ISLAMIC UNIVERSITY OF TECHNOLOGY(IUT)
Department of Computer Science and Engineering

A. M. Ridwan
Student Id: 094409
Computer Science & Engineering
Islamic University of Technology
Board Bazar, Gazipur
E-mail: arifcit@iut-dhaka.edu

Faiyad Rahman
Student Id: 094423
Computer Science & Engineering
Islamic University of Technology
Board Bazar, Gazipur
E-mail: nahidcit@iut-dhaka.edu

Human Activity Recognition Using Smart Phone Sensors

B.Sc Thesis
October, 2013

Supervisor: Md. Kamrul Hasan
Assistant Professor
Computer Science & Engineering
Islamic University of Technology
Board Bazar, Gazipur
E-mail: hasan@iut-dhaka.edu

Acknowledgement

At first we would like to thank our respected supervisor Md. Kamrul Hasan, PhD, Assistant Professor,CSE Department for his guidence, supervision and earnest support through out the whole thesis work. We thank all the teacher of CSE department for their support and enthusiasm.

October, 2013

A. M. Ridwan

Faiyad Rahman

Table of Contents

List of Figures

Abstract

In this work, algorithms are developed and evaluated to detect physical activities from data acquired using the sensors available in smart phones-accelerometer, gyroscopes, global positioning system (GPS), carrying it in pockets. Data was collected from 20 subjects without supervision or observation. Subjects were asked to perform a sequence of everyday tasks but not told specifically where or how to do them. Mean, energy, frequency-domain entropy, and correlation of acceleration data was calculated and several classifiers using these features were tested. Decision tree classifiers showed the best performance recognizing everyday activities with an overall accuracy rate of 84%. The results show that although some activities are recognized well with subject-independent training data, others appear to require subject-specific training data. The results suggest that multiple accelerometers aid in recognition because conjunctions in acceleration feature values can effectively discriminate many activities.

Chapter 1

Introduction

1.1 Background

Providing accurate and opportune information on people's activities and behaviors is one of the most important and well-studied tasks in pervasive computing. Innumerable applications can be visualized for medical, security, entertainment, and tactical purposes. In recent years, the prominent development of sensing devices (e.g., accelerometers, cameras, GPS, etc.) has facilitated the process of measuring attributes related to the individuals and their surroundings. In addition, most sensors are nowadays equipped with communication capabilities which allow their integration with other mobile devices within Personal Area Networks (PANs) or Body Area Networks (BANs). However, most applications require much more than simply collecting measurements from variables of interest. In fact, additional challenges for enabling context awareness involve the design of techniques to perform data analysis and inference, as the raw data (e.g., acceleration signals or electrocardiogram) provided by the sensors are, in many cases, useless. Even though the first works in human activity recognition (HAR) date back to the late 90's [1], there are still significant research challenges within the field. In this work, we concentrate on one of the most relevant ones: the implementation of a HAR system in a mobile phone. This task becomes difficult because of the energy and computational constraints present in the devices. Such limitations are particularly critical for activity recognition applications which entail high demands due to decoding, feature extraction, classification, and transmission of large amounts of raw data. Furthermore, to the best of our knowledge, available machine learning API's such as WEKA [2] and JDM [3] are not supported by current mobile platforms. This fact ac-

centuates the necessity of an efficient mobile library for evaluating machine learning algorithms and implementing HAR systems in mobile devices. At the same time, we foresee that a mobile HAR system will bring important advantages and benefits. For instance, a HAR system running in a BAN should substantially reduce energy expenditures, as raw data would not have to be continuously sent to a server for processing. The system would also become more robust and responsive because it would not depend on unreliable wireless communication links, which may be unavailable or error prone; this is particularly important for medical or military applications that require real-time decision making. Finally, a mobile HAR system would be more scalable since the server load would be alleviated by the locally performed feature extraction and classification computations. In this paper, we introduce a mobile framework in support of real-time human activity recognition under the Android platform -reported as best-selling smartphone platform in 2010 by Canalys [4] - to address previously mentioned issues. A mobile application was implemented on top of Centinela [5], a HAR system based on acceleration and physiological signals to automatically recognize physical activities. The application utilizes the C4.5 classification algorithm as a proof of concept. The implementation and evaluation of our framework present the following main results and contributions:

- A library for the mobile evaluation of classification algorithms (MECLA) was successfully implemented.
- An application for real-time HAR was implemented in an Android cellular phone. It uses MECLA and supports multiple sensing devices integrated in a BAN.
- The evaluation shows that the system can be effectively deployed in current cellular phones. Indeed, the application can run for up to 12.5 continuous hours with a response time of no more than 8% of the window length and an overall accuracy of 92.6%.
- Different users with diverse characteristics participated in training and testing phases, assuring flexibility to support new users without the need of re-training the system.

1.2 Keywords

Human activity recognition, Mobile devices and sensors, sensor data, Human pose recognition, Classification Algorithms

1.3 Motivation

The main motivation behind the research is to make activity recognition easy and accurate. With the use of external sensors, desired accuracy has been achieved. But with the rise of technology era, modern smart phones are equipped with all the sensors one requires to detect human activity. Till now only primary activities are recognizable using the mobile phone sensors. To detect much more complex activities is the main goal of this research.

We chose smart phones to reduce/eliminate the use of external sensors. As carrying external sensors is a hassle and also leads user discomfort.

Also till now all the activity recognition decisions are done in cloud based classification. We plan on creating a standalone application for the mobile phones which will not need cloud to recognize activities.

Chapter 2

Literature Review

2.1 Install multiple inertial sensors on the body. [1, 2, 3, 4]:

Activity Detection Approach:

Supervised Learning:

Homes and their furnishings have highly variable layouts, and individuals perform activities in many different ways. The same activity (e.g. brushing teeth) may result in a significantly different sensor activation profile based upon the habits, or routines of the home occupant and the layout and organization of the particular home. One approach to handling such variability is to use supervised learning with an explicit training phase.

Probabilistic classification:

Probabilistic reasoning offers a way to deal with ambiguous and noisy information from multiple sensors.

Model-based vs instance-based learning:

Model-based algorithms use the training examples to construct a mathematical model of the target classification function, which avoids the need to save all examples as raw data. This could help alleviate end-user privacy concerns.

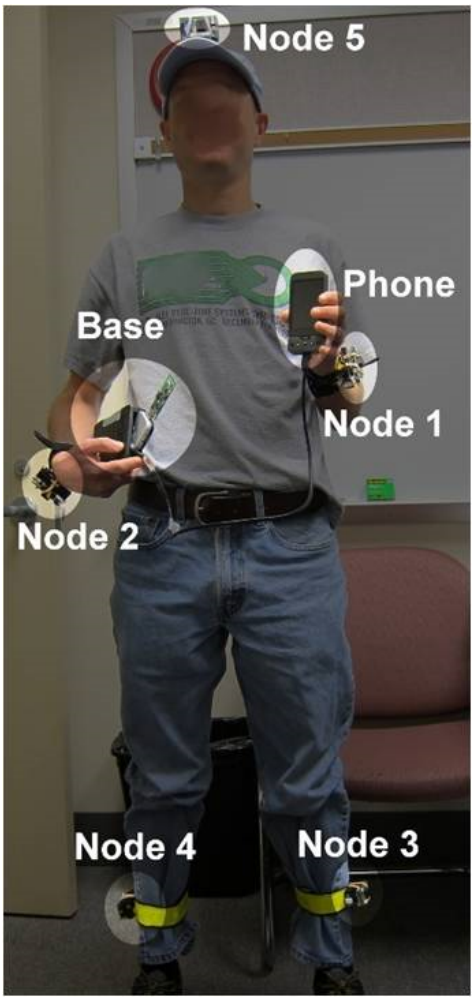


Figure 2.1: Botnet Architecture

Activity recognition system architecture:

The proposed system consists of three major components: (1) The environmental state-change sensors used to collect information about use of objects in the environment, (2) the context-aware experience sampling tool (ESM) used by the end user to label his or her own activities, and (3) the pattern recognition and classification algorithms for recognizing activities after constructing a model based on a training set.

Environmental State-Change Sensors:

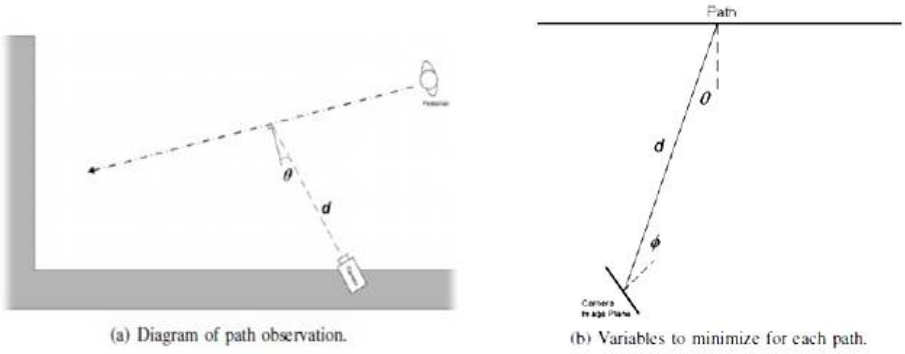
Although other low-cost wireless sensing systems have been developed, notably Berkeley Motes and Smart-ITS, their power and cost points still pose a challenge for researchers interested in distributing hundreds of units in a single home to collect synchronized data for several weeks or longer. The cost of these devices is relatively high because they are designed as multi-purpose sensors.
Therefore, we have designed a new set of tape-on sensors optimized to perform a single task at low cost: measuring change in the state of an object in home. To achieve well-synchronized measurements, the most precise real-time clock hardware was used in each board. Further, the signals from each board were linearly interpolated to match the reference clock better after the end of the study. These highly-specialized boards are 3-5 times less expensive than Smart-ITS and Motes, which dramatically increases the number that can be installed in homes working within a tight research budget. The estimated battery life of the data collection board is one year if the external sensor is activated an average of 10 times per day for 30 seconds.

Context-Aware Experience Sampling:

Supervised learning algorithms require training data. In the laboratory, obtaining annotated data is a straightforward process. Researchers can directly observe and label activity in real-time, or later through observation of video sequences. In the home environment, however, direct observation is prohibitively time-consuming and invasive. One alternative is to use the Experience Sampling Method (ESM). When using ESM, subjects carry a personal digital assistant (PDA) that is used as timing device to trigger self-reported diary entries. The PDA samples (via a beep) for information. Multiple choice questions can then be answered by the user.

2.2 Use one or multiple camera to do a vision-based recognition [5, 6]:

The goal of camera placement for optimal path observability is to position a camera to observe the entire path of motion while maximizing the view of the subject in the image. The first part of that goal, observing the entire path, requires the camera to be far enough away from the subject that the entire motion is captured within the camera field of view. The second part of the goal, maximizing the view of the subject, requires the camera to be as close to the subject as possible, so that the subject is as large as possible in the image. Figure 2 depicts the reason for this. For a perspective projection camera with a fixed field of view, the size of an object in an image decreases as the distance to the object increases. In digital imaging, the area of an object in an image corresponds to a number of pixels that measure the object. As a result, we can define observability metrics directly in terms of pixel resolution ((1) and (2)).



(a) Diagram of path observation.

(b) Variables to minimize for each path.

Figure 2.2: (a)Diagram of Path Observation (b)Variables to minimize for each path

Chapter 3

Background Study

3.1 Human Activity Recognition:

Sensor-based activity recognition integrates the emerging area of sensor networks with novel data mining and machine learning techniques to model a wide range of human activities. Mobile devices (e.g. smart phones) provide sufficient sensor data and calculation power to enable physical activity recognition to provide an estimation of the energy consumption during everyday life. Sensor-based activity recognition researchers believe that by empowering ubiquitous computers and sensors to monitor the behavior of agents these computers will be better suited to act on our behalf. Approaches to activity recognition are through logic and reasoning, Probabilistic reasoning, WI FI based activity recognition, Data mining approach. Logic-based approaches keep track of all logically consistent explanations of the observed actions. Thus, all possible and consistent plans or goals must be considered. Kautz provided a formal theory of plan recognition. He described plan recognition as a logical inference process of circumscription. All actions, plans are uniformly referred to as goals, and a recognizer's knowledge is represented by a set of first-order statements called event hierarchy encoded in first-order logic, which defines abstraction, decomposition and functional relationships between types of events. Probability theory and statistical learning models are more recently applied in activity recognition to reason about actions, plans and goals. When activity recognition is performed indoors and in cities using the widely available Wi-Fi signals and 802.11 access points, there is much noise and uncertainty. These uncertainties are modeled using a dynamic Bayesian network model. A multiple goal model that can reason about user's interleaving goals where a deterministic state transition

model is applied. A better model that models the concurrent and interleaving activities in a probabilistic approach is proposed by Hu and Yang. A user action discovery model is presented by Yin where the Wi-Fi signals are segmented to produce possible actions. Different from traditional machine learning approaches, an approach based on data mining has been recently proposed. The problem of activity recognition is formulated as a pattern-based classification problem. They proposed a data mining approach based on discriminative patterns which describe significant changes between any two activity classes of data to recognize sequential, interleaved and concurrent activities in a unified solution. Gilbert use 2D corners in both space and time. These are grouped spatially and temporally using a hierarchical process, with an increasing search area. At each stage of the hierarchy, the most distinctive and descriptive features are learned efficiently through data mining.

3.2 Mobile Devices and Sensors:

Many recent wearable systems for activity recognition place a single type of sensor, typically accelerometers, in multiple locations (anywhere from two to 12) on the body. However, this approach's obtrusive usage model has limited its mass adoption. In addition, its use of a single sensor type restricts the range of activities it can recognize for example, accelerometers are mainly useful for inferring a limited set of physical activities. An alternate approach is to use multiple sensor types-that is, multimodal sensors-and collect data from a single body location. Some older research in activity and context recognition explores this approach. Recent studies have shown that the information gained from multimodal sensors can offset the information lost when sensor readings are collected from a single location. The sensors' complementary cues are also useful for recognizing a wider range of activities. For example, an accelerometer and audio together can detect whether the user is sitting versus sitting and watching TV.

3.3 Sensor data:

Different types of sensors are:

1. proximity sensor
2. orientation sensor

3. light sensor
4. pressure sensor
5. accelerometer sensor
6. gravity sensor
7. gyroscope sensor
8. linear acceleration sensor
9. rotation vector sensor and
10. temperature sensor

3.4 Human Pose Recognition:

Mobile phones have very good cameras nowadays. These cameras can now be used to determine the user's activities by recognizing the human pose. In this type of sensing, a camera is placed in a public place where lots of people are coming and going and by studying their activity, the camera recognizes the pose. Whether a person is in a idle pose or active pose, these are determined. The placement of the camera is the key factor in here because different people have different types of poses and style so it may be difficult for the camera to determine the pose if it is at an inconvenient location.

3.5 Classification techniques:

3.5.1 Iterative Dichotomizer 3(ID3):

ID3, or Iterative Dichotomiser 3 Algorithm, is a Decision Tree learning algorithm. The name is correct in that it creates Decision Trees for "dichotomizing" data instances, or classifying them discretely through branching nodes until a classification "bucket" is reached (leaf node).By using ID3 and other machine-learning algorithms from Artificial Intelligence, expert systems can engage in tasks usually done by human experts, such as doctors diagnosing diseases by examining various symptoms (the attributes) of patients (the data instances) in a complex Decision Tree. Of course, accurate Decision Trees are fundamental to Data Mining and Databases.

The input data of ID3 is known as sets of "training" or "learning" data instances, which will be used by the algorithm to generate the Decision Tree. The machine is "learning" from this set of preliminary data.
Entropy tells us how well an attribute will separate the given example according to the target classification class.

Entropy(S) = -Ppos log2 Ppos - Pneg log2 Pneg

- Ppos = Proportion of positive examples
- Pneg = proportion of negative example

3.5.2 K- Nearest Neighbor:

In pattern recognition, the k-nearest neighbor algorithm (k-NN) is a non-parametric method for classifying objects based on closest training examples in the feature space. k-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. The k-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of its nearest neighbor.

The same method can be used for regression, by simply assigning the property value for the object to be the average of the values of its k nearest neighbors. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. (A common weighting scheme is to give each neighbor a weight of 1/d, where d is the distance to the neighbor. This scheme is a generalization of linear interpolation.)

The neighbors are taken from a set of objects for which the correct classification (or, in the case of regression, the value of the property) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. The k-nearest neighbor algorithm is sensitive to the local structure of the data.

Nearest neighbor rules in effect implicitly compute the decision boundary.

It is also possible to compute the decision boundary explicitly, and to do so efficiently, so that the computational complexity is a function of the boundary complexity.

Nearest neighbor problem has been extensively studied in the field of Computational geometry under the name closest pair of point's problem.

3.6 Problem statement:

The aim is to design a simple, yet accurate system that can learn and recognize complex and versatile human activity in real time. The method will implement the use of accelerometer, gyroscope, gravity sensor, GPS, barometer and proximity sensor, all that come along with the latest smart phones. The device will stay in the user's pockets.

We have to compare the classification algorithms and design a module which fits our requirements- accuracy and real time activity recognition.

We also look to eliminate the necessity of cloud based recognition. So all the processes will be done in the smart phone itself within the application. The application implementing the methods will be a standalone app which will require minimal user interaction to gather data sets or recognizing the activities. Apart from "start/end" session, user will not need to do anything

3.7 Proposed Solution Approach:

3.7.1 Data Collection:

We will be using a smart phone- a Samsung Galaxy S II in this case, to gather raw data from the test users. The phone has the following sensors in it:

- 3D Accelerometer (50 Hz)
- Gyroscope
- GPS
- Gravity sensor

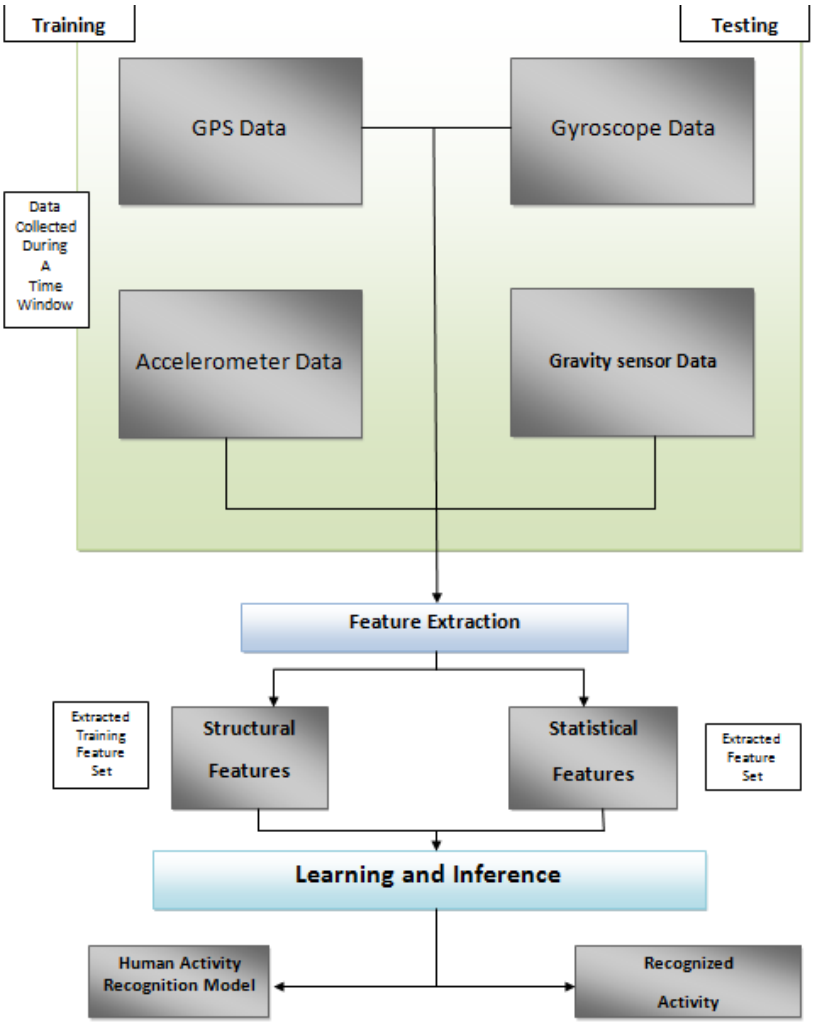

Training
Testing
GPS Data
Gyroscope Data
Data Collected During A Time Window
Accelerometer Data
Gravity sensor Data
Feature Extraction
Extracted Training Feature Set
Structural Features
Statistical Features
Extracted Feature Set
Learning and Inference
Human Activity Recognition Model
Recognized Activity

Figure 3.1:

The phone will be in the user's front pocket, which is around waist height.

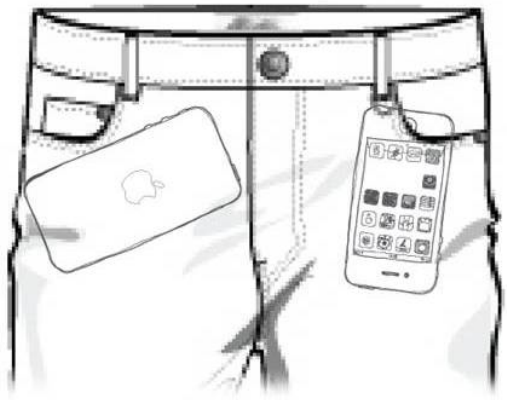


Figure 3.2:

We used 5 test users to gather data and they were told to do the following 6 activities:

- Walking
- Jogging
- Ascending stairs
- Descending stairs
- Sit
- Stand

The users were free to do other activities too. The data gathering phase was monitored by the research team.

So after the initial data gathering we got the following table of data sets

The accelerometer gave us data for x, y and z axis. The gathered data were then given as input in Mat lab and we obtained waveforms for acceleration vs. time.

ID	Walk	Jog	Up	Down	Sit	Stand	Total
1	74	15	13	25	17	7	151
2	48	15	30	20	0	0	113
3	62	58	25	23	13	9	190
4	65	57	25	22	6	8	183
5	65	54	25	25	77	27	273

Table 3.1: Number of Examples per User and Activity

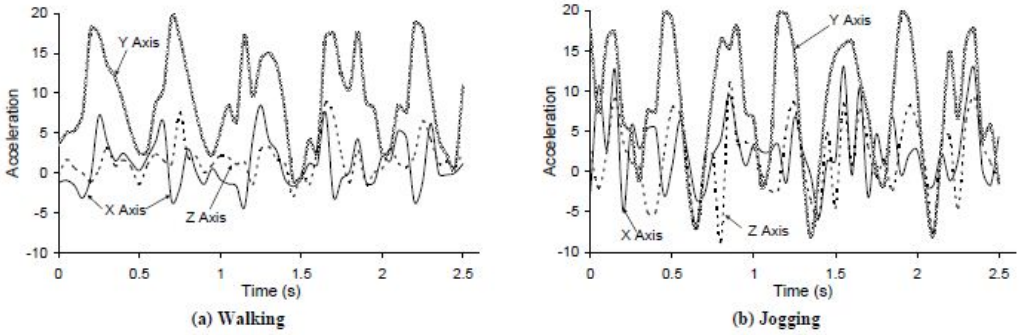


Figure 3.3:

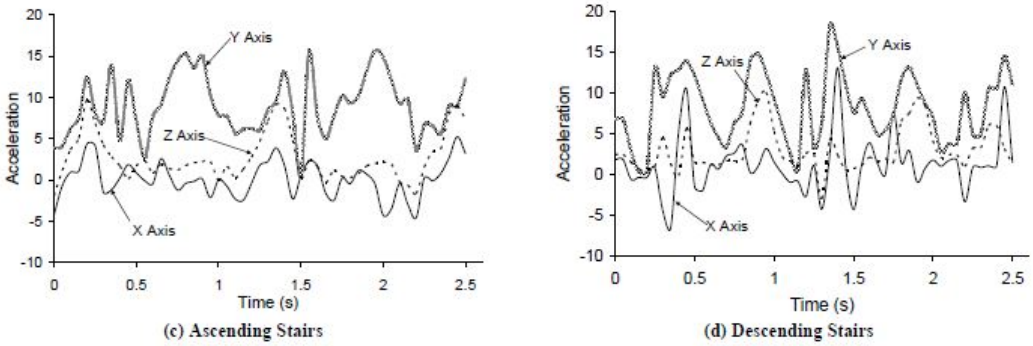


Figure 3.4:

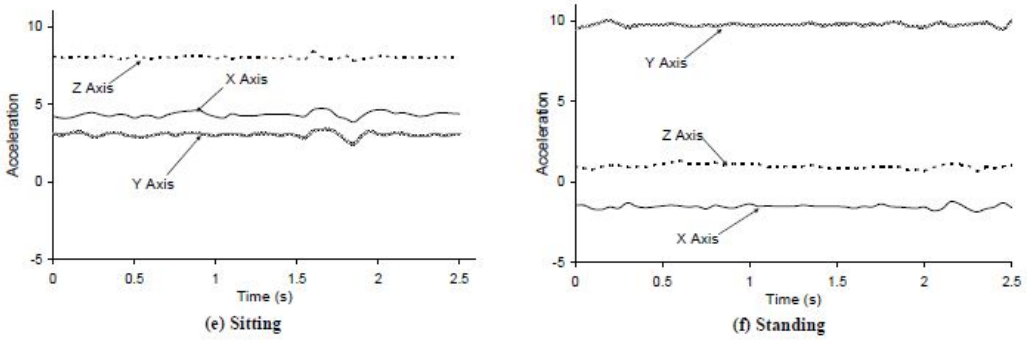


Figure 3.5:

From the gathered data we can see that if we want to recognize ascending/descending stairs, it is much more easily done by using the readings from gravity sensor.

3.7.2 Detecting Falls:

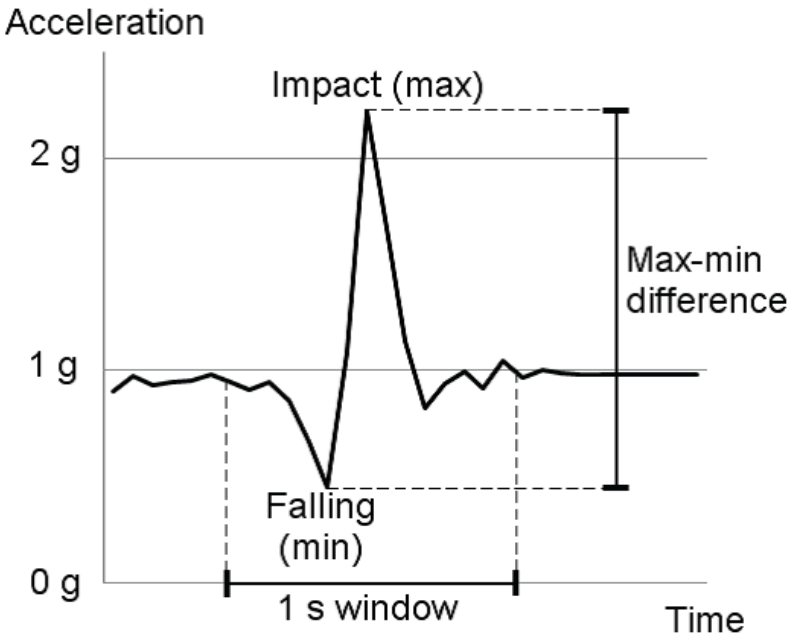


Figure 3.6:

A typical acceleration pattern during a fall is a decrease in acceleration followed by an increase, as shown in Figure 9. This is because an accelerometer at rest registers 1 g (the Earth's gravity) and during free fall 0 g. When a person starts falling, the acceleration decreases from 1 g to around 0.5 g (perfect free fall is never achieved). Upon the impact with the ground, a short strong increase in the acceleration is measured. To detect falls with a threshold, we used the length of the acceleration vector, which means that we ignored the direction of the acceleration. The minimum and the maximum acceleration within a one-second window were measured. If the difference between the maximum and the minimum exceeded 1 g and the maximum came after the minimum, we declared that a fall had occurred. We augmented fall detection with the measurement of the person's orientation after a potential fall.

3.7.3 Feature Extraction:

Features were computed on 512 sample windows of acceleration data with 256 samples overlapping between consecutive windows. At a sampling frequency of 76.25 Hz, each window represents 6.7 seconds. Mean, energy, frequency-domain entropy, and correlation features were extracted from the sliding windows signals for activity recognition. Feature extraction on sliding windows with 50% overlap has demonstrated success in past works. A window of several seconds was used to sufficiently capture cycles in activities such as walking, window scrubbing, or vacuuming. The 512 sample window size enabled fast computation of FFTs used for some of the features. The DC feature is the mean acceleration value of the signal over the window. The energy feature was calculated as the sum of the squared discrete FFT component magnitudes of the signal. The sum was divided by the window length for normalization. Additionally, the DC component of the FFT was excluded in this sum since the DC characteristic of the signal is already measured by another feature. Note that the FFT algorithm used produced 512 components for each 512 sample window. Use of mean and energy of acceleration features has been shown to result in accurate recognition of certain postures and activities.

Frequency-domain entropy is calculated as the normalized information entropy of the discrete FFT component magnitudes of the signal. Again, the DC component of the FFT was excluded in this calculation. This feature may support discrimination of activities with similar energy values. For instance, biking and running may result in roughly the same amounts of energy in the hip acceleration data. However, because biking involves a nearly uni-

form circular movement of the legs, a discrete FFT of hip acceleration in the vertical direction may show a single dominant frequency component at 1 Hz and very low magnitude for all other frequencies. This would result in low frequency-domain entropy. Running on the other hand may result in complex hip acceleration and many major FFT frequency components between 0.5 Hz and 2 Hz. This would result in higher frequency-domain entropy. Features that measure correlation or acceleration between axes can improve recognition of activities involving movements of multiple body parts. Correlation is calculated between the two axes of each accelerometer hoarder board and between all pair wise combinations of axes on different hoarder boards. Some of these feature for two activities. It was anticipated that certain activities would be difficult to discriminate using these features. For example, "watching TV" and "sitting" should exhibit very similar if not identical body acceleration. Additionally, activities such as "stretching" could show marked variation from person to person and for the same person at different times. Stretching could involve light or moderate energy acceleration in the upper body, torso, or lower body.

3.7.4 Human Activity Recognition Model

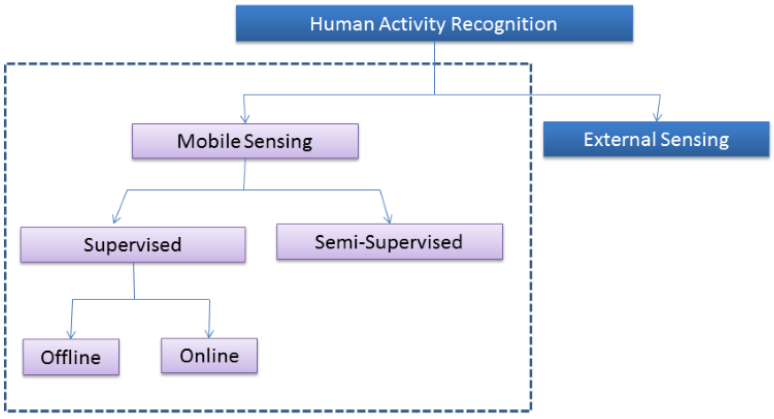


Figure 3.7:

In recent years, the prominent development of sensing devices (e.g., accelerometers, cameras, GPS,etc.) has facilitated the process of collecting attributes related to the individuals and their surroundings. However, most applications require much more than simply gathering measurements from

variables of interest. In fact, additional challenges for enabling context awareness involve knowledge discovery since the raw data (e.g., acceleration signals or electrocardiogram) provided by the sensors are often useless. For this purpose, HAR systems make use of machine learning tools, which are helpful to build patterns to describe, analyze, and predict data.

In a machine learning context, patterns are to be discovered from a set of given examples or observations denominated instances. Such input set is called training set. In our specific case, each instance is a feature vector extracted from signals within a time window. The examples in the training set may or may not be labeled, i.e., associated to a known class (e.g. walking, running, etc.). In some cases, labeling data is not feasible because it may require an expert to manually examine the examples and assign a label based upon their experience. This process is usually tedious, expensive, and time consuming in many data mining applications. There exist two learning approaches, namely supervised and unsupervised learning, which deal with labeled and unlabeled data, respectively. Since a human activity recognition system should return a label such as walking, sitting, running, etc, most HAR systems work in a supervised fashion. Indeed, it might be very hard to discriminate activities in a completely unsupervised context. Some other systems work in a semi supervised fashion allowing part of the data to be unlabeled.

Supervised learning:

Labeling sensed data from individuals performing different activities is a relatively easy task. Some systems store sensor data in a non-volatile medium while a person from the research team supervises the collection process and manually registers activity labels and time stamps. Other systems feature a mobile application that allows the user to select the activity to be performed from a list. In this way, each sample is matched to an activity label, and then stored in the server.

Decision trees:

Decision trees build a hierarchical model in which attributes are mapped to nodes and edges represent the possible attribute values. Each branch from the root to a leaf node is a classification rule. C4.5 is perhaps the most widely used decision tree classifier and is based on the concept of information gain to select which attributes should be placed in the top nodes.

Bayesian methods:

Bayesian methods calculate posterior probabilities for each class using estimated conditional probabilities from the training set. The Bayesian Network (BN) classifier and Naive Bayes (NB)-which is a specific case of BN- are the principal exponents of this family of classifiers. A key issue in Bayesian Networks is the topology construction, as it is necessary to make assumptions on the independence among features. For instance, the NB classifier assumes that all features are conditionally independent given a class value, yet such assumption does not hold in many cases. As a matter of fact, acceleration signals are highly correlated, as well as physiological signals such as heart rate, respiration rate, and ECG amplitude.

Semi-supervised learning:

Relatively few approaches have implemented activity recognition in a semi-supervised fashion, thus, having part of the data without labels. In practice, annotating data might be difficult in some scenarios, particularly when the granularity of the activities is very high or the user is not willing to cooperate with the collection process. Since semi-supervised learning is a minority in HAR, there are no standard algorithms or methods, but each system implements its own approach. Section V-C provides more details on the state-of-the-art semi-supervised activity recognition approaches.

3.7.5 Classification:

The classification model (shown in Figure 9) was generated by the Iterative Dichotomizer 3 (ID3) algorithm using 5-second-long time windows with 50% overlap. As a proof of concept, in the present study, three activities were considered, namely running, walking, and sitting. Two new users (a male and a female), who did not participate in the training phase, performed each activity in a sequential fashion during approximately 5 minutes. A total of 360 instances (i.e., time windows) were classified. The evaluation encompasses three main aspects: accuracy, response time, and energy consumption.

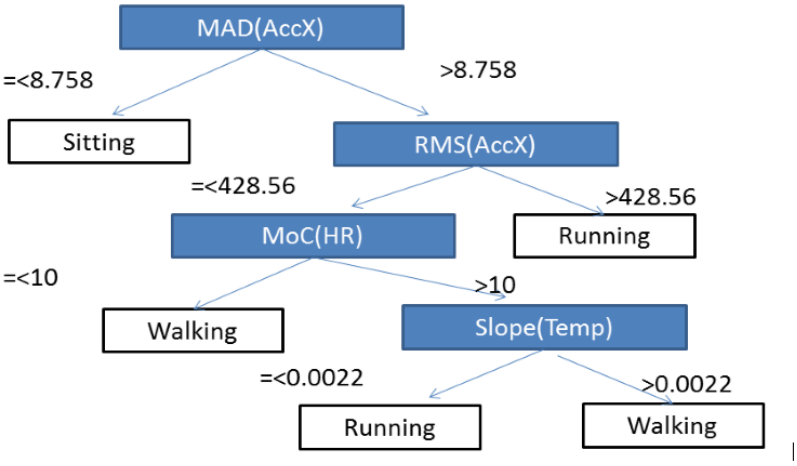


Figure 3.8:

ID3 Algorithm

Algorithm 1 ID3 - Mapper

Require: TD = data f or training
Ensure: (attribute - activity), 1
1: for all object ϵ L
do
2: for all attribute ϵ object.attributes
do
3: yield $\Leftarrow$ (attribute), 1
4: end for
5: end for

Algorithm 2 ID3 - Reducer Counter

Require: (attribute - activity), ocurrences
Ensure: (attribute - value, activity),(activity probability)
1: probability $\Leftarrow$ sum(values)
2: yield $\Leftarrow$ (key.attribute, key.value),(key.activity, key.value

Algorithm 2 ID3 - Reducer Counter

Algorithm 3 ID3 - Tree Reducer
Require: (attribute - value),(activity - activity value - probability)
Ensure: (attribute - value),(attribute entropy - attribute probability)
1: globalFreq ⇐ sum(values.probability)
2: for all activity ϵ value.activities
do
3: p local ⇐ .probability
4: entropy ⇐ activity.probability*log2(activity.probability)
5: entropy ⇐ activity.probability*log2(activity.probability)
6: end for
7: entropy ⇐ entropy*(-1)
8: yield(key), (entropy)

Algorithm 4 ID3 - Gain Calculator

Require: (activity-value), (entropy-localfrequency)
Ensure: attribute, gain
1: for all activity ϵ value.activities do 2: gain ⇐ gain + calculate gain(activity.entropy)
3: end for
4: yield ⇐ key.attribute,gain

Algorithm 5 ID3 - Summarize Gain by Attribute

Require: attribute, gain
Ensure: attribute - global gain
1: s gain ⇐ sum(values)
2: yield ⇐ key, s gain

3.7.6 Results:

From the initial sets of data gathered from table 1

So from the results we can say that the working module is functioning properly and with the right amount of extra features we will be able to increase the accuracy.

Activity	Accuracy
Walking	89.6%
Running	91.3%%
Biking	73.2%
Sitting	95%
Climbing	Stairs 98%
Jumping	96.4%
Sleeping	88.7%

Table 3.2:

3.8 Source Code:

The application was developed using Android SDK 4.3 in Eclipse. It used the Google maps API along with integration with Google Earth for simulation.

The codes are as followed:

DistanceNotifier.java:

```
public class DistanceNotifier implements StepListener, SpeakingTimer.Listener
{

public interface Listener {
public void valueChanged(float value);
public void passValue();
}
}

private Listener mListener;
float mDistance = 0;

PedometerSettings mSettings;
Utils mUtils;

boolean mIsMetric;
float mStepLength;
```

```
public DistanceNotifier(Listener listener, PedometerSettings settings, Utils
utils) {
mListener = listener;
mUtils = utils;
mSettings = settings;
reloadSettings();
}

public void setDistance(float distance) {
mDistance = distance;
notifyListener();
}
public void reloadSettings() {
mIsMetric = mSettings.isMetric();
mStepLength = mSettings.getStepLength();
notifyListener();
}
public void onStep()

if (mIsMetric) {
mDistance += (float)(// kilometers
mStepLength // centimeters
/ 100000.0); // centimeters/kilometer
}
else {
mDistance += (float)(// miles
mStepLength // inches
/ 63360.0); // inches/mile
}
notifyListener();
}

private void notifyListener() {
mListener.valueChanged(mDistance);
}

public void passValue() {
// Callback of StepListener - Not implemented
}
```

```
public void speak() {
if (mSettings.shouldTellDistance()) {
if (mDistance >= .001f)

mUtils.say((“” + (mDistance + 0.000001f)).substring(0, 4) + (mIsMetric
? " kilometers" : " miles"));
// TODO: format numbers (no "." at the end)
}
}
}

}
```

PaceNotifier.java:

```
package name.pedometer;


import java.util.ArrayList;
public class PaceNotifier implements StepListener, SpeakingTimer.Listener {


public interface Listener {
public void paceChanged(int value);
public void passValue();
}
private ArrayList<Listener> mListeners = new ArrayList<Listener>();


int mCounter = 0;


private long mLastStepTime = 0;
private long[] mLastStepDeltas = {-1, -1, -1, -1};
private int mLastStepDeltasIndex = 0;
private long mPace = 0;


PedometerSettings mSettings;
Utils mUtils;
```

```
/** Desired pace, adjusted by the user */
int mDesiredPace;

/** Should we speak? */
boolean mShouldTellFasterslower;

/** When did the TTS speak last time */
private long mSpokenAt = 0;

public PaceNotifier(PedometerSettings settings, Utils utils) {
mUtils = utils;
mSettings = settings;
mDesiredPace = mSettings.getDesiredPace();
reloadSettings();
}
public void setPace(int pace) {
mPace = pace;
int avg = (int)(60*1000.0 / mPace);
for (int i = 0; i < mLastStepDeltas.length; i++) {
mLastStepDeltas[i] = avg;
}
notifyListener();
}
public void reloadSettings()

mShouldTellFasterslower = mSettings.shouldTellFasterslower() && mSettings.getMaintainO
== PedometerSettings.M_PACE;
notifyListener(); }

public void addListener(Listener l) {
mListeners.add(l);
}

public void setDesiredPace(int desiredPace) {
mDesiredPace = desiredPace;
}

public void onStep() {
```

```
long thisStepTime = System.currentTimeMillis();
mCounter ++;

// Calculate pace based on last x steps
if (mLastStepTime > 0) {
long delta = thisStepTime - mLastStepTime;

mLastStepDeltas[mLastStepDeltasIndex] = delta;
mLastStepDeltasIndex = (mLastStepDeltasIndex + 1) % mLastStepDeltas.length;

long sum = 0;
boolean isMeaningfull = true;
for (int i = 0; i < mLastStepDeltas.length; i++) {
if (mLastStepDeltas[i] < 0) {
isMeaningfull = false;
break;
}
sum += mLastStepDeltas[i];
}
if (isMeaningfull && sum > 0) {
long avg = sum / mLastStepDeltas.length;
mPace = 60*1000 / avg;

// TODO: remove duplication. This also exists in SpeedNotifier
if (mShouldTellFasterslower && !mUtils.isSpeakingEnabled()) {
if (thisStepTime - mSpokenAt > 3000 && !mUtils.isSpeakingNow()) {
float little = 0.10f;
float normal = 0.30f;
float much = 0.50f;

boolean spoken = true;
if (mPace < mDesiredPace * (1 - much)) {
mUtils.say("much faster!");
}
else
if (mPace > mDesiredPace * (1 + much)) {
mUtils.say("much slower!");
}
else
```

```
if (mPace < mDesiredPace * (1 - normal)) {
mUtils.say("faster!");
}
else if (mPace > mDesiredPace * (1 + normal)) {
mUtils.say("slower!");
}
else
if (mPace < mDesiredPace * (1 - little)) {
mUtils.say("a little faster!");
}
else
if (mPace > mDesiredPace * (1 + little)) {
mUtils.say("a little slower!");
}
else {
spoken = false;
}
if (spoken) {
mSpokenAt = thisStepTime;
}
}
}
}
else {
mPace = -1;
}
}
mLastStepTime = thisStepTime;
notifyListener();
}

private void notifyListener() {
for (Listener listener : mListeners) {
listener.paceChanged((int)mPace);
}
}

public void passValue() {
// Not used
}
```

```
//————————————————————————————
// Speaking

public void speak() {
if (mSettings.shouldTellPace()) {
if (mPace > 0) {
mUtils.say(mPace + " steps per minute");
}
}
}

}
```

SpeedNotifier.java:

```
package name.pedometer;

public class SpeedNotifier implements PaceNotifier.Listener, SpeakingTimer.Listener
{

public interface Listener {
public void valueChanged(float value);
public void passValue();
}
private Listener mListener;

int mCounter = 0;
float mSpeed = 0;

boolean mIsMetric;
float mStepLength;

PedometerSettings mSettings;
Utils mUtils;
```

```
/** Desired speed, adjusted by the user */
float mDesiredSpeed;

/** Should we speak? */
boolean mShouldTellFasterslower;
boolean mShouldTellSpeed;

/** When did the TTS speak last time */
private long mSpokenAt = 0;

public SpeedNotifier(Listener listener, PedometerSettings settings, Utils utils)
{
mListener = listener;
mUtils = utils;
mSettings = settings;
mDesiredSpeed = mSettings.getDesiredSpeed();
reloadSettings();
}
public void setSpeed(float speed) {
mSpeed = speed;
notifyListener();

public void reloadSettings() {
mIsMetric = mSettings.isMetric();
mStepLength = mSettings.getStepLength();
mShouldTellSpeed = mSettings.shouldTellSpeed();
mShouldTellFasterslower = mSettings.shouldTellFasterslower() && mSettings.getMaintainO
== PedometerSettings.M_SPEED; notifyListener();
}
public void setDesiredSpeed(float desiredSpeed) {
mDesiredSpeed = desiredSpeed;
}

private void notifyListener() {
mListener.valueChanged(mSpeed);
}

public void paceChanged(int value) {
if (mIsMetric) {
```

```
mSpeed = // kilometers / hour
value * mStepLength // centimeters / minute
/ 100000f * 60f; // centimeters/kilometer
}
else {
mSpeed = // miles / hour
value * mStepLength // inches / minute
/ 63360f * 60f; // inches/mile
}
tellFasterSlower();
notifyListener();
}

/**
* Say slower/faster, if needed.
*/
private void tellFasterSlower() {
if (mShouldTellFasterslower && mUtils.isSpeakingEnabled()) {
long now = System.currentTimeMillis();
if (now - mSpokenAt > 3000 && !mUtils.isSpeakingNow()) {
float little = 0.10f;
float normal = 0.30f;
float much = 0.50f;

boolean spoken = true;
if (mSpeed < mDesiredSpeed * (1 - much)) {
mUtils.say("much faster!");
}
else
if (mSpeed > mDesiredSpeed * (1 + much)) {
mUtils.say("much slower!");
}
else
if (mSpeed < mDesiredSpeed * (1 - normal))

mUtils.say("faster!");

else
if (mSpeed > mDesiredSpeed * (1 + normal)) {
mUtils.say("slower!");
```

```
}
else
if (mSpeed < mDesiredSpeed * (1 - little)) {
mUtils.say("a little faster!");
}
else
if (mSpeed > mDesiredSpeed * (1 + little)) {
mUtils.say("a little slower!");
}
else {
spoken = false;
}
if (spoken) {
mSpokenAt = now;
}
}
}
}

public void passValue() {
// Not used
}

public void speak() {
if (mSettings.shouldTellSpeed()) {
if (mSpeed >= .01f) {
mUtils.say(("" + (mSpeed + 0.000001f)).substring(0, 4) + (mIsMetric ? "
kilometers per hour" : " miles per hour"));
}
}

}

}
```

StepListener.java:

```
package name.pedometer;
public interface StepListener {
public void onStep();
public void passValue();
}
```

Utils.java:

```
package name.pedometer;

import java.util.Locale;

import android.app.Service;
import android.speech.tts.TextToSpeech;
import android.text.format.Time;
import android.util.Log;

public class Utils implements TextToSpeech.OnInitListener {
private static final String TAG = "Utils";
private Service mService;

private static Utils instance = null;

private Utils() {
}

public static Utils getInstance() {
if (instance == null) {
instance = new Utils();
}
return instance;
}

public void setService(Service service) {
mService = service;
```

```
}

/********** SPEAKING **********/

private TextToSpeech mTts;
private boolean mSpeak = false;
private boolean mSpeakingEngineAvailable = false;

public void initTTS() {
// Initialize text-to-speech. This is an asynchronous operation.
// The OnInitListener (second argument) is called after initialization com-
pletes.
Log.i(TAG, "Initializing TextToSpeech...");
mTts = new TextToSpeech(mService, this // TextToSpeech.OnInitListener
);
}
public void shutdownTTS() {
Log.i(TAG, "Shutting Down TextToSpeech...");

mSpeakingEngineAvailable = false;
mTts.shutdown();
Log.i(TAG, "TextToSpeech Shut Down.");

}
public void say(String text) {
if (mSpeak && mSpeakingEngineAvailable) {
mTts.speak(text, TextToSpeech.QUEUE_ADD, // Drop all pending entries
in the playback queue. null);
}
}

// Implements TextToSpeech.OnInitListener. public void onInit(int status)
{
// status can be either TextToSpeech.SUCCESS or TextToSpeech.ERROR.
if (status == TextToSpeech.SUCCESS) {
int result = mTts.setLanguage(Locale.US);
if (result == TextToSpeech.LANG_MISSING_DATA || result == Text-
ToSpeech.LANG_NOT_SUPPORTED) {
```

```
// Language data is missing or the language is not supported.
Log.e(TAG, "Language is not available.");
} else {
Log.i(TAG, "TextToSpeech Initialized.");
mSpeakingEngineAvailable = true;
}
} else {
// Initialization failed.
Log.e(TAG, "Could not initialize TextToSpeech.");
}
}

public void setSpeak(boolean speak) {
mSpeak = speak;
}

public boolean isSpeakingEnabled() {
return mSpeak;
}

public boolean isSpeakingNow() {
return mTts.isSpeaking();
}

public void ding() {
}

/********** Time **********/

public static long currentTimeInMillis() {
Time time = new Time();
time.setToNow();
return time.toMillis(false);
}
}
```

3.9 Application Screenshot

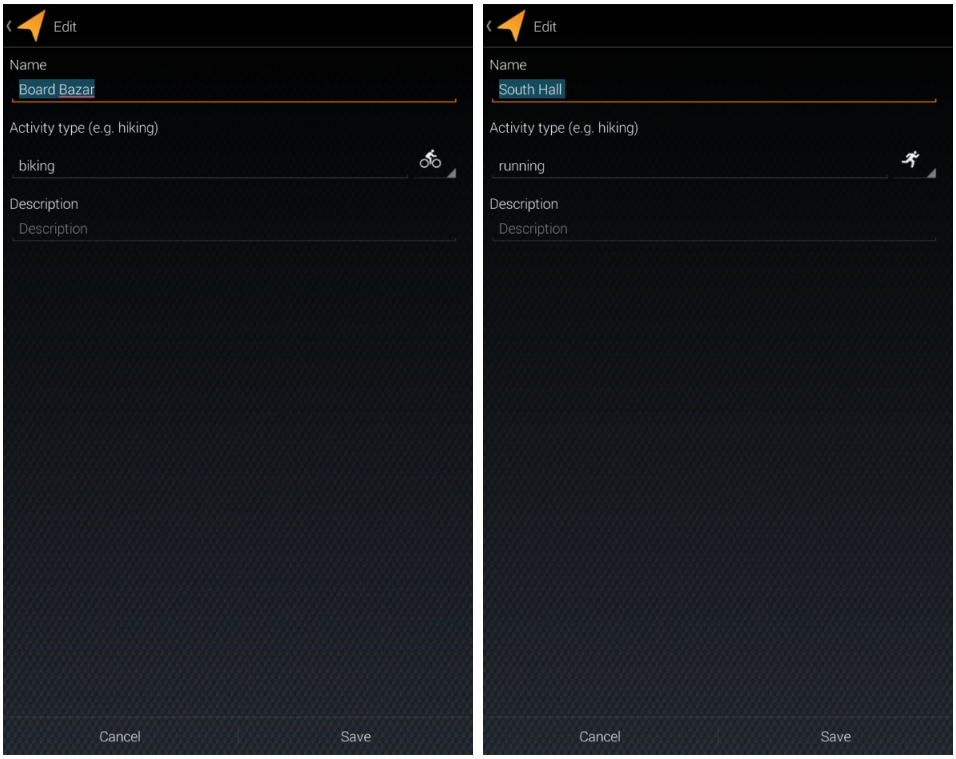


Figure 3.9:

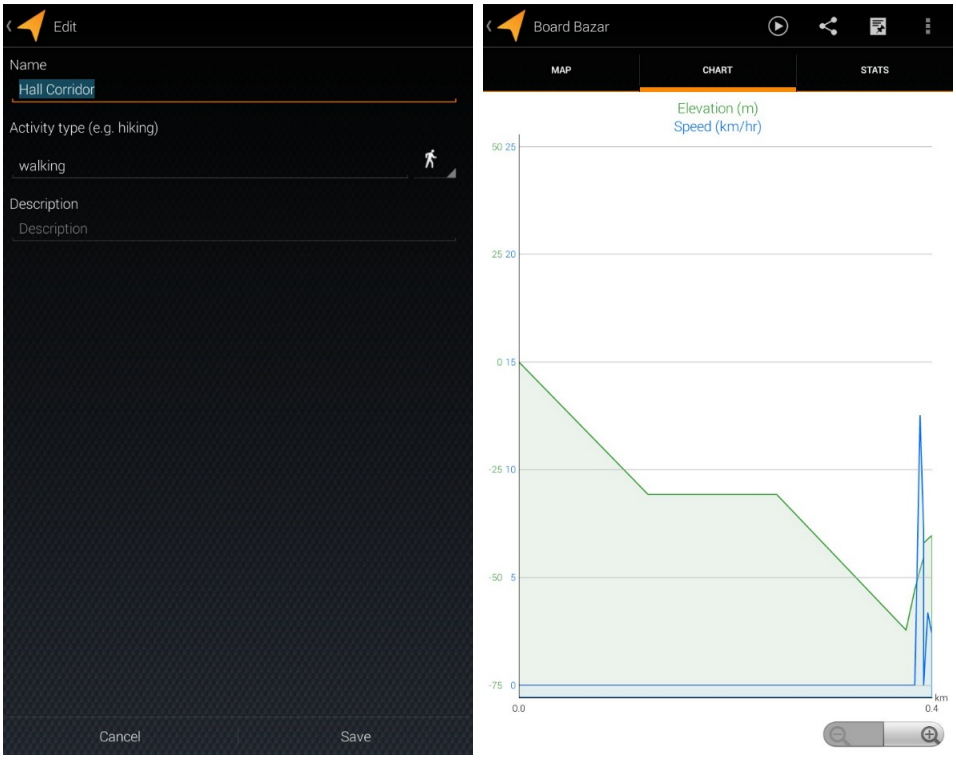

Edit
Name
Hall Corridor
Activity type (e.g. hiking)
walking
Description
Description
Cancel
Save
Board Bazar
MAP
CHART
STATS
Elevation (m)
Speed (km/hr)
50 25
25 20
0 15
-25 10
-50 5
-75 0
0.0
0.4
km

Figure 3.10:

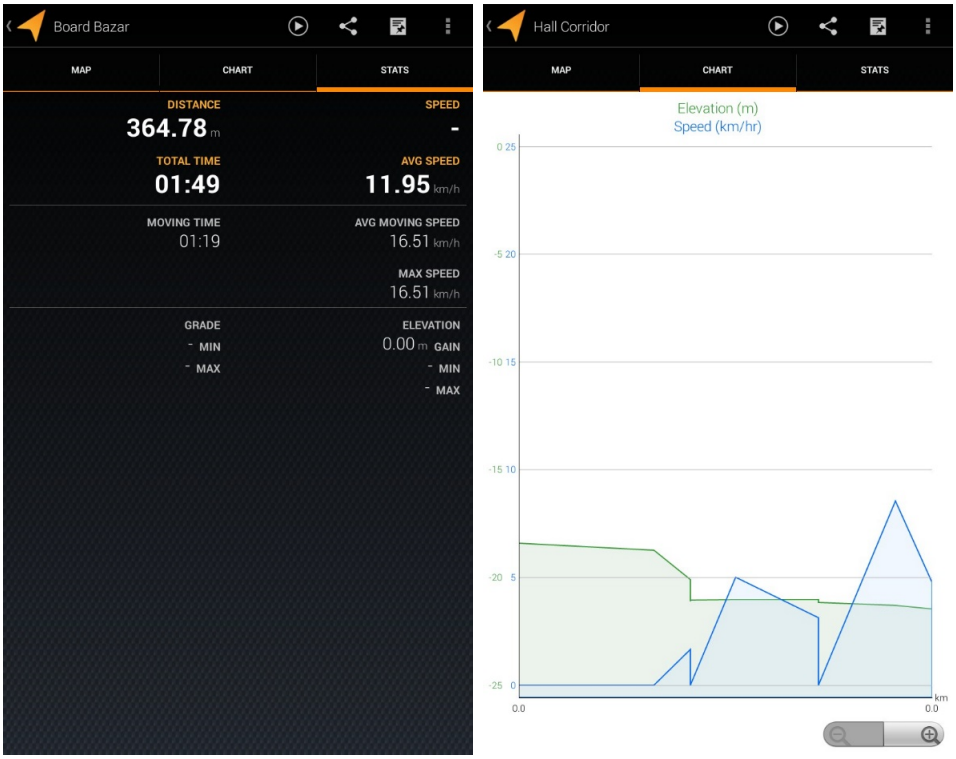

Board Bazar
MAP
CHART
STATS
DISTANCE
364.78 m
SPEED
-
TOTAL TIME
01:49
AVG SPEED
11.95 km/h
MOVING TIME
01:19
AVG MOVING SPEED
16.51 km/h
MAX SPEED
16.51 km/h
GRADE
- MIN
- MAX
ELEVATION
0.00 m GAIN
- MIN
- MAX
Hall Corridor
MAP
CHART
STATS
Elevation (m)
Speed (km/hr)
0 25
-5 20
-10 15
-15 10
-20 5
-25 0
km
0.0
0.0

Figure 3.11:

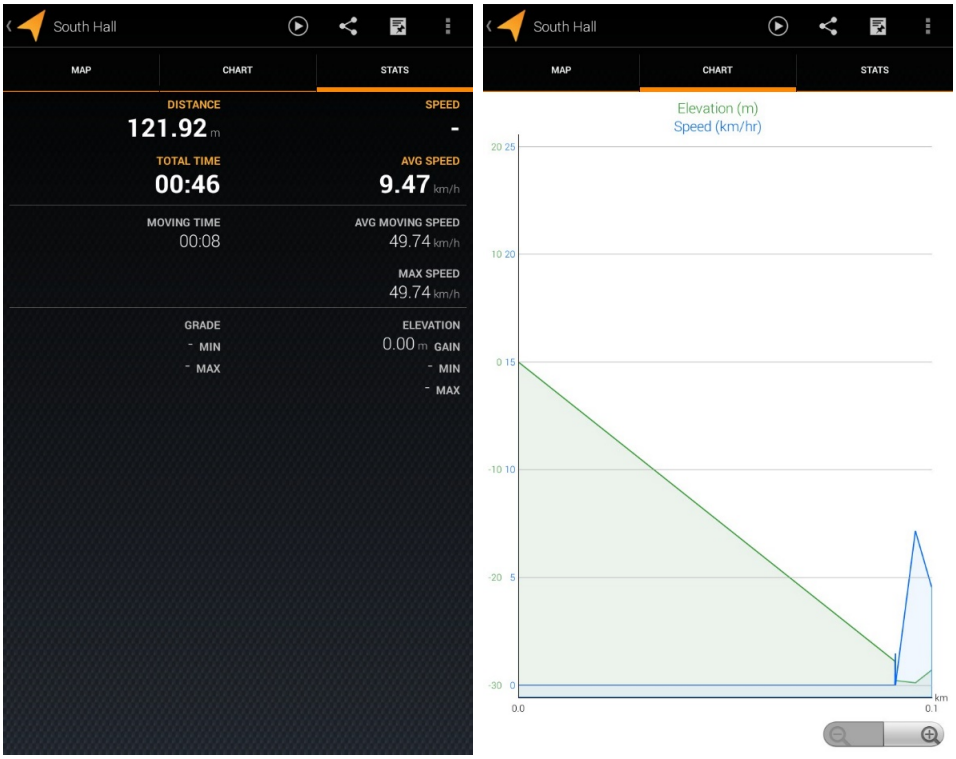

South Hall
MAP
CHART
STATS
DISTANCE
121.92 m
SPEED
-
TOTAL TIME
00:46
AVG SPEED
9.47 km/h
MOVING TIME
00:08
AVG MOVING SPEED
49.74 km/h
MAX SPEED
49.74 km/h
GRADE
- MIN
- MAX
ELEVATION
0.00 m GAIN
- MIN
- MAX
South Hall
MAP
CHART
STATS
Elevation (m)
Speed (km/hr)
20 25
10 20
0 15
-10 10
-20 5
-30 0
km
0.0
0.1

Figure 3.12:

Chapter 4

Conclusion

4.1 Conclusion

The amount of possibility in this field is endless and there is always room for improvement. The continuation of this research will open the door for further researches. Pin point activity recognition using mobile sensors will lead to a world full of data with overwhelming opportunities.

References

1 Carlos Paniagua*, Huber Flores, Satish Narayana Sriram,"Mobile Sensor Data Classification for Human Activity Recognition using MapReduce on Cloud ",The 9th International Conference on Mobile Web Information Systems (MobiWIS 2012)

2 Mitja LuŽtrek1, Hristijan Gjoreski1, Simon Kozina1, Bođidara Cvetkovi, Violeta Mirchevska, Matjađ Gams, Jođef Stefan,"Detecting Falls with Location Sensors and Accelerometers",Proceedings of the Twenty-Third Innovative Applications of Artificial Intelligence Conference

3 Tanzeem Choudhury,Sunny Consolvo, Beverly Harrison,Jeffrey Hightower,Anthony LaMarca, Louis LeGrand,Ali Rahimi, and Adam Rea Gaetano Borriello,Bruce Hemingway,Predrag "Pedja" Klasnja, Karl Koscher, James A. Landay,Jonathan Lester, and Danny Wyatt,Dirk Haehnel,"The Mobile Sensing Platform An Embedded Activity Recognition System",Pervasive Computing, IEEE (Volume:7, Issue:2)

4 Robert Bodor, Andrew Drenner, Michael Janssen, Paul Schrater and Nikolaos Papanikolopoulos,"Mobile Camera Positioning to Optimize the Observability of Human Activity Recognition Tasks",Intelligent Robots and Systems,2005. 2005 IEEE/RSJ International conference

5 Eunju Kim, Sumi Helal, and Diane Cook,"Human Activity Recognition and Pattern Discovery",Pervasive Computing, IEEE (Volume:9, Issue:1)

6 Kai-Tai Song and Wei-Jyun Chun, "Human Activity recognition using a mobile camera", The 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)

7 Oť scar D. Lara and Miguel A. Labrador,"A Survey on Human Activity Recognition using Wearable Sensors",Communications Surveys & Tutorials, IEEE (Volume:PP, Issue:99)

8 Zhixian Yan, Vigneshwaran Subbaraju, Dipanjan Chakraborty, Archan Misra, Karl Aberer,"Energy-Efficient Continuous Activity Recognition on Mobile Phones: An Activity-Adaptive Approach",Proceedings of the 16th International Symposium on Wearable Computers (ISWC)