PROJECT REPORT

ON

"Activity Recognition using a Smartphone App"

Submitted in partial fulfilment of the requirements

For the degree of

BACHELOR OF ENGINEERING

BY

1. **Ms Shivangi Saxena**
2. **Ms Zeba Raheman**

Under the guidance of

Prof. Sanjivani Deokar

Year 2013-2014

Department Of Computer Engineering

Lokmanya Tilak College of Engineering

Sector-4, Vikas Nagar, Koparkhairane, Navi Mumbai

PROJECT REPORT

"Activity Recognition using a Smartphone App"

Submitted By

1. **Ms Shivangi Saxena**
2. **Ms Zeba Raheman**

In partial fulfilment of the Degree of B.E. in Computer Engineering is approved.

Guide:

Ms Sanjivani Deokar

Examiner:

(1) _____

(Internal Guide)

(2) _____

_____

Prof. Anil Chhangani

(Head of Department)

_____

Dr V. K. Yakkundi

(Principal)

Date of Submission:

(College Stamp)

Table of Contents

Abstract

Actigraphy is a non-invasive method of monitoring human rest/activity cycles. A small actigraph unit, also called an actimetry sensor, is worn by a user to measure gross motor activity. Actigraphy through wrist watches and cell phones is a well-established procedure to monitor human activity. Its lightweight and non-intrusive nature make it a valuable tool to access the circadian cycle.

In this project, we collect data from a smartphone app, which is then analysed to detect the kind of motion the user is involved in (at rest, walking or running) and also calculate the amount of calories burnt while performing that activity. Data mining is performed to detect the relation between the changes in displacement along the x, y- & z-axis, and the corresponding activity performed. Displacement curves are created to give an idea about these patterns.

The project can be divided into two sections. The first sections corresponds to gathering data with the help of a hardware sensor in the cell phone, called an accelerometer. The accelerometer is a hardware sensor used to detect a shake motion. It can be used to detect when the user is walking, sleeping, running etc.

The second part of the project deals with using the gathered information for classification. For doing this classification, the ID3 algorithm is used. The ID3 algorithm is used to build a tree from a fixed set of examples (training data). The resulting tree is used to classify future samples (test data). The advantage of learning a decision tree is that a program, rather than a knowledge engineer, elicits knowledge from an expert.

With the processing results, the user can use the app as a fitness tracker, with information about what kind of activities they performed and how many calories they burnt.

Acknowledgements

We would like to take this opportunity to express our gratitude to all those who gave us the opportunity to complete this project.

We remain immensely obliged to *Prof. Sanjivani Deokar* for providing us with the moral and technical support and guiding us at each step of the project.

We also want to thank *Prof. Jyoti Sawant*, the Project Coordinator for being extremely helpful and cooperative.

We are grateful to our college, Lokmanya Tilak College of Engineering, Navi Mumbai, our *Principal Dr Vivek K. Yakkundi*, the Head of Computer Engineering, *Prof. Anil Chhangani* and all the teaching and non-teaching staff of the department for their kind support.

Chapter I: Background & Motivation

I. Introduction

Smartphones are gradually becoming more and more common in the mobile phone market. These phones don't just provide users with basic telephony applications but also allow multitasking & the deployment of a variety of sensors.

The integration of these mobile devices in our daily life is rapidly growing. It is envisioned that such devices will seamlessly keep track of our activities, learn from them, and subsequently help us to make better decisions regarding our future actions. Our project relies on this key concept. The objective of this project is to use the capabilities of the smartphone for recognising the user's activities.

Recognition is accomplished by exploiting the information retrieved from inertial sensors such as accelerometers. An accelerometer is an electromechanical device that will measure acceleration forces. It has become a customary sensor in most smartphones, especially ones built in the last two years. These sensors provide an unobtrusive method of tracking a user's physical activities without requiring much input from the user.

We developed a Smartphone app that takes in the data from these sensors and stores them in a database. A classifier is then used to recognise the user's activity. For training the classifier, a dataset of about 900 activity traces was collected. The training data was then made to go through a decision tree classifier, which then detected the kind of activity being performed. This detection was done by calculating entropy of each activity trace.

Thus we obtain a classifier that senses the user's activities. The readings obtained can also be used to obtain the amount of calories burnt by the user while performing the said activity.

II. Motivation

In the last decade or so, we have seen an explosion of mobile devices and services. With increasing advancement, more and more cell phones are turning into “smart” phones, with not just telephonic abilities, but also various other sensors and an innovative operating system. The proliferation of such phones is due to the assorted functions and utilities provided by them. Current smart phones are as much communication devices as they are a personal computer packaged into a pocket-sized gadget.

The sensors present in these devices are currently used for ergonomic purposes, along with detecting human gestures. Most smartphones now come with a gyroscope, an accelerometer, digital compass & GPS. These sensors can be incredibly useful in finding out more information about the user. This information was thought to be useful only in providing a better experience to the user. But in recent times, app developers are beginning to design apps that make use of this information to give users feedback on their daily habits & routines.

The development of activity recognition (AR) applications using smartphones has several advantages such as easy device portability without the need for additional fixed equipment, and comfort to the user due to the unobtrusive sensing. This contrasts with other established AR approaches which use specific purpose hardware devices such as in sensor body networks. Although the use of numerous sensors could improve the performance of a recognition algorithm, it is unrealistic to expect that the general public will use them in their daily activities because of the difficulty and the time required to wear them. Instead, we use smartphones, which are increasingly becoming ubiquitous with most users.

By involving smartphones, we aim towards successful human activity recognition with users being comfortable as a priority. Being different from wearable sensor-based systems which require separated processing unit, by using smartphone it is expected to have context processing integrated into one device where data has been collected. This will promise faster and more reliable decision result as it no longer requires specific communication medium between processing unit and sensor. Response with particular services once recognition is successful, will also now be possible as smartphone has a

wide range of connectivity options. With those several possibilities, eventually, smartphone-based system will enable human activity recognition being portable even during intense users' mobility activities or movements. A major advantage of actigraphy (using smartphones) is that it can conveniently record continuously for 24-hours a day for days, weeks or even longer.

III. Statement of the Problem

While going hiking, jogging, or simply taking a long walk around our neighbourhood, we might wonder as to the distance we covered, or specific details regarding how many calories we burnt while performing a particular activity. A specialized device for getting this data might be an expensive option. Instead, the data collection can be done simply with the help of a basic smartphone.

The problem as directed by this project is as follows-
The app developed as part of this project allows for the collection of data by the user. This data is then worked upon by a decision tree classifier. The app then informs the user about the activities performed along with the amount of calories burnt. To use the decision tree classifier, the entropy for each activity trace is calculate. The activity finally classified would be the one with the least value of entropy.

Thus, a total working pedometer and tracking system can be built using simple software in our smartphones without any additional hardware.

Chapter II: Literature Survey

I. Introduction

This survey was conducted using the references to the IEEE paper "Activity recognition using smartphones sensors" by Alvina Anjum and Muhammad U. Ilyas which states that Motion sensor embedded smartphones have provided a new platform for activity inference. These sensors, initially used for cell phone feature enhancement, are now being used for a variety of applications. Providing cell phone users with information about their own physical activity in an understandable format can enable users to make more informed and healthier lifestyle choices. In modern society which lead to various physical and mental diseases, such as obesity, coronary heart diseases, type II diabetes and depression, which request enormous medical cost. According to World Health Organization, there are at least 1.9 million people die as a result of physical inactivity annually. In U.S. alone, it leads to about 300, 000 preventable deaths and more than 90$ billion direct health cost annually. Even though people are aware of the benefits of exercises, there is a lack of external intervention which can properly bring the busy people out of the sedentary routine, thus an automatic and personal reminder will be very helpful if it can monitor one's physical activities and persuade people to participate in physical activities regularly at the right time and place. Activity recognition technology is a key enabling technology to tackle this problem as it's able to monitor individual's physical daily activities and the lasting duration so as to estimate the calories consumed each day.

There are several ways to recognize people's daily activities. One way is using cameras to visually detect people's motion. The drawback of this solution is that to monitor a moving person, large number of cameras need be deployed with high cost. And also the system should be designed to aggregate the information from each camera and deal with the influential factors such as lighting condition, mounting distance and angel, which make the system very complicate and impractical. Another way is using personal companion devices such as mobile phones or watches with sensing and computing power embedded to detect physical activities. The merit of this solution is that we don't need to deploy additional devices and the system is simple and easy to use. Since people carry their personal companion devices all the time and have the full

control of their own devices, thus those devices won't make the users feel intrusive or cause extra money burden. Out of the two companion devices, the watches are normally placed on the wrist. Since the casual moving of arms doesn't have a direct and obvious relationship with ongoing activities, also modern watches are still not powerful enough to do data processing, therefore personal watches have a lot of constraints in detecting one's physical activities. On the contrary, mobile phones are becoming increasingly intelligent and powerful. When they are carried by people in pockets or bags, they are moving with the pace of the human body, thus they appear to be the ideal platforms for detecting people's physical activities such as sitting, walking, running and etc. Modern mobile phones like iPhone or Nokia N97 are embedded with various sensors such as the accelerometer, proximity sensor, magnetometer, GPS and etc. Of all these embedded sensors, the accelerometer is commonly used for activity recognition. Although GPS could detect one's movement in terms of location and speed, it cannot tell the user moves in an accurate manner. In particular, GPS doesn't work inside buildings where people spend most of their time in. Therefore, using the accelerometer-embedded mobile phones to recognize people's physical activities becomes the primary choice among all the solutions.

In this work, we built a smartphone application which tracks users' physical activities and the amount of the calories burn in doing that activity.

Here the smartphone collects the data through the accelerometer present in the phone. An accelerometer is a sensor present in all mobile devices which help to detect whether the phone screen is active or not. With the accelerometer-embedded mobile phone, there are two possibilities to monitor people's physical activities. One is turning the mobile phone as a pedometer, other is recognizing precise physical activities such as walking, running, bicycling, driving and etc.

About 60% of people put their mobile phones in their pockets. With different clothes dressed each day, people are used to putting the mobile phone in different pocket (often the most convenient one). Depending on the position, material and size of the pocket, the mobile phones often have varying orientation, especially when the very pocket swings with human body. Here the accelerometer sensor inside the phone will take the position and orientation associated with the moving pocket. With the varying orientation of the

mobile phone, the experienced force will cause varying effect on the three components of the acceleration signal (x-axis, y axis and z-axis).

We have also referred to a paper based on Otsu's algorithm which help us to give the unbiased results on the activity recognition of the human being, Otsu's algorithm assumes that the image to be thresholded contains two classes of pixels or bi-modal histogram, then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal. This method performs better than the other local and global thresholding methods and produces suitable binary images, which can be used in further processing stages.

Another algorithm which help us to calculate the entropy and gives us the best suited result is ID3 algorithm. ID3 is a simple decision learning algorithm developed by J. Ross Quinlan (1986). ID3 constructs decision tree by employing a top-down, greedy search through the given sets of training data to test each attribute at every node. It uses statistical property call *information gain* to select which attribute to test at each node in the tree. Information gain measures how well a given attribute separates the training examples according to their target classification.

II. Technical Information

As mentioned above, the project functions in two stages: first being the Data Acquisition stage and the next being the Data Processing stage.

1. Data Acquisition

This stage focuses on collecting data through the app, using the phone's hardware sensors. At the press of a button, the app is activated and starts collecting data on the movement of the user. For this a 3D axis accelerometer is employed, that is present on the cell phone. The accelerometer is a hardware sensor that detects motion (acceleration) along the x- , y- and z-axis. The coordinates obtained help us to then figure out the user's current activity.

2. Data Processing

The Decision tree learning algorithm is used to classify an activity into one of the 3 classes– At rest, Walking & Running. This algorithm is being successfully used in expert systems in capturing knowledge. Decision trees classify instances by traverse from root node to leaf node. We start from root node of decision tree, testing the attribute specified by this node, then moving down the tree branch according to the attribute value in the given set. This process is the repeated at the sub-tree level.

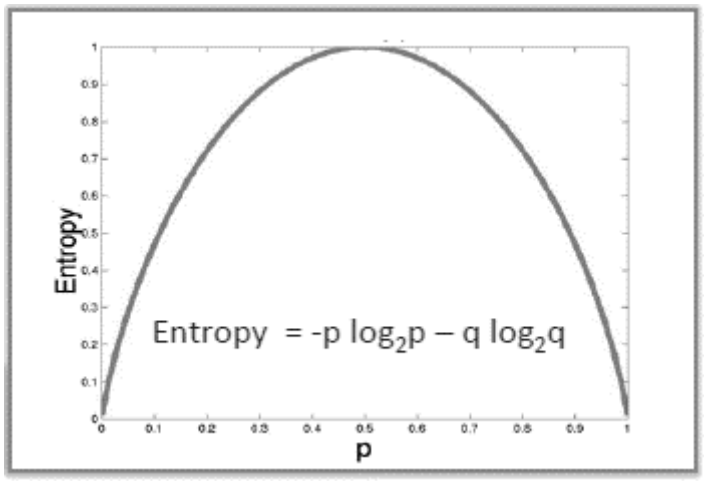
This algorithm is ideal for this project as:

1. Instance is represented as attribute-value pairs. The coordinate values of each activity trace must be continuous-valued data in a specific range which would be subjective to each activity.
2. The training data may contain errors. These can be removed using pruning techniques (which are beyond the scope of this project).
3. The target function has discrete output values. The output, as mentioned above, would be a clear message indicating the activity performed.

The ID3 Basic algorithm is used, where the following steps are performed:

1. Calculate the entropy of every attribute using the training data set S.
2. Split the set S into subsets using the attribute for which entropy is minimum (or, equivalently, information gain is maximum).

3. Make a decision tree node containing that attribute.



Chapter III: System Analysis & Design

I. Software Requirements

- Windows XP (32-bit), Vista (32- or 64-bit) or Windows 7 (32- & 64-bit) /8 OS
- Eclipse 3.6.2 (Helios) or greater
- Eclipse JDT plugin (included in most Eclipse IDE packages)
- JDK 6
- Android Development Tools(ADT) plugin
- Android SDK 3.x
- MATLAB R2011a

II. Hardware Requirements (minimum)

- Android Platform 2.3 (Gingerbread)
- Internal storage 200 MB
- 32 MB RAM
- 200MHz Processor
- 3- axis Accelerometer
- Gyroscope
- 2.7” QVGA display

III. Overall System Description

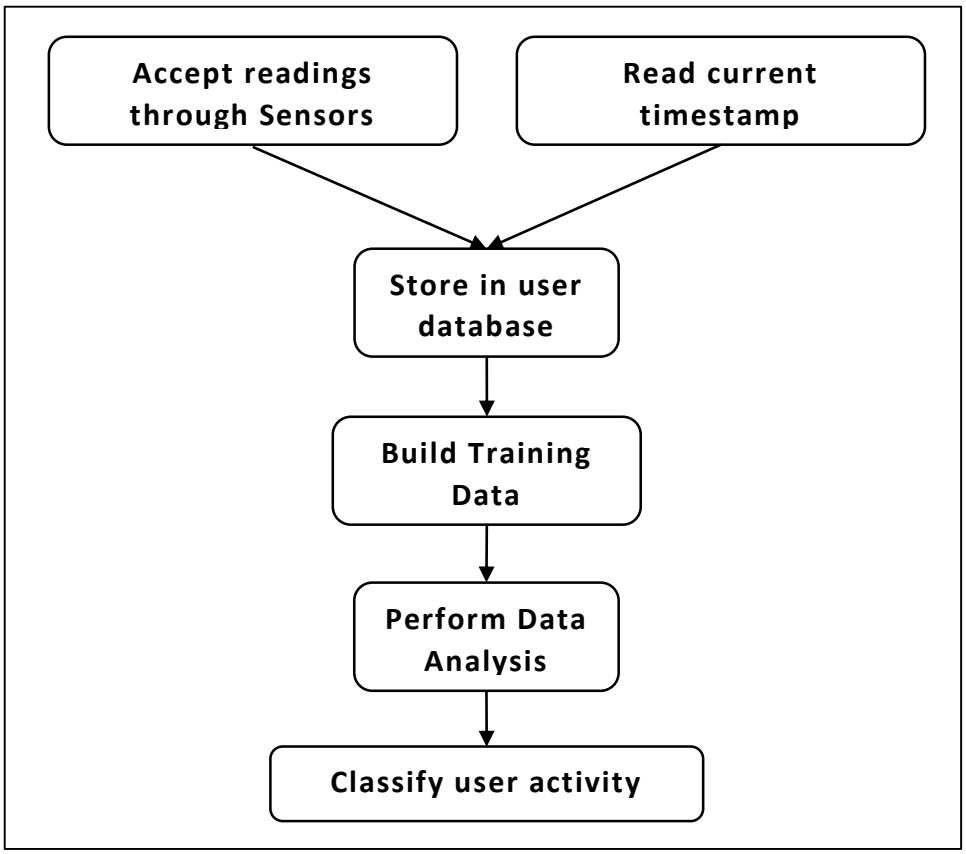


Figure 1: System Design Flowchart

1. System Architecture

This system uses the Model-View-Presenter architecture. It separates the UI concerns between the data of the UI (Model), the display of the UI (View), and the logic of the UI (Presenter). The MVP pattern is similar to the MVC (Model-View-Controller) pattern, except that in MVP, the presenter contains the UI business logic for the view and communicates with it through an interface.

- Model

  Models represent knowledge. A model could be a single object, or it could be some structure of objects. They can also represent activities or resources. For this app, we have created a database which stores the accelerometer readings of the user. We also have various resources that our defined in XML files by locale. The Android resource system keeps track of all non-code assets associated with an application.

A *res* folder keeps track of all these resources. These include, say, the logo and buttons used in the system. It also contains a layout folder that links to all the XML files (see View).

- View

  A view is a (visual) representation of its model. It would ordinarily highlight certain attributes of the model and suppress others. The User Interface of the app is displayed through the XML files under the *layout* folder. Along with containing all the UI elements, View also handles events for these views.

- Presenter

  Presenters handle the communication between the model and the view. Acting as a gateway to the model, its job is to query the model and then update the view accordingly. In this app, the presenter would have two functions: to add the accelerometer readings to the database and also to classify the activity performed by the user. For the first part, it uses the *DatabaseManager.java* file while for the second it uses the *AndroidActivity.java* file.

Using the MVP pattern makes our code more organized and easier to test. Testing in Android can be tricky as most of the code ends up in Activities. MVP helps by creating test cases and apply test-driven development in an easy way.

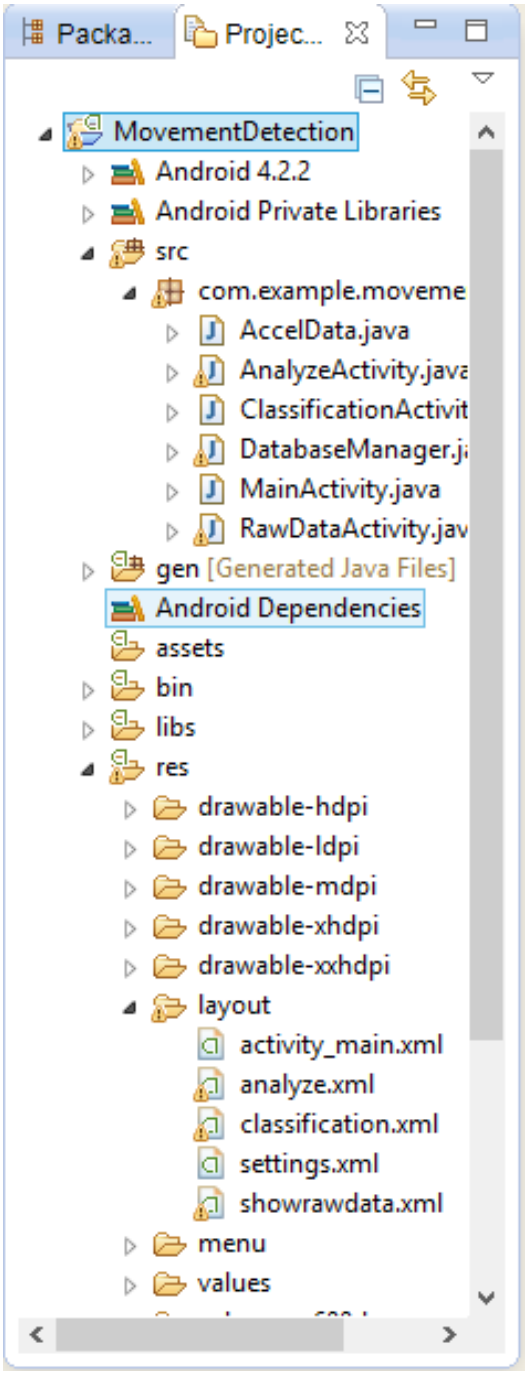


Figure 2: Project Structure

2. Workflow Diagram

Since an activity diagram would not cover the event-control mechanisms or display the organization of components in the app, we have instead used a workflow diagram. The adjoining workflow diagram serves as an adjunct to the system architecture. It can be easily seen how the app components fit the M-V-P architectural pattern.

The workflow also gives a clear picture on the business logic of the app.

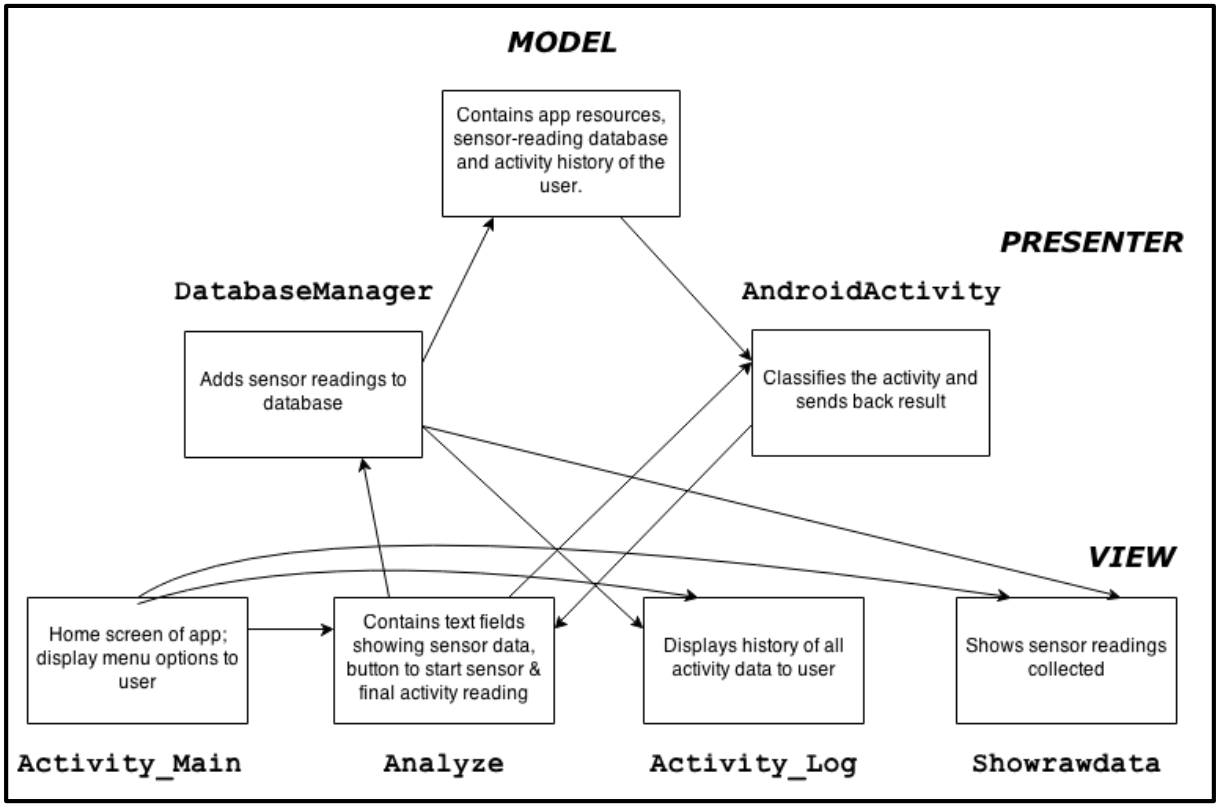


Figure 3: Workflow Diagram

3. Activity Diagram

It gives an idea of the sequence of activities that the user will have to perform while accessing the app.

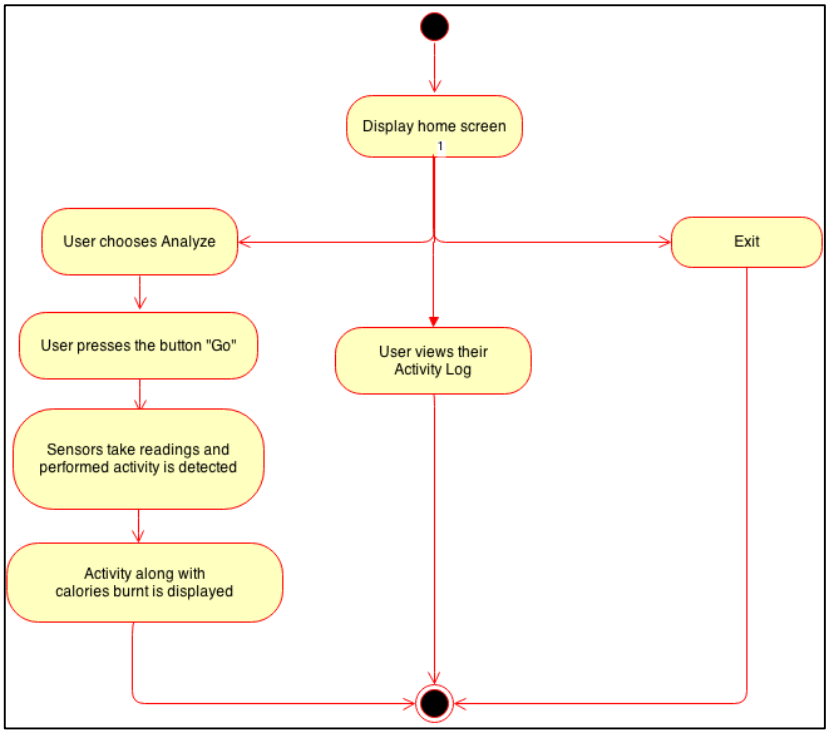


Figure 4: Activity Diagram

4. Use Case Diagram

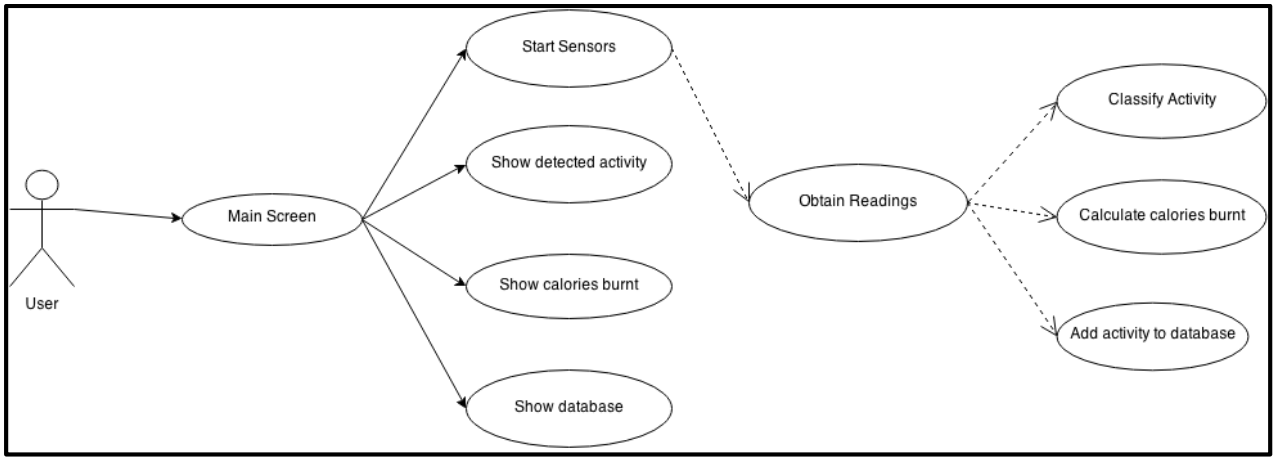


Figure 5: Use Case Diagram

Since Android only allows a single user to be logged in to the OS, therefore no login screen is included. Through the main menu the user can access various options. At the *Start Sensors* state, the user would press a button and the accelerometer will start taking readings. The user will then be shown what activity was detected based on these readings and the calories burnt in it, through the *Show Detected Activity* and the *Show Calories Burnt* state. He/she also has access to a database called the "Activity Log", in which a history of the user's activities will be maintained.

IV. Overview of the modules and the subsystems

1. Accept readings through sensors

Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data with high precision and accuracy, and are useful if you want to monitor three-dimensional device movement or positioning, or you want to monitor changes in the ambient environment near a device.

The Android platform has three kinds of sensors- Motion sensors, Environmental sensors and Position sensors. Motion sensors are useful for monitoring device movement, such as tilt, shake, rotation, or swing. In this app, we only require use of this sensor.
The SensorEventListener interface is implemented in our app's java file, called AnalyzeActivity.java. This interface is used for receiving notifications from the SensorManager when sensor values have changed.

SensorManager lets us access the device's sensors. An instance of this class is received by calling *Context.getSystemService()* with the argument SENSOR_SERVICE. It is important to note that the sensors are not disabled even if the screen is turned off.

According to the Android Developer's website, an acceleration sensor determines the acceleration that is applied to a device (Ad) by measuring the forces that are applied to the sensor itself (F_s) using the following relationship:

$$A_d = - \sum F_s / \text{mass}$$

However, the force of gravity is always influencing the measured acceleration according to the following relationship:

$$A_d = -g - \sum F / \text{mass}$$

For this reason, when the device is sitting on a table (and not accelerating), the accelerometer reads a magnitude of $g = 9.81\ m/s^2$. Similarly, when the device is in free fall and therefore rapidly accelerating toward the ground at $9.81\ m/s^2$, its accelerometer reads a magnitude of $g = 0\ m/s^2$. Therefore, to measure the real acceleration of the device, the contribution of the force of gravity must be removed from the accelerometer data.

This can be achieved by applying a high-pass filter. Conversely, a low-pass filter can be used to isolate the force of gravity.

To do this, we used the following method:

```
public float[] highPass(float x, float y, float z)
{

	float[] filteredValues = new float[3];
	float gravity[] = new float[3];
	final float ALPHA = (float) 0.8;
	gravity[0] = ALPHA * gravity[0] + (1 - ALPHA) * x;
	gravity[1] = ALPHA * gravity[1] + (1 - ALPHA) * y;
	gravity[2] = ALPHA * gravity[2] + (1 - ALPHA) * z;

	filteredValues[0] = x - gravity[0];
	filteredValues[1] = y - gravity[1];
	filteredValues[2] = z - gravity[2];

	return filteredValues;

}
```

2. Store in database

The APIs needed to use a database on Android are available in the android.database.sqlite package. One of the main principles of SQL databases is the schema: a formal declaration of how the database is organized. The schema is reflected in the SQL statements that you use to create your database.

To create the database tables, the following code was used:

```
public void onCreate(SQLiteDatabase db) {
		// TODO Auto-generated method stub
		db.execSQL("CREATE TABLE UserMovement(xvalue float not null,
yvalue float not null,zvalue float not null);");
		db.execSQL("CREATE TABLE UserActivityLog(action text not
null);");
	}
```

Just like files that you save on the device's internal storage, Android stores your database in private disk space that's associated application. The data is secure, because by default this area is not accessible to other applications. The sensor data of the user obtained in subsystem 1 along with the current timestamp is taken and stored into a database and used for processing.

3. Build training data

As mentioned in [1], we accept movement data of the user through the accelerometer. There are three classes to which the user's activity can belong:

- *At Rest*: when the user is stationary.

- *Walking*: when the user is walking.
- *Running*: when the user is running.

Before classifying the data, it is important to first find the patterns and differences in data for each of these activities. This is done by first building a training set. Training refers to the process of applying a specific mathematical algorithm to the data in the structure in order to extract patterns. Using a training set is an important part of evaluating any data mining model.

In this project, a training set of about 900 activity traces was used. The user uses the app by pressing the button "Go" on the Analyse Activity screen. This makes the accelerometer sensors active and movements along the x-axis, y-axis and z-axis are sensed. These movements are then logged into a database, along with the current timestamp. To build the training set, we took in approximately 300 values along all axes, for all the above classes, i.e. At Rest, Walking and Running.

For these values, the *mean* and *standard deviation* for each axis for every activity was calculated. Calculating the mean and standard deviation is necessary to obtain the probability of each of these events.

Once the calculations were made, graphs were plotted on Matlab to obtain probability distribution graphs for all axes. The coordinate values for both Walking and Running take the form of a data series with slight patterns, whereas those for "At Rest" remain more or less constant (as seen in Section V).

V. Diagrams/Tables

Figure 6: Activity Recognition Process Pipeline

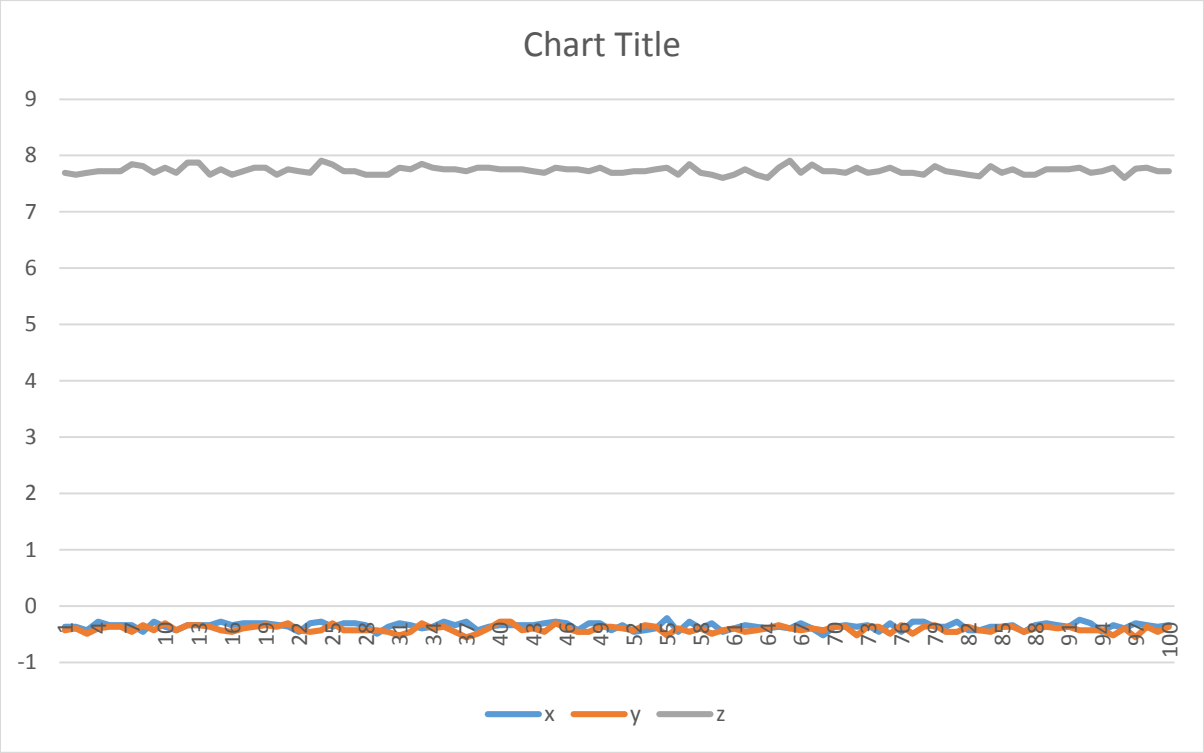


Figure 7: Variation in x-, y- & z-coordinate values during the At Rest activity

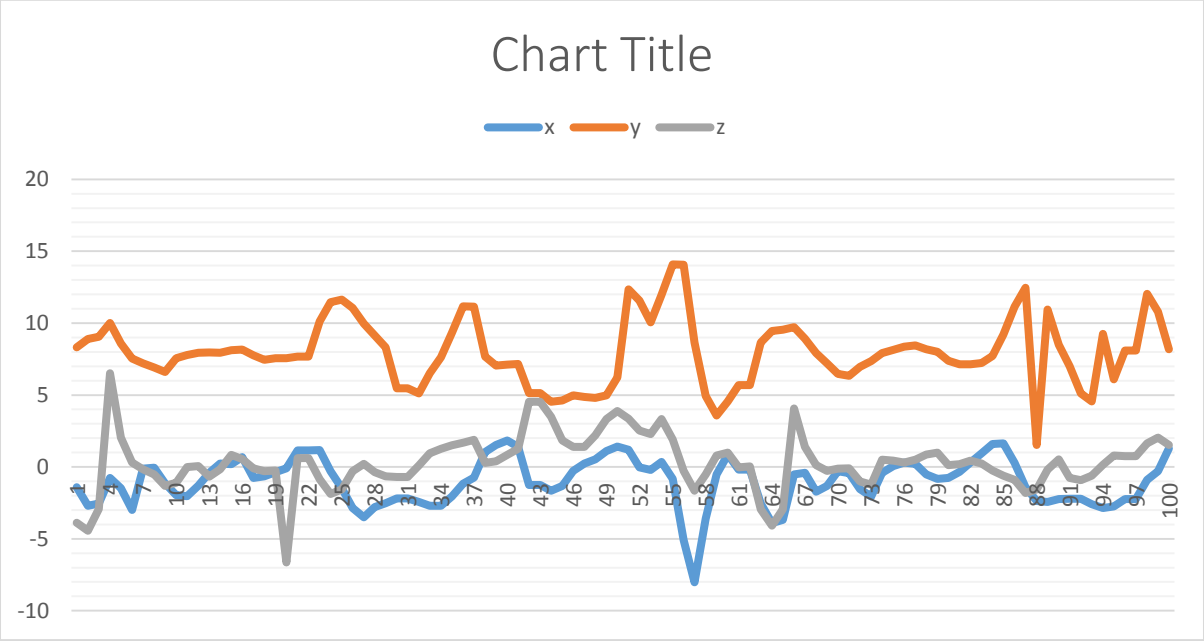


Figure 8: Variation in x-, y- & z-coordinate values during the Walking activity

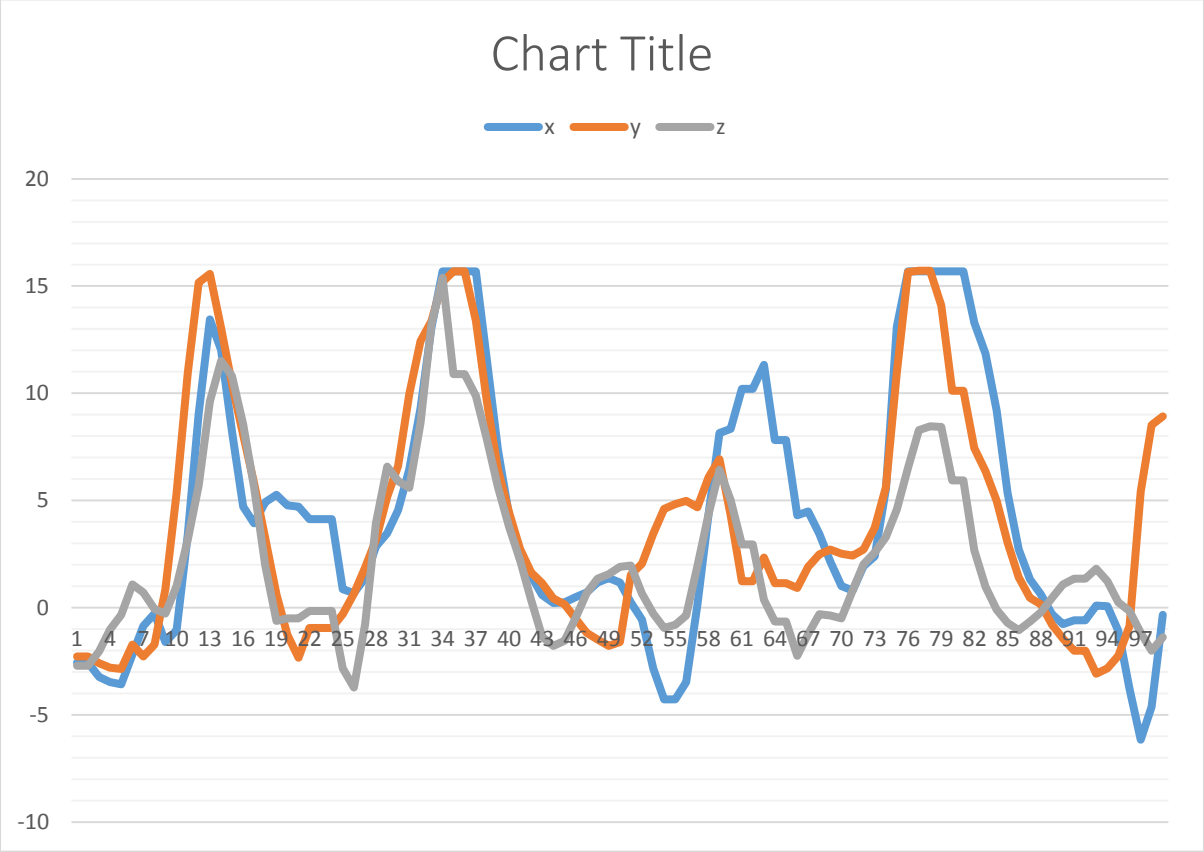


Figure 9: Variation in x-, y- & z-coordinate values during the Running activity

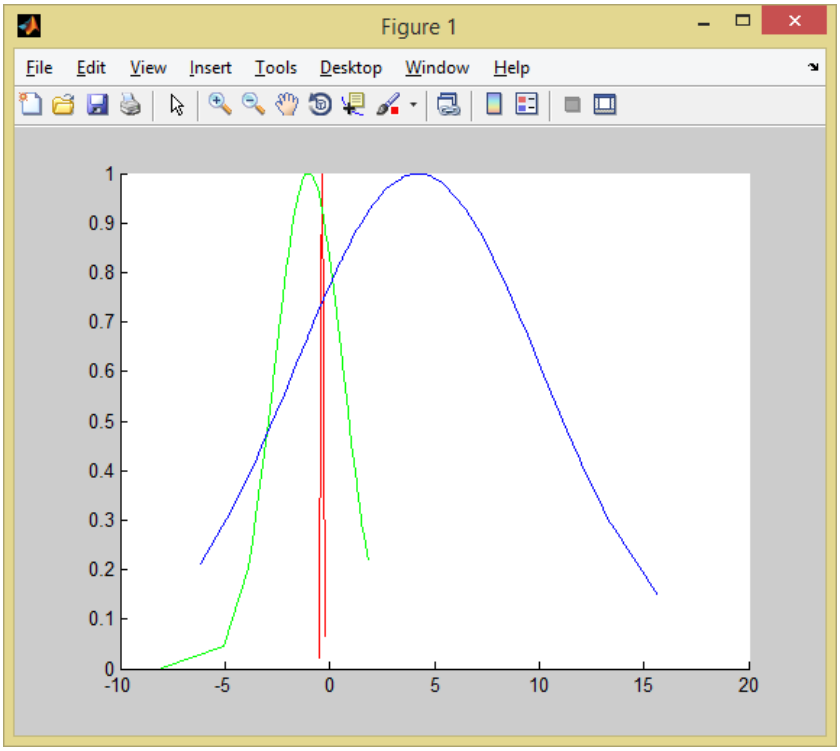


Figure 10: P.d.f graph for x-coordinate values, where red means At Rest, green means Walking & blue means Running

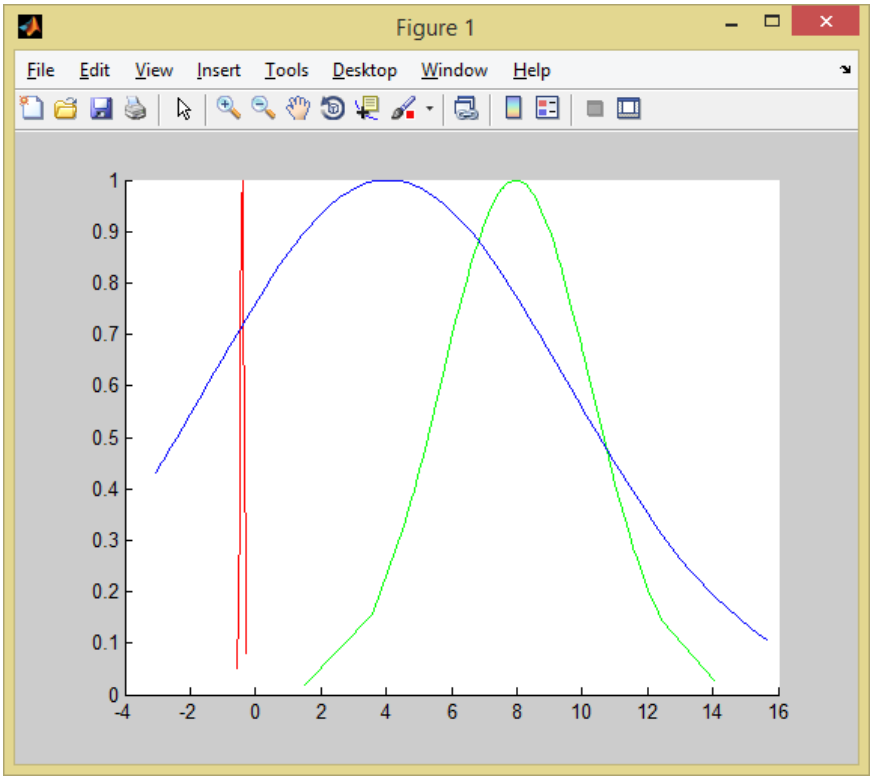


Figure 11: P.d.f graph for y-coordinate values, where red means At Rest, green means Walking & blue means Running

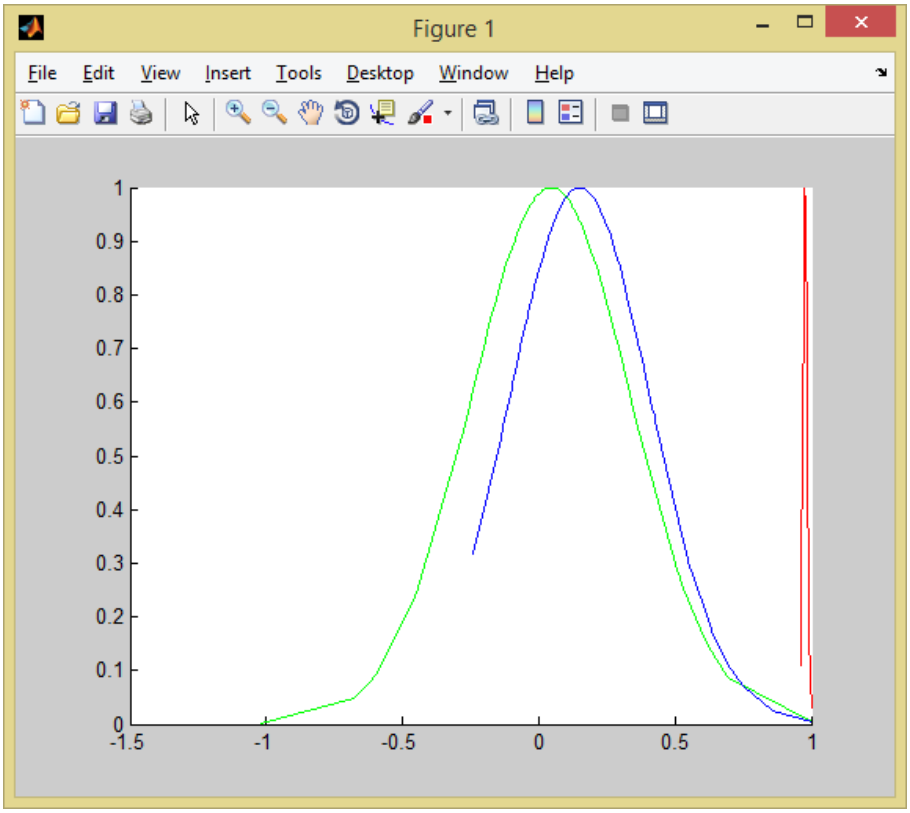


Figure 12: P.d.f graph for z-coordinate values, where red means At Rest, green means Walking & blue means Running

VI. Algorithm Used

Entropy is a measure of unpredictability of information content. As said, for a random variable X with n outcomes $\{x_1, ..., x_n\}$, the entropy, a measure of uncertainty and denoted by H(X), is defined as

$$H(X) = -\sum_{i=1}^{n} p(xi) \log_b p(xi)$$

Where $p(x_i)$ is the probability mass function of outcome x_i.

To obtain the value of the probability distribution function (p.d.f.), we use the results of the graph created on Matlab for each of the axes. Thresholding of the p.d.f. is done to find the probability function for each of the coordinate. Using Otsu's algorithm, we calculate the p.d.f. of a particular coordinate value as that of the activity with the highest value.

Once the probability function has been calculated, it becomes rather straightforward to obtain the entropy. For this project, the objective would be to select the activity that minimises the value of entropy, thus maximising the information gain.

Entropy has the maximum value when the instances (axes values) are equally distributed among each of the 3 activities. In this case the pi is $1/3$, which would be independent of i.

$$E = -\sum_{i=1}^{3} \left(\frac{1}{3}\right) \log_2 \left(\frac{1}{3}\right) = -3 \left(\frac{1}{3}\right) \log_2 \left(\frac{1}{3}\right) = -\log_2 \left(\frac{1}{3}\right) = \log_2 3$$

Hence value of entropy in this case would be 1.5850.

In a similar manner, entropy for each activity trace is calculated using pi from the probability distribution function generated in Matlab.

A training data set of about 900 activity traces is taken. To calculate the entropy of the training data, following steps were followed:

Step 1:

Calculate the range of values of all coordinate with respect to each activity.

At Rest:

	Maximum	**Minimum**
x-value	-0.21536174	-0.52302134
y-value	-0.27689368	-0.55378735

z-value	7.9068527	7.599193

Walking:

	Maximum	**Minimum**
x-value	1.8459578	-8.029917
y-value	14.090811	1.521748
z-value	6.5223837	-6.646085

Running:

	Maximum	**Minimum**
x-value	15.69064	-6.1531925
y-value	15.721407	-3.0765963
z-value	15.382982	-3.7226815

Step 2:

Calculate the probability of every activity for each axis. For our training set, we calculated them as follows (where P_r refers to probability of running, P_w to that of walking & P_{ar} to that of At Rest):

x-axis:	
$P_r(x)$	0.905156
$P_w(x)$	0.0984
$P_{ar}(x)$	0.00103
y-axis:	
$P_r(y)$	0.53687
$P_w(y)$	0.2014
$P_{ar}(y)$	0.00923
z-axis:	
$P_r(z)$	0.2584
$P_w(z)$	0.72604
$P_{ar}(z)$	0.01556

Step 3:

Group all activities for each axis. (E.g. group the probabilities of x-coordinate values for at rest, Walking & Running together)

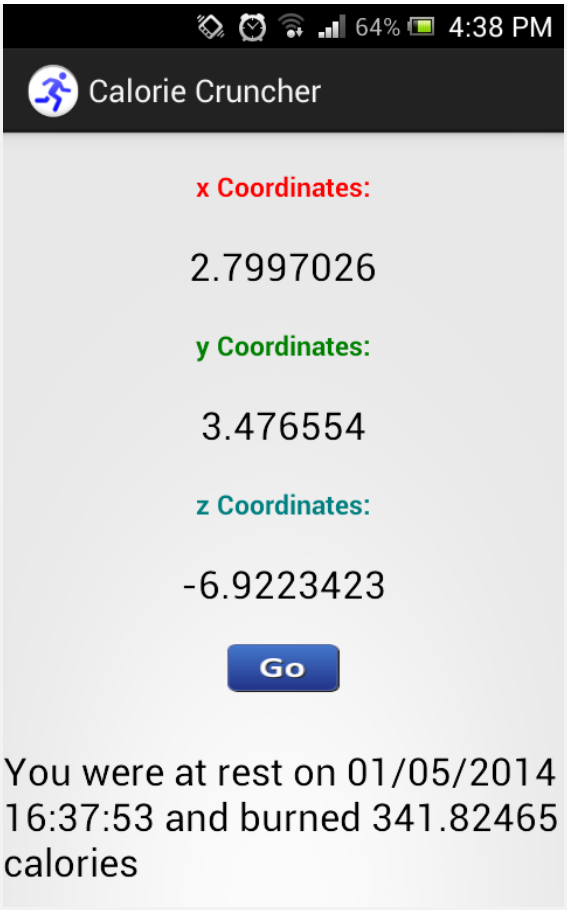
Step 4:

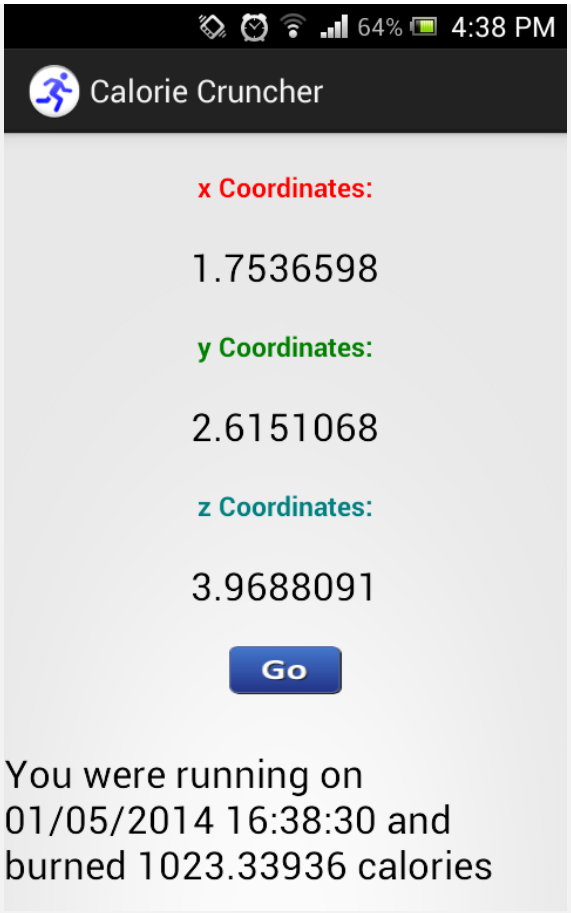
Calculate the entropy using the entropy function mentioned in Section VI. For our training set, the values are:

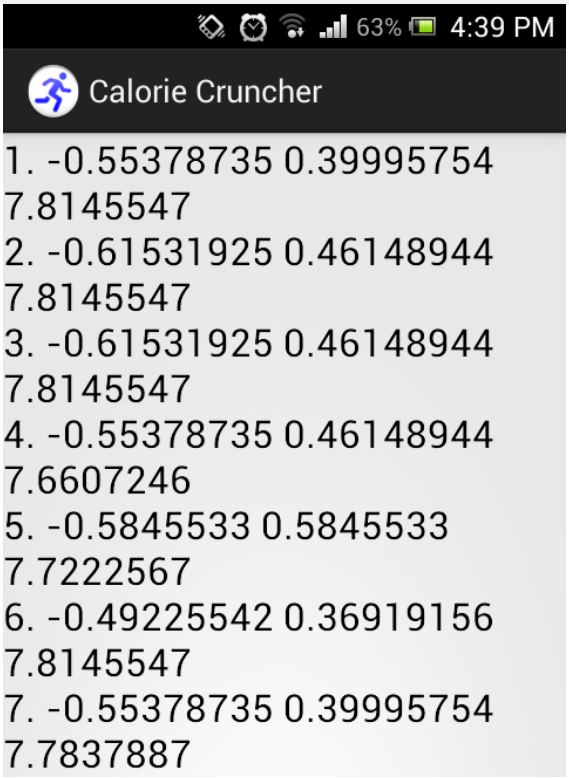
At Rest	
x – axis	0.0071
y – axis	0.0432
z – axis	0.0648
Walking	
x – axis	0.2282
y – axis	0.3227
z – axis	0.2324
Running	
x – axis	0.0934
y – axis	0.3339
z – axis	0.3497

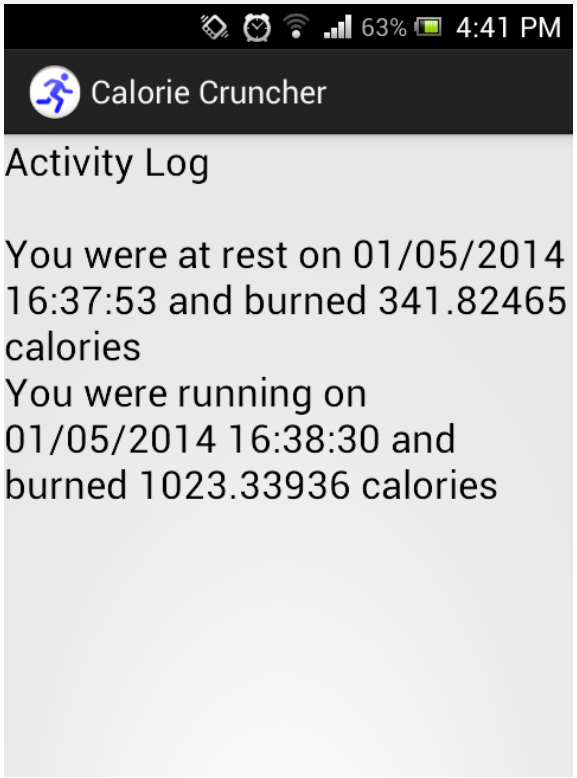
Step 5:

After receiving the current value from the app, find the difference between it & each of the above nine values. The value giving us the minimum difference between them will give us the corresponding activity being performed by the user.


64% 4:38 PM
Calorie Cruncher
x Coordinates:
2.7997026
y Coordinates:
3.476554
z Coordinates:
-6.9223423
Go
You were at rest on 01/05/2014 16:37:53 and burned 341.82465 calories


64% 4:38 PM
Calorie Cruncher
x Coordinates:
1.7536598
y Coordinates:
2.6151068
z Coordinates:
3.9688091
Go
You were running on 01/05/2014 16:38:30 and burned 1023.33936 calories


63% 4:39 PM
Calorie Cruncher
1. -0.55378735 0.39995754 7.8145547
2. -0.61531925 0.46148944 7.8145547
3. -0.61531925 0.46148944 7.8145547
4. -0.55378735 0.46148944 7.6607246
5. -0.5845533 0.5845533 7.7222567
6. -0.49225542 0.36919156 7.8145547
7. -0.55378735 0.39995754 7.7837887


63% 4:41 PM
Calorie Cruncher
Activity Log
You were at rest on 01/05/2014 16:37:53 and burned 341.82465 calories
You were running on 01/05/2014 16:38:30 and burned 1023.33936 calories

VII. Test Cases

Testing at an early stage was promoted for every part of the project, before the full set of requirements have been defined and the coding process has been started. Unit testing was commonly used throughout the app.

A unit test is a piece of code written by a developer that executes a specific functionality in the code to be tested. The percentage of code which is tested by unit tests is typically called test coverage.

A unit test targets a small unit of code, e.g., a method or a class, (local tests). Unit tests ensure that code works as intended. They are also very helpful to ensure that the code still works as intended in case we need to modify code for fixing a bug or extending functionality. Having a high test coverage of our code allows us to continue developing features without having to perform lots of manual tests.

Automated testing of Android applications is especially important because of the huge variety of available devices. As it is not possible to test Android application on all possible device configurations, it is common practice to run Android test on typical device configurations.

Android testing is based on JUnit. Testing for Android can be classified into tests which require only the JVM and tests which require the Android system.

Following things need to be tested in the app:

- Activity life cycle and events
  We should test if the activity handles the Android life cycle events correctly.
- File system and database operations
  Write and read access from and to the file system should be tested including the handling of databases.
- Different device configurations
  We should also test if your application behaves well on different device configurations.

JUnit is fully supported by Eclipse and the Android ADT plugin lets you create Android testing projects. Furthermore, you can run the tests and analyse the results without

leaving the IDE. This also provides a more subtle advantage; being able to run the tests from Eclipse allows you to debug the tests that are not behaving correctly.

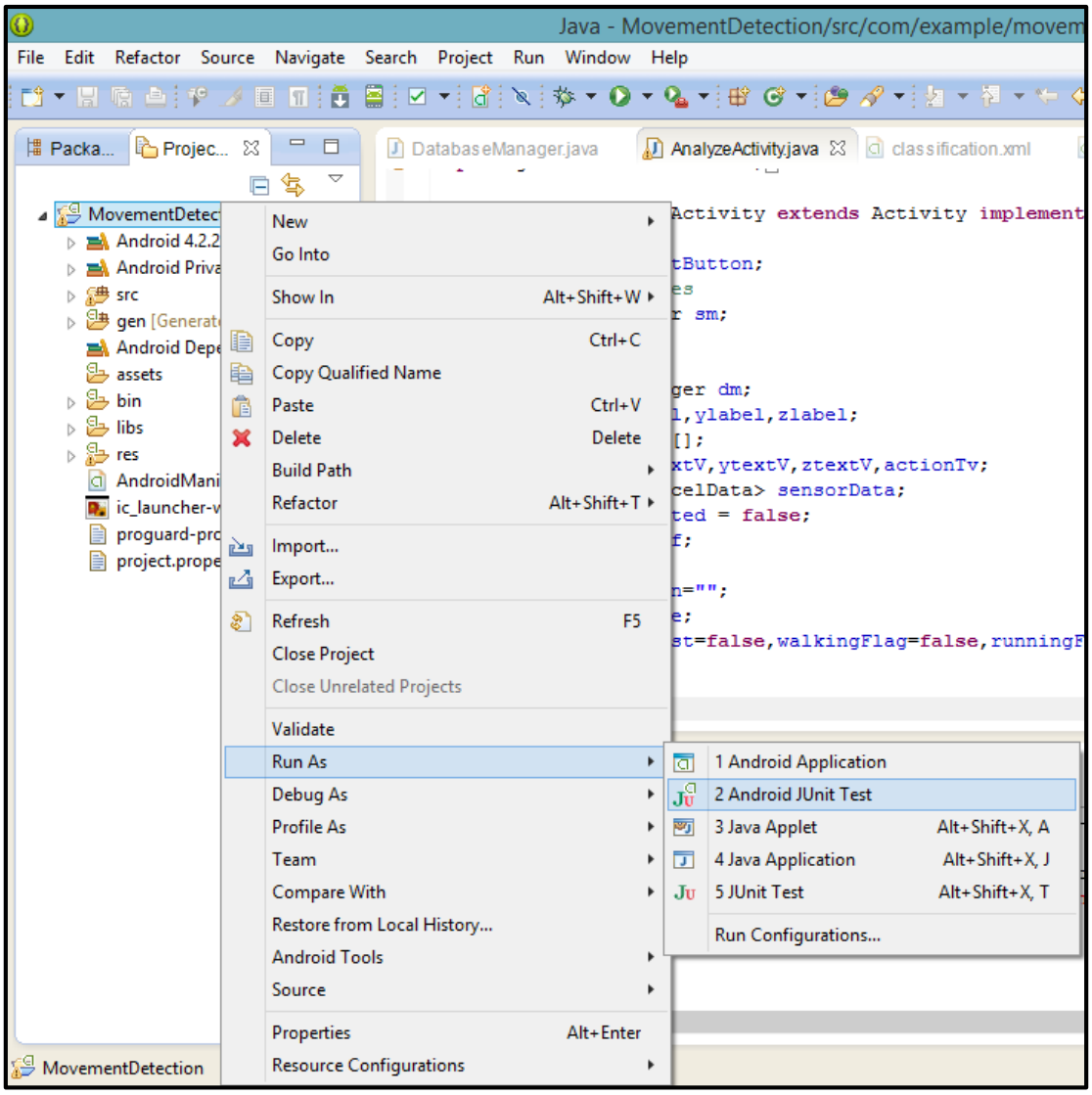


Figure 13: Running JUnit Test

The preferred way of organizing tests is to keep them in separate Android test projects or source folders. The Android tooling for Eclipse emphasizes the usage of separate projects.

Instead of Android components, an Android test application contains one or more test classes.

The project under test must be added as dependency to the test project. The AndroidManifest.xml file must also specify that the test project uses the *android.test.runner* library and specifies the test runner for the unit test.

The Android testing environment includes an instrumentation framework that lets tests control and examine the application. The instrumentation framework is the foundation of the testing framework. Instrumentation controls the application under test and permits the injection of mock components required by the application to run.

A test project also specifies the package of the application to test in the AndroidManifest.xml file under the *android:target Package* attribute. The following figure shows a part of AndroidManifest.xml for our project.

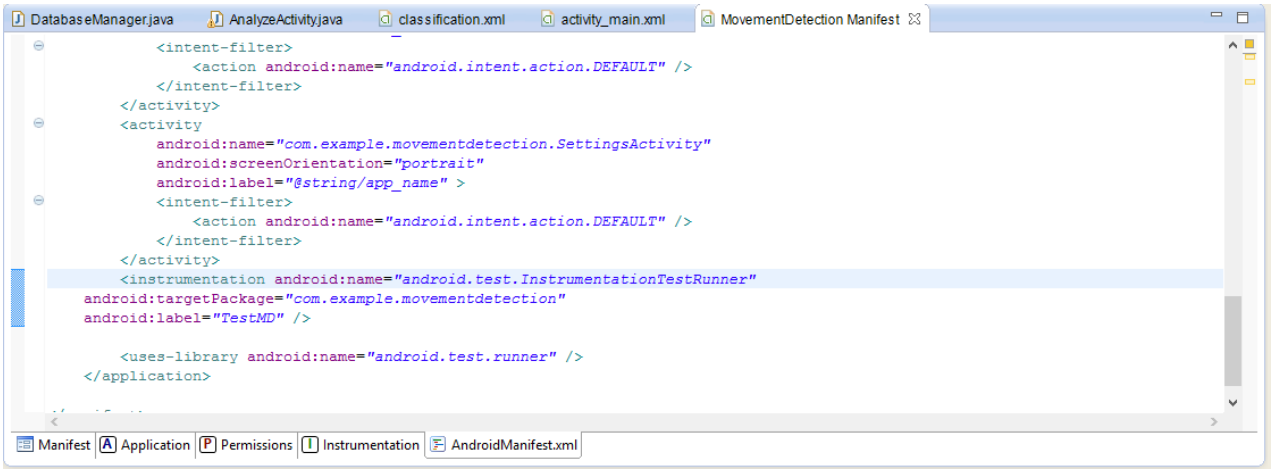


Figure 14: Android Manifest

To start the test from Eclipse, we right-click on the test class and select Run As → Android JUnit Test. The emulator runs as shown:

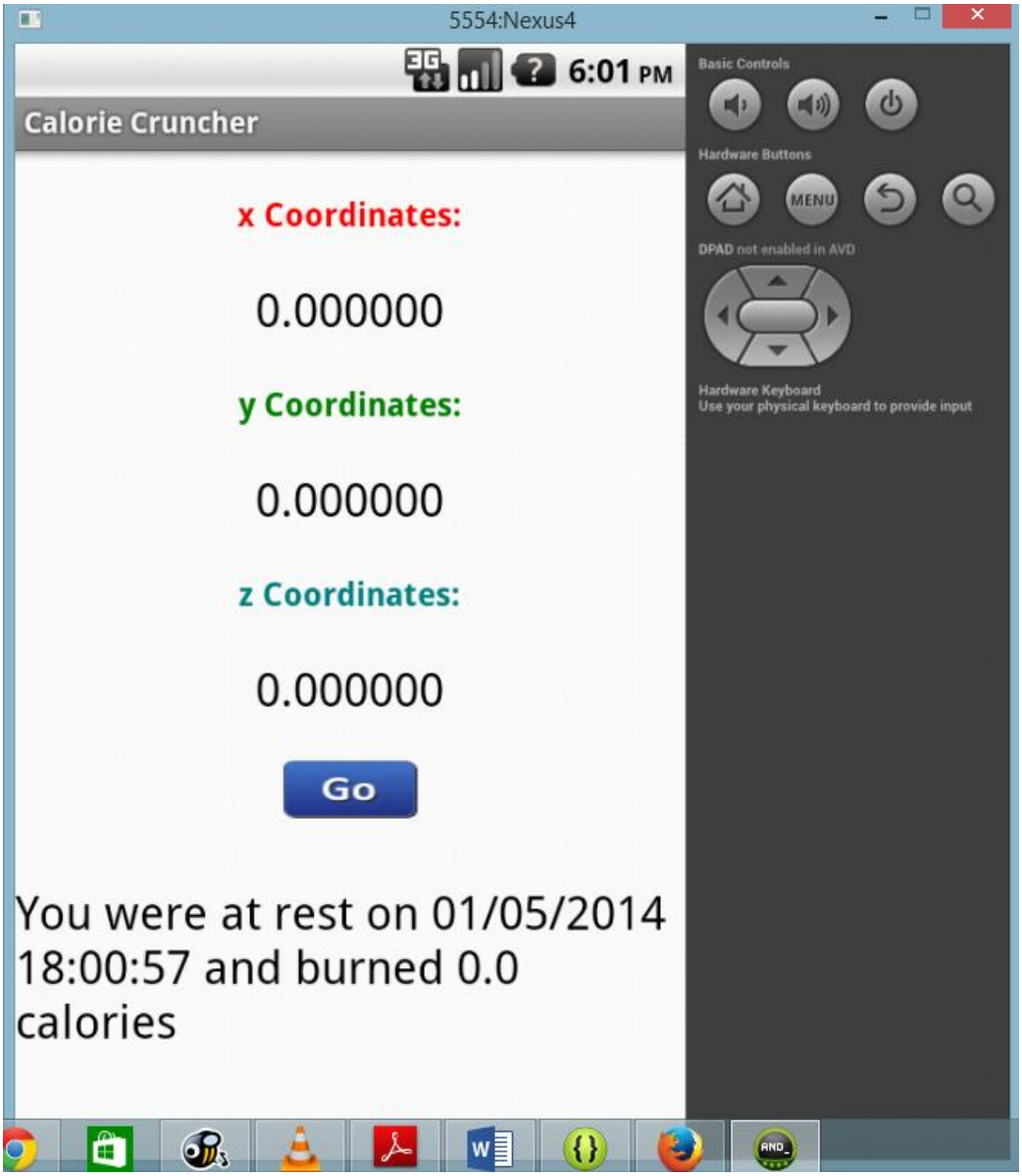


Figure 15: Emulator screenshot

Obviously, since the app is running on the desktop, it does not have access to the accelerometer. Therefore, the readings shown will be incorrect.

Finally, special consideration should be taken if your tests involve UI components. Only the main thread is allowed to alter the UI in Android. Thus a special annotation *@UIThreadTest* is used to indicate that a particular test should be run on that thread and would have the ability to alter the UI.

Conclusion

Through this project, we explored one of the many ways that the sensors in a smartphone can be used as a tool by the users to get feedback about their activities and fitness levels. The app is efficient and intuitive. It allows the device to serve the user without the user having to spend time programming the device.

The app is different from other apps in the following manner: (1) It requires no additional hardware setup & relies only on the physical sensors that can be found even on low-end smartphones (2) It does not require any calibration (3) It can detect 3 different kinds of activities- walking, running & staying at rest (say, while standing or sleeping).

So far, this project works on the assumption that people carry their cell phones for the majority of the day. It also assumes that the phone is usually in their hand or their pocket. We expect that with the introduction of "wearable" gadgets and smart-watches that are increasingly becoming popular, the app would be able to give more accurate results. Also, we would like to add a function that would use the historical data of a user to give suggestions & information about leading a more active lifestyle.

We are currently working on making the app as unobtrusively as possible, so that data will be collected without the user having to inform the app when they go to sleep.

References

1. "*Activity Recognition Using Smartphone Sensors*" – by *Alvina Anjum*, *Muhammad U. Ilyas*, Published in – "People Centric Sensing and Communications 2013".
2. "*Activity recognition from accelerometer data*" – by *Ravi, N., D, N., Mysore, P., Littman, M.L.* , Published - In Proceedings of the Seventeenth Conference on Innovative Applications of Artificial Intelligence (IAAI), AAAI Press (2005)
3. "*Oracle® Data Mining 11g Release 1 (11.1) B28129-04*" – by *Oracle®*, Dated May 2008.
4. "*Classification Algorithms in Human Activity Recognition using Smartphones*" – by *Mohd Fikri Azli bin Abdullah, Ali Fahmi Perwira Negara, Md. Shohel Sayeed, Deok-Jai Choi, Kalaiarasi Sonai Muthu* Published - International Journal of Computer and Information Engineering Dated June 2012
5. "*Pattern Recognition and Machine Learning*" – by *Christopher Bishop*, Published by *Springer* Date- February 1, 2010 ISBN - *978-0387310732*.
6. "*Training and Testing Data Sets*" - http://technet.microsoft.com/en-us/library/bb895173.aspx
7. "*Saving Data in SQL Databases*" - http://developer.android.com/training/basics/data-storage/databases.html
8. "*Activity Recognition on an Accelerometer Embedded Mobile Phone with Varying Positions and Orientations"* – Lin Sun, Daqing Zhang, Bin Li, Bin Guo, Shijian Li