\sin(\theta/2)$ and the direction of the axis of rotation, respectively. It describes the difference between the coordinate of the phone and coordinate of the earth which represents the relative position of the phone to the earth. Figure 4 shows the coordinate of the earth. The gyroscope sensor measures the rate of rotation around the device’s x-, y- and z-axes. The coordinate system is the same as that used for the acceleration sensor. Rotation is positive in the counterclockwise direction as shown in Figure 5.

![Coordinate System Used by Sensors](image)
We collect sensor data with a sampling rate of 5 Hz. This rate can not only guarantee the accuracy of activity recognition, but also limits the amount of collected data to a reasonable volume. If we choose a higher data collection rate, such as 20 Hz, a huge amount of data will be uploaded to the database when the application runs for a long time, and the server will have a considerable burden when a large number of users are connected to it. We believe 5 Hz is the best choice that balances recognition accuracy.
with the volume of sensing data. The data collection module passes a block of data to the
data communication module every 5 seconds. 5 seconds is a reasonable transfer interval,
because the amount of collected data that needs to be uploaded to the server is not very
large and will not cause too much transmission latency.

3.1.2 Data Communication Module

The data communication module is used to transfer collected sensor data to the
server for activity recognition. Before the data collection module starts monitoring and
collecting sensing data, the data communication module needs to establish a connection
with the server via WiFi. If it cannot connect to the server, the application will abort
because if it cannot upload the collected data to the server, data collection is futile. We
use a Java Socket to establish the connection and the channel to transfer sensing data.
Once the data communication module receives a block of sensing data collected by the
data collection module, it will upload them to the server immediately. As we mentioned
above, the data communication module uploads the data every 5 seconds along with the
raw data collected from sensors, a timestamp, and the phone’s IMEI number. We can see
that the IMEI number is used to distinguish different users and timestamps are used to
calculate the data collection intervals.

The following is a data block example to describe the format of the information to
be transferred:

Data Format:
IMEI 123456786543210
Time-stamp 1338688788083
Acceleration 25 3 // followed by 25*3 matrix
[ ][ ][ ]
3.2 Server Design

The server is a bridge between the client and the activity recognizer engine. It accepts connections from all clients, parses their input data, inserts them into the DBMS, and invokes the engine. The server has three main components: the communication component, the DBMS handler, and the engine caller. The communication component serves as the “customs” of the server system. It receives the connections outside and parses the input data. The DBMS handler receives the parsed data and inserts them into the database. It is an abstract DBMS tool for the server. The engine caller is a lightweight component that invokes the real activity recognizer engine and reports its result. Figures 4 and 5 show the workflow of the entire server system and the server’s interaction with the database, respectively. The following subsections will describe the design of each component.
3.2.1 Communication Component

Figure 5 illustrates the communication component’s functionality. There is a ServerSocket listening to a particular port for incoming connections. This ServerSocket has an exclusive thread to guarantee the server’s quick response to the connection request from any clients. After a connection from the client to the server is established, the exclusive thread will create another thread to handle the communication to the client, i.e. for each connection, there is an independent thread for it to handle the communication. These threads will first parse the incoming data from the client. The uploaded data of the client is marked by several labels such as IMEI, TIMESTAMP, and soon. Therefore, the server can easily parse the incoming data by labels. After parsing, it will first query the phone ID registered in the database to make sure the client is registered. Finally, after the thread finishes the above tasks, which means that the client of the connection is registered
and the data it uploaded is qualified, it will hand over the parsed data to the DBMS handler to update the database.

![Server-Database Architecture Diagram]

Figure 7. Server-Database Architecture

### 3.2.2 DBMS Handler

There is only one DBMS handler for the whole server, as shown in figure 6. This design is to reduce the negative effects of multiple concurrent threads since they may
need to work with the database simultaneously. The DBMS handler is the only tool for the server to insert or retrieve the data in the database. It also handles potential errors during the operation of DBMS. There are several techniques to support basic database operations such as those provided by JDBC in Java.

![DBMS Handler Design](image)

**Figure 8.** DBMS Handler Design

### 3.2.3 Engine Caller

The engine caller is a lightweight component whose input is various tasks. It is designed to overcome the heterogeneity of the activity recognition engine. The
heterogeneity comes from many aspects. One of them is the different working environment. Some engines are implemented in Java and run via the Java virtual machine while others may be written in C/C++ and need to be compiled for different runtime environments. Engine invocation can also vary since these tasks operate in different environment like Python or directly with the operating system. Therefore, to provide the general abstract model of an “activity recognition engine,” the engine caller defines several interfaces so that the engine invocation can be implemented independently with respect to the others. Therefore, it is easy to expand since each engine only needs to implement its own invocation interface.

3.3 Data Model and Data Mining

Figure 7 shows our database design. We use four main tables: PHONE, ACTIVITY_TYPE, ACTIVITY, and SENSOR_READING. PHONE stores any or all of the following identifiers (IDs) for a mobile phone client: Bluetooth MAC address, WiFi MAC address, IMEI, IMSI, and phone number. In practice, we use the IMEI as a phone client ID. ACTIVITY_TYPE stores the following activity types: sitting, standing, walking, climbing upstairs, climbing downstairs, and “unspecified.” SENSOR_READING stores accelerometer, orientation, and gyroscope readings from a mobile phone client. Optionally, a client can include its geolocation and IMEI. Each row in this table is marked as training data or test data, which is discussed shortly. The ACTIVITY relation table maps a particular activity type to a phone and contiguous block of sensor readings.
We preprocess the data and classify each sensor reading as an activity type. We implement these functions in Python using Orange [16]. We have three types of sensor readings: (1) marked training data that corresponds to a known activity type and is referenced in the ACTIVITY table, (2) other data that the system has previously classified as an activity type; and (3) all other data. We use type (1) sensor readings as
training data and type (3) sensor readings as test data. In our implementation, we use a
decision tree classifier due to its accuracy as reported in [6].
CHAPTER 4

IMPLEMENTATION

This chapter presents the implementation of the system based on the architecture introduced in Chapter 3. As illustrated in Figure 1, the system consists of a mobile phone client, a central server, and a database. The application installed on the mobile phone client includes: 1) Data Collector, which is responsible for reading data from accelerometer sensors and orientation sensors; 2) Communication Module, which is in charge of uploading data and receiving feedback. The backstage server is composed of an http server, a database for storing collected sensor data for each user and a data analysis server for analyzing the unprocessed sensor data.

4.1 Mobile Phone Client Implementation

For the client, we develop the application for data collection and data transmission on Android Nexus S phone. The Nexus S has the Samsung Exynos 3110 processor. This processor combines a 45 nm 1 GHz ARM Cortex A8 based CPU core with a PowerVR SGX 540 GPU. The CPU core, code-named "Hummingbird", was co-developed by Samsung and Intrinsity. The GPU, designed by Imagination Technologies, supports OpenGL ES 1.1/2.0 and is capable of processing up to 20 million triangles per second.
The Nexus S has 512 MB of dedicated RAM (Mobile DDR) and 16 GB of NAND memory, partitioned as 1 GB internal storage and 15 GB microSD. It also provides an embedded accelerometer and an embedded orientation sensor. In the following, we describe the implementation details of the application.

We implement the application in Java, with Eclipse and Android 1.6 SDK. The application can be divided into three major components: user interface, monitoring daemon and communication module. After the user starts the application, the application automatically connects to the server automatically by using Java socket. If the connection fails, users can try again or abort the application. Once the application successfully connects to the server, the monitoring daemon will start to keep monitoring the data read from the accelerometer sensor, the rotation sensor and the gyroscope sensor. It will display the values on the screen and update the values every 0.2 second. This function is implemented by using a SensorManager instance. In the first place, the accelerometer sensor, the rotation sensor and the gyroscope sensor need to be registered in the SensorManager. Then the SensorManager should be assigned with a monitoring frequency which is 5 Hz. Now the SensorManager can start to read the sensor’s data every 0.2 second. The communication module starts working at the same time. It will upload sensor data to the server every 5 seconds through the socket that has been established. The reason why sensor data is uploaded to the server every 5 second is the mobile phone is not supposed to store a large amount of data in the memory because of the memory’s limited capacity. If the user wants to stop monitoring, he can press a button at the bottom of screen to finish. Then the SensorManager stops monitoring the registered
sensors and unregister them. Meanwhile, the application will send a finish instruction to the server to ask for the analyzing result. After that, the application will jump to another user interface to display the feedback and the communication module will start to receive feedback from the server. Once the application receives the result, it displayed it on the screen.

4.2 Database Implementation

As we discussed above, the server is composed of a database, and a data analysis center. All the data collected from sensors are stored in MySQL database. The database contains four tables: activity type table, phone table, activity table and user record table. The activity type table includes six numbers corresponding to six activities stored in activity table. The structure was chosen because it saves a large amount of storage space by storing the numbers instead of the long string in the database. We distinguish different users based on the smartphone from which the server receives the data, so we have a smartphone table. The basic idea to identify a certain cellphone is to use the IMEI number of the cellphone, because the IMEI number of the cellphone is globally unique. However, IMEI number is kind of personal information, so we decide to store the hash code of IMEI numbers for the security issues. In the user data table, we not only store the raw sensor data received from users and analyzing results for each block of data, but also include users' personal information such as height, weight and age. The personal information is very useful for generating some suggestions for users based on their data analyzing results.
4.3 Data Analysis Server Implementation

4.3.1 Data Analysis Procedure

When the server receives an amount of data, it saves it to the database based on the hash code of the smartphone’s IMEI and triggers the data analysis center with the copy of the data. The data analysis center will keep working until it receives the finish instruction from the client. The analyzing unit is one second, which means the data analysis center need to figure out the activity the user performed within that certain second based on 5 samples from the accelerometer sensor, 5 samples from the rotation sensor and 5 samples from the gyroscope sensor, because the sample rate is 5 Hz. Currently, we use a decision tree with Orange. It is straightforward to use other classifiers such as naïve Bayes, k-nearest neighbors, etc. For instance, suppose we have training and test data in the files “training.tab” and “test.tab”, respectively. We can load these data and classify them using the following Python code:

```python
import orange, orngTree;

train = orange.ExampleTable('training.tab')
test = orange.ExampleTable('test.tab')

bayes = orange.BayesLearner(train)
tree = orngTree.TreeLearner(train)
knn = orange.kNNLearner(train, k=5)

# Classify each test example.
for ex in test:
    activity_type_bayes = bayes.getClass()
    activity_type_tree = tree.getClass()
    activity_type_knn = knn.getClass()

    # Process the classes.
```
Once the data analysis center finishes analyzing a block of data, it will store the result into the database.

4.3.2 Data Analysis Tool - Orange

Orange is a component-based data mining and machine learning software suite, featuring visual programming front-end for explorative data analysis and visualization, and Python bindings and libraries for scripting. It includes a set of components for data preprocessing, feature scoring and filtering, modeling, model evaluation, and exploration techniques. It is implemented in C++ and Python. Its graphical user interface builds upon the cross-platform Qt framework. Orange is supported on various versions of Linux, Apple's Mac OS X, and Microsoft Windows.

Orange provides with a number of data mining and machine learning algorithms, such as naïve Bayes algorithm, k-nearest neighbors algorithm and CN2 algorithm. In our system, we choose decision tree learning algorithm.

Decision tree learning, used in statistics, data mining and machine learning, uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. More descriptive names for such tree models are classification trees or regression trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making. The
reason why we choose decision trees as the data analysis algorithm is because decision
trees are powerful and popular tools for classification and prediction and decision trees
represent rules which can be understood by humans and used in knowledge system such
as database.

For our activity recognition system, in order to establish such a decision tree for
activity recognition, we first need a certain amount of training data for each activity type.
After receiving these training data, the algorithm will construct a tree type structure based
on the features of training data. The leaves represent activity types, and the branches are
decision paths. When the tree is constructed, the decision model is fixed. Once the server
receives a block of raw data that need to be classified, decision tree algorithm will search
the tree from the root based on the features of the incoming data. The algorithm will
finally reach a leaf node whose activity type features are most close to that of the raw
data. Then we can figure out what kind of activity the incoming data represents.
CHAPTER 5

EVALUATION

This chapter includes some experiments that have been performed to validate the design and implementation of the activity recognition system. First, we benchmark the resource consumption of client application running on an Android Nexus S phone. We demonstrate that our sensor data collection application for the client can be deployed on an off-the-shelf smartphone. Next, we examine the accuracy of activity recognition module through an experiment with several people.

5.1 Resource Consumption Evaluation

Smart phones are equipped with limited computing resources and memory resources. If the application running on the phone consumes a lot of resources, it would affect the execution of other applications on the phone, and then it would not be a practical application for real-life use. Therefore, resource consumption evaluation of the application is necessary. We evaluate the performance of the application on an Android Nexus S smartphone, for CPU, battery and memory. Phones in this experiment are equipped with an extended life battery (1500 mAh) and 16GB of NAND memory, partitioned as 1GB internal storage and 15GB microSD.
Table 1. Average CPU and Memory Usage of the Application on the Mobile Phone

Table 1 shows the average CPU and memory consumption of the application running on the smartphones. During the experiment to measure the CPU and memory usage, we found the application never uses more than 10% CPU resource and the memory usage is always below 14KB. It means this application consumes only limited computing resource and storage resource. We are not surprised about this result, because in our original design we have taken the resource consumption problem into consideration. Smartphones usually do not have a very fast CPU and a very large amount of memory because of its limited size. Meanwhile, users always have some other applications running in the background which also keep using CPU resource and memory resource, so we do not want our application to interfere with those applications.

We perform an experiment with five people to captures battery life performance and data generation rates for the application running on the mobile phone. Subjects are asked to go about their normal daily routines. Figure 8 and Figure 9 present per-subjects values for daily data generation and battery life, respectively. Battery life varies from user to user by 36%. This value increases to 42% for data generation. However, the results for data generation indicate that 16 MB external micro SD storage is sufficient. Similarly, the experiment shows that the battery life is consistently above 15 hours, which is sufficient
to run our application for an entire day if recharged once very briefly during the day as well as each night.

Figure 10. Daily Data Generation by Subjects during One Week Experiment
5.2 Recognition Accuracy Evaluation and Analysis

5.2.1 Benchmark Case

For the benchmark case, we use three motion sensors (accelerometer sensor, rotation vector sensor and gyroscope sensor) with a sampling rate of 5Hz. And we use the same group of people for training data collection and recognition accuracy test. The summary of the result in the activity recognition experiments of the benchmark case is presented in Table 2. This table specifies the predictive accuracy associated with each of the activities, for the learning algorithm and for a simple "straw man" strategy. The straw man strategy always predicts the specified activity (i.e., walking for the first row in Table 2 and jogging for the second row of Table 2) or, when assessing the overall performance of the
classifier (i.e., the last row of Table 2), always predicts the most frequently occurring activity, which happens to be walking. The baseline straw man strategy allows us to consider the degree of class imbalance when evaluating the performance of the activity recognition system.

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Predicted Activity</th>
<th>Walking</th>
<th>Jogging</th>
<th>Stairs</th>
<th>Sitting</th>
<th>Standing</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking</td>
<td>1480</td>
<td>6</td>
<td>130</td>
<td>1</td>
<td>1</td>
<td>91.0</td>
<td></td>
</tr>
<tr>
<td>Jogging</td>
<td>7</td>
<td>1170</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>97.8</td>
<td></td>
</tr>
<tr>
<td>Stairs</td>
<td>179</td>
<td>25</td>
<td>754</td>
<td>3</td>
<td>4</td>
<td>78.1</td>
<td></td>
</tr>
<tr>
<td>Sitting</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>266</td>
<td>3</td>
<td>97.4</td>
<td></td>
</tr>
<tr>
<td>Standing</td>
<td>2</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>201</td>
<td>94.8</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Accuracies of Activity Recognition

Table 2 demonstrates that in most cases we can achieve high levels of accuracy. For the two most common activities, walking and jogging, we generally achieve the accuracies above 90%. Jogging appears easier to identify than walking, which seems to make sense since jogging involves more extreme changes in acceleration. It appears much more difficult to identify the two stair climbing activities, but as we shall see, that is because those two similar activities are often confused with one another. Note that although there are very few instances of sitting and standing, we can still identify these activities quite well, because, as noted earlier, the two activities cause the device to
change orientation and this is easily detected from the accelerometer data. Our results indicate that the decision tree learning algorithm consistently performs well, and the accuracy of activity recognition is acceptable.

5.2.2 Recognition Accuracy Comparison

In the section, we make a comparison between the recognition accuracies with those of another system.

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Walking</th>
<th>Jogging</th>
<th>Stairs</th>
<th>Sitting</th>
<th>Standing</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking</td>
<td>1524</td>
<td>7</td>
<td>148</td>
<td>2</td>
<td>2</td>
<td>90.6</td>
</tr>
<tr>
<td>Jogging</td>
<td>10</td>
<td>1280</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>96.9</td>
</tr>
<tr>
<td>Stairs</td>
<td>185</td>
<td>33</td>
<td>784</td>
<td>4</td>
<td>4</td>
<td>77.6</td>
</tr>
<tr>
<td>Sitting</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>272</td>
<td>4</td>
<td>96.5</td>
</tr>
<tr>
<td>Standing</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>209</td>
<td>93.7</td>
</tr>
</tbody>
</table>

Table 3. Accuracies of Activity Recognition of Another Work[6]

Table 3 shows the activity recognition accuracies of J. Kwapisz, G. Weiss and S. Moore’s work [6]. In their work, the only motion sensor they use is the accelerometer sensor, and instead of collecting sample data 5 times per second, they choose a much higher sampling rate – 20Hz. Although they use classification techniques from WEKA data mining suite rather than Orange to induce models for predicting the user activities,
the classification algorithm they choose is also decision trees. From the summary results for their activity recognition system shown in Table 3, we can see that they are also able to achieve a very high level of accuracy for the four most common activities: walking, jogging, sitting and standing. For the two activities climbing upstairs and climbing downstairs, they also said these two activities are difficult to distinguish from each other. For recognizing climbing-up and climbing-down stairs, the accuracy they are able to achieve is just around 60%. Overall, the activity recognition accuracy their work can obtain is almost the same as that of ours. However, the sensor readings we need are 25% less than their system. For their work, they choose a sampling rate of 20Hz and use only one accelerometer, so the amount of data per second they need to collect is 3*20 = 60 float type values. Whereas, we use three different motion sensors and choose 5Hz as the data sampling rate, so the amount of data per second we have to collect is 3*3*5 = 45 float type values. So for every second, we save 15 float type data storage which is equal to 25% storage space save. Therefore, storage space saving is our system’s advantage over their work.

5.2.3 Increase the Sampling Rate

As we described in Chapter 4, the classification algorithm we choose for data mining and activity recognition is decision tree learning algorithm. One of the most important features of decision tree classification algorithm is the larger amount of training data you collect for establishing the decision tree model, the higher level of classification you will achieve in the prediction process. Therefore, we decide to increase
the size of training data for decision tree algorithm’s preprocessing procedure to see whether we can achieve a higher level of recognition accuracy. The method we use to increase training data size is to increase sampling rate.

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Predicted Activity</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Walking</td>
<td>Jogging</td>
</tr>
<tr>
<td>Walking</td>
<td>1495</td>
<td>3</td>
</tr>
<tr>
<td>Jogging</td>
<td>5</td>
<td>1184</td>
</tr>
<tr>
<td>Stairs</td>
<td>169</td>
<td>20</td>
</tr>
<tr>
<td>Sitting</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Standing</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4. Accuracies of Activity Recognition with Sampling Rate of 10Hz

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Predicted Activity</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Walking</td>
<td>Jogging</td>
</tr>
<tr>
<td>Walking</td>
<td>1500</td>
<td>4</td>
</tr>
<tr>
<td>Jogging</td>
<td>5</td>
<td>1203</td>
</tr>
<tr>
<td>Stairs</td>
<td>160</td>
<td>22</td>
</tr>
<tr>
<td>Sitting</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Standing</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5. Accuracies of Activity Recognition with Sampling Rate of 20Hz
Table 4 and Table 5 present the activity recognition accuracies of our system with higher sampling rates – 10Hz and 20Hz respectively. From these two tables, we can see that we indeed can achieve a higher level of activity recognition accuracy. However, the overall increment of accuracies is not very impressive. For the four most common activities, the system with the sampling rate of 5Hz already can reach a really high accuracy which is over 90% for each type activity. So the accuracy improvement of 1% - 2% is reasonable. However, the accuracy improvement of the two stairs climbing activity is also limited. Based on our analysis, we think the reason for this effect is people usually do not climb stairs with a very fast speed, so the motion sensor readings do not change sharply. Therefore, even you collect the sensor readings with a very high frequency, the trend of data value change does not vary very obviously. Therefore, increasing the sensor reading sampling rate only cannot effectively improve the recognition accuracy of stairs climbing activities.

5.2.4 Using a Different Phone Client

In this experiment, we use a different smart phone to collect training data and test recognition accuracy. It is an Android G1 phone which is an almost off-shelf model. It features an ARM-based, dual-core CPU capable of up to 4million triangles/sec, a 98MB RAM and a 70MB of internal storage. It uses an 1150mAh rechargeable lithium ion battery. It also provides an embedded accelerometer, a rotation vector sensor and a gyroscope sensor. So it can satisfy the minimal requirement of our experiment.
Table 6. Accuracies of Activity Recognition with Different Phone Client

Table 6 shows the accuracies of activity recognition with the Android G1 phone. Compared this result to that of the benchmark case, we can find that the accuracies they are able to achieve are almost the same. It means the motion sensors equipped in the Android G1 smart phone perform almost as well as those installed in that Nexus S phone. So we can see for these three motion sensors we use in our system do not change a lot during these years. However, the recognition accuracies provided by using the outmoded phone are always slightly lower than those by using the fashionable smart phone. Based on our analysis, we believe it is because of the CPU performance gap between these two devices. The CPU stalled in Android G1 is much slower than that in Nexus S, so the processing delay would be slightly different and can affect the performance of activity recognition. Nevertheless, the overall performance of the Android G1 phone is almost as good as that of Nexus S. Therefore, we can assert that our activity recognition system is
also suitable for old-fashioned smart phones as long as they are equipped with required motion sensors.

### 5.2.5 Using Different Number of Sensors

In the following experiment, we want to test each sensor’s contribution to the accuracies of activity recognition. Due to the fact that so far no one has been able to do activity recognition without using the accelerometer sensor, so conducting experiments for using rotation vector sensor and gyroscope sensor alone to recognize activities is meaningless. So our experiment can be divided into three parts: use the accelerometer sensor alone, use the accelerometer sensor plus the rotation vector sensor and use the accelerometer sensor plus the gyroscope sensor. Then compare the accuracies of activity recognition provided by them with those of the benchmark case.

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Predicted Activity</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking</td>
<td>1332 13 210 7 8</td>
<td>84.8</td>
</tr>
<tr>
<td>Jogging</td>
<td>25 947 78 0 5</td>
<td>90.2</td>
</tr>
<tr>
<td>Stairs</td>
<td>221 43 703 3 4</td>
<td>72.1</td>
</tr>
<tr>
<td>Sitting</td>
<td>9 0 7 212 9</td>
<td>89.5</td>
</tr>
<tr>
<td>Standing</td>
<td>8 4 15 0 186</td>
<td>87.3</td>
</tr>
</tbody>
</table>

Table 7. Accuracies of Activity Recognition with One Accelerometer Alone
Table 7 presents the accuracies of activity recognition by using one accelerometer alone with the sampling rate of 5Hz. Overall, the performance of activity recognition is good. However, compare Table 7 with Table 2 we can see that accuracies of activity recognition by using accelerometer sensor, rotation vector sensor and gyroscope sensor are much better than those by using accelerometer alone. It means rotation vector sensor and gyroscope sensor can help increase the accuracies. Whereas, their contribution cannot compare with that of the accelerometer, and the accelerometer plays the central role in activity recognition. From Table 3 and Table 7, we can see that when compared with the accuracies provided by J. Kwapisz, G. Weiss and S. Moore’s work [6], the activity recognition performance by choosing a lower sampling rate (5Hz) is not as good as that by choosing a higher sampling rate (20Hz). And for the activity recognition system using the accelerometer alone, increasing the sampling rate improves the recognition accuracies a lot.
Table 8. Accuracies of Activity Recognition with Accelerometer and Rotation Sensor

Table 8 shows that accuracies of activity recognition with the combination of the accelerometer sensor and the rotation vector sensor. From Table 8 we can see that the overall performance of activity recognition is improved. But the improvement is not very impressive for all activities. For the two inactive activities sitting and standing, rotation vector sensor improves the accuracies very well, but for the other three active activities, it seems the rotation vector sensor does not offer a lot of help. The reason for that is because rotation vector sensor describes the orientation of the phone with a combination of an angle and an axis, in which the device has rotated through an angle around an axis <x, y, z>. It means it describes the difference between the coordinate of the phone and coordinate of the earth which represents the relative position of the phone to the earth, and sensor readings provided by it are only about angles. So it only offers the information of the static state of the phone instead of the movement or the motion of the smartphone.
Therefore, it mainly contributes to recognize inactive type of activities and merely helps improve the recognition accuracies of active type of activities.

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Predicted Activity</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking</td>
<td>1312</td>
<td>89.3</td>
</tr>
<tr>
<td>Jogging</td>
<td>1085</td>
<td>95.5</td>
</tr>
<tr>
<td>Stairs</td>
<td>711</td>
<td>75.9</td>
</tr>
<tr>
<td>Sitting</td>
<td>252</td>
<td>94.9</td>
</tr>
<tr>
<td>Standing</td>
<td>188</td>
<td>93.1</td>
</tr>
</tbody>
</table>

Table 9. Accuracies of Activity Recognition with Accelerometer and Gyroscope Sensor

Table 9 shows that accuracies of activity recognition with the combination of the accelerometer sensor and the gyroscope sensor. From Table 9 we can see that the overall performance of activity recognition is also improved. From the comparison of Table 8 and Table 9, we can see that recognition accuracy improvement by adding a rotation vector sensor and by adding a gyroscope sensor is almost the same. But rotation vector sensor is good at recognizing inactive type of activities (sitting and standing), and gyroscope sensor is not only good at recognizing inactive type of activities but also good at recognizing active type of activities (walking, jogging and climbing stairs). As we presented in Chapter 4, gyroscope sensor describes angular velocity about x, y and z axes. So it is a sensor that detects the self-rotation motion of the phone. Now it is easy to
understand why it is good at recognizing active type of activities. And the reason why it is also good at recognizing inactive type of activities comes from the sensor readings themselves. For the inactive type of activities, the phone is almost static over time. Therefore, the values of sensor readings from the gyroscope sensor are very small. For sitting and standing, the values of training data from the gyroscope sensor are extremely close to zero. However, for other active type of activities, the values are always much larger than zero. To sum up, based on the data and analysis above, the three sensors’ contribution to activity recognition can be ranked as: accelerometer sensor > gyroscope sensor > rotation vector sensor.

5.2.6 Recognize Similar Activities

In the benchmark case, we discussed that it appears that it is very difficult for our system to distinguish climbing up stairs activity and climbing down stairs activity, because the sensor data collected for these two activities are so similar to each other and the classification algorithm is always confused with recognizing each other. Therefore, in order to test whether our system does not do well in distinguish similar activities we conduct an experiment with adding a new activity which is similar to jogging – running. It means we want to see whether our system is able to activities with different moving speed. According to Wiki [31], the definition of jogging as compared with running is not standard. One definition describes jogging as running slower than 6 miles per hour (10 km/h). Therefore, in our experiment, we define that the activity with a moving speed
faster than 6 miles per hour is running and the activity with a moving speed slower than 6 miles per hour is jogging.

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Walking</th>
<th>Jogging</th>
<th>Running</th>
<th>Stairs</th>
<th>Sitting</th>
<th>Standing</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking</td>
<td>1371</td>
<td>7</td>
<td>2</td>
<td>127</td>
<td>0</td>
<td>1</td>
<td>90.9</td>
</tr>
<tr>
<td>Jogging</td>
<td>6</td>
<td>981</td>
<td>112</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>87.2</td>
</tr>
<tr>
<td>Running</td>
<td>4</td>
<td>6</td>
<td>1203</td>
<td>24</td>
<td>0</td>
<td>1</td>
<td>96.8</td>
</tr>
<tr>
<td>Stairs</td>
<td>75</td>
<td>98</td>
<td>73</td>
<td>725</td>
<td>1</td>
<td>1</td>
<td>74.5</td>
</tr>
<tr>
<td>Sitting</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>258</td>
<td>2</td>
<td>97.2</td>
</tr>
<tr>
<td>Standing</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>198</td>
<td>93.0</td>
</tr>
</tbody>
</table>

Table 10. Accuracies of Activity Recognition with Fast Speed Running

Table 10 shows the accuracies of activity recognition with adding a new activity – running. Overall, we still can achieve a high level of accuracy for each activity. And for most activities, walking, climbing stairs, sitting and standing, recognition accuracies for them merely change. For the running activity, surprisingly our system is able to achieve a very high accuracy which is almost 97%. However, unfortunately, adding the running activity influences recognizing jogging activity a lot. From the comparison of Table 2 and Table 10, we can figure out that the recognition accuracy of jogging decrease by almost 10%, and as might have been expected, in most misrecognition cases, jogging is confused with running. We believe the reason for this is because that accelerometer is not able to
distinguish different speed. As we discussed early, the accelerometer sensor plays the central role in activity recognition. However, the sensor readings from the accelerometer sensor are the linear acceleration along the x, y, and z axis of the phone, not the linear moving speed of the phone along these three axes. Then it becomes difficult for the system to measure the instantaneous velocity of the phone by using sensor readings from the accelerometer sensor alone. Fortunately, the flaw can be partially reduced by using gyroscope sensor. Because the sensor readings provided by the gyroscope sensor represents the instantaneous angular velocity of the phone about the x, y, and z axis. But the problem still cannot be totally solved by the gyroscope sensor, because of the limitation of training data. As we know, when some people are running at the same speed, it cannot be guaranteed that people are using the same stride frequency because of the difference between their heights. Stride frequency directly influences the gyroscope sensor reading, so the bigger stride frequency difference is in the training set, the more difficulties the system will have in distinguishing similar motions with different speed.

5.2.7 Using Separate Test Set

For all the experiments we conducted previously, we always use the same group of people (five people) to collect training data for each activity and to test the activity recognition accuracy of the system. In the following experiment we use a different group of people to test our system’s activity recognition accuracy to see how using a different test set would affect the recognition accuracies. We do not modify or update the training data for each activity from the original five people. We only ask three other people to
wear an Android phone with our sensor data collecting application installed to sensor readings for their activities and then apply these data to do activity recognition and analysis.

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Walking</th>
<th>Jogging</th>
<th>Stairs</th>
<th>Sitting</th>
<th>Standing</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking</td>
<td>1152</td>
<td>29</td>
<td>210</td>
<td>4</td>
<td>6</td>
<td>82.2</td>
</tr>
<tr>
<td>Jogging</td>
<td>32</td>
<td>973</td>
<td>62</td>
<td>7</td>
<td>9</td>
<td>89.8</td>
</tr>
<tr>
<td>Stairs</td>
<td>191</td>
<td>35</td>
<td>703</td>
<td>5</td>
<td>7</td>
<td>74.7</td>
</tr>
<tr>
<td>Sitting</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>259</td>
<td>2</td>
<td>96.9</td>
</tr>
<tr>
<td>Standing</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>197</td>
<td>93.1</td>
</tr>
</tbody>
</table>

Table 11. Accuracies of Activity Recognition with Different Test Set

Table 11 presents the experiment result of using a different group of people to do recognition accuracy testament. For the two inactive activities, sitting and standing, the accuracies are still maintained at a high level. However, for the active type of activities, the accuracies showed in Table 11 are surprisingly low compared with those of Table 2 provided by the benchmark case. This fact indicates that the activity recognition model established so far does not apply very well to new test set. Furthermore, the inaccuracies demonstrate that the activity classification model the decision tree learning algorithm established is not very universally applicable for recognizing active type of activities. It means the training data collected from the original five people for each activity is also not
universally typical enough or is lack of typicality. The main reason for the inaccuracies is because of the limited number of people for training data collection. As we described above, due to the limitation of the scale of our experiment, we only have a five people group in charge of training data collection. So the training data collected reflects the common activity features of these five persons, and the decision tree classification model based on these training data does very well in recognizing their activities which is the reason why our system is able to achieve a very high level of accuracies in the benchmark case experiment. Whereas the common characteristics of these five people’s activities cannot be applied to all the people around the world. So the number of people plays an essential role in constructing a widely applicable model, because the larger number of people who participate in training data collecting, the more common activity features we will have. Therefore, in order to make the classification model more universally applicable, we need more people to collect training data and each person also need to collect as much training data for each activity as possible in order to offer a much more exact activity characteristic.
CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this thesis, we discussed the design, implementation and evaluation of a comprehensive activity recognition system capable of monitoring and recognition people’s daily activities. The application on the mobile phone is a real-time, continuous sensing application in charge of data collection and data transmission. Results analyzed from the server can help users better understand the wellbeing impact of their day to day physical activity. Our activity recognition system implementation demonstrates the viability of personal wellbeing applications using smartphones.

However, our system still has some limitations. For example, since we only use one smart phone to collect sensor data, the placement of the phone is a big problem. For the same activity, if we put the phone on different part of user’s body, the data we collect from the sensors are extremely different, and then the decision tree algorithm establishes a totally different classification model. Therefore, in order to simplify the experiment, we assume that users always put the phones in their pants’ pocket. What is more, we only
tried the decision tree learning algorithm to do data mining, so we have no idea of the performance of the system if we use other learning algorithms.

Our work would not have been possible without establishing our Android-based data collection platform, and we view this software and hardware architecture, through which data is transmitted by the phone to our Internet-based server, as a key resource produced as a consequence of this work. By having this in place we will be able to mine other mobile sensor data much more quickly.

6.2 Future Work

We plan to improve our activity recognition in several ways. Areas of improvements involve: 1) learning to recognize additional activities, such as bicycling and car-riding, 2) obtaining training data from more users to improve our accuracy of activity recognition, 3) generating additional and more sophisticated features when aggregating the raw time-series data, and 4) evaluating the impact of carrying the cell phone in different locations, such as on a belt loop. In addition, we plan to significantly enhance our data analysis platform so that we can generate results in real-time, although our current results were generated off-line not being reported back to the mobile phone and the users. We plan to provide real-time results in two ways. The first way is to minimize the intelligence required on the phone by having the phone transmit the data to the Internet-based sever over the cellular connection, as usual, with the server applying the activity recognition model and transmitting the results back to the phone. In one variant, the phone will send the raw accelerometer data and orientation sensor data and in another variant the phone
will perform the data transformation step and only transmit the data when an example is generated. The second method involves implementing the activity recognition model directly on the smartphone. Given the computational power of these devices, this is certainly a feasible option. One key advantage of this method is that it removes the need of a server, which makes the solution perfectly scalable and ensures the user’s privacy since the sensor data is kept locally on the device.
BIBLIOGRAPHY


Accelerometer Embedded Mobile Phone with Varying Positions and Orientations,”

Acceleration Features with Mobile Phones,” in Proc. IMCE, 2009.


