Elevation Android App

CS 6365 Project Report

Elevation App is a Light-Weight Android Application that leverages the micro-sensors on a smart phone viz., Accelerometer to determine the gross and net change in elevation of the phone from a starting point. https://github.com/smanchem/Elevation

Sandeep Manchem

4/25/2014

Contents

Introduction 3

Motivation 3

Objective 3

Approach 4

Algorithm 4

Observations 6

Elevation Android App 6

Fig 1: Start Screen 7

Fig 4: Data being Gathered Fig 5: Results after Pause/Stop 8

Fig 6: Logs Section Fig 7: Exit Button (Settings Button) 9

Evaluation 9

Fig 8: Going from 1st floor to 6th floor (18 m) 10

Fig 9: Going from 6th floor to 1st floor (18 m) 11

Related Work 11

Lessons Learned 12

Future Work 12

References 12

Appendix 12

Introduction

The ubiquitous and all pervasive Smartphones have become a part and parcel of our current lifestyle. One of the key reasons for their success apart from internet connectivity and cloud support is that they are equipped with a myriad of micro-sensors that redefine "*All in one*". Two such crucial micro-sensors are the *Accelerometer* and the *Gyroscope*. So far these two sensors have been used primarily for games and motion detection. There have been attempts to use them for a wider range of applications but few have succeeded owing to the noisy data they generate. This project attempts to develop filters for the Accelerometer data in order to compute the gross ascent and descent of the Smartphone from a starting point.

Motivation

Several mobile applications are available that enable a user to compute their jogging/hiking statistics viz. distance covered, average speed, maximum speed, route taken and so on. But very few apps show you the elevation achieved during the jog/hike, the gross ascent and descent, which I believe are useful pieces of information for a healthy workout session and essential to compute calories burned. The ones that do provide elevation information do it using GPS location information which does not work indoors and which is not accurate for short distances. The motivation behind building the Elevation app is to leverage the built-in *Accelerometer* to compute the gross and net ascent and descent during a jogging/hiking session without relying on GPS and at the same understand the true behavior of these micro-sensors and evaluate their reliability.

Objective

The purpose of the project is to understand the behavior of the *Accelerometer* micro-sensor in Smartphones and evaluate the accuracy and reliability of its readings. Develop filters for the *Accelerometer* data from a Smartphone and use that to compute the elevation achieved by the device starting from an initial point of origin. Subsequently develop this into a Mobile Application on the Android Platform that computes the gross ascent and descent achieved by the user during an activity viz. Jogging/Hiking. The raw sensor information is expected to be heavy, depending on

the sampling rate, and at the same time useful for analysis and hence can be stored in a database to be queried later on.

Approach

The approach adapted to achieve the objective in this project is to first gather the raw data from the Accelerometer Sensors in a Smart Phone and analyze it. Upon analysis, develop filters to reduce noise and improve the accuracy of the computation of elevation.

With reasonably accurate algorithm and filters in place, develop an Android Application that computes the gross ascent and descent of the user while performing day to day activities.

Algorithm

The naïve algorithm to compute distance from acceleration is to integrate the data once to obtain velocity and once again to obtain distance. But this approach quadruples any error in measuring the acceleration. We already know that the micro-sensors, especially the Accelerometer Sensor in Smartphones, are prone to inaccuracies and generate a lot of noise. So the naïve algorithm is useless.

Next we move onto an approximation approach by increasing the sampling rate and using the algorithm below:

```
Velocity = 0

Distance = 0

Update_velocity (Acceleration) {

      Velocity = Velocity + Acceleration;

}

Update_distance (Velocity) {

      Distance = Distance + Velocity;

}
```

This algorithm upon implementation and testing proved to be unreliable and laden with erroneous updates. Upon careful study of the data generated by the algorithm it was clear that the Acceleration values themselves were varying to a great degree (noise). Hence, it became imperative to develop a filter before optimizing the Distance computation algorithm.

We apply a Filter Parameter, alpha, to the raw sensor values using the following formula:

```
gravity = alpha * gravity + (1 - alpha) * acceleration;
linear_acceleration = acceleration - gravity;
```

The Acceleration due to Gravity component is subtracted from the sensor reading $(gravity = 9.81\ m/s^2)$, to obtain the linear acceleration of the phone. The value of alpha is chosen based on the empirical observations that when the phone is at rest on a table top, the readings of linear acceleration should be zero. So we arrive at a suitable value for alpha and compute the linear acceleration.

Then we use the following algorithm to compute the distance moved by phone using the linear acceleration readings:

// z is the linear acceleration read every 10 milliseconds, hence needs to be multiplied by 0.01

```
double acc = z*0.01;
if (z > 0.1 || z < -0.1) {
     velocity = velocity + acc;
     if (velocity > 0.15) {
          ascent = ascent + (velocity * duration * 0.001);
     } else if (velocity < -0.1){
          descent = descent + (velocity * duration * 0.001);
     }
} else if (velocity > 0.15) {
     asccent = ascent + (velocity * duration * 0.001);
} else if (velocity < -0.1){
     descent = descent + (velocity * duration * 0.001);
}

prevTime = currentTime;
```

`(velocity * duration)` is the distance travelled in the duration. Since the duration (time between two actual readings) is measured in milliseconds, we need to multiply it by a factor of 0.001 to get it to seconds.

The numbers we compare "`z`" and "`velocity`" with, in the `if()` cases were arrived empirically based on observations. The experiment used to arrive at these numbers was going up and down in an elevator.

Observations

The position of the phone is crucial in determining the accuracy of the readings. The readings are most accurate when the phone is lying flat with the screen upwards. If it is in another position, the readings are wild.

The accelerometer readings vary from phone to phone. So the resulting android app cannot be portable unless a calibration tool is integrating into the App to determine "alpha" at installation time or before running it each time.

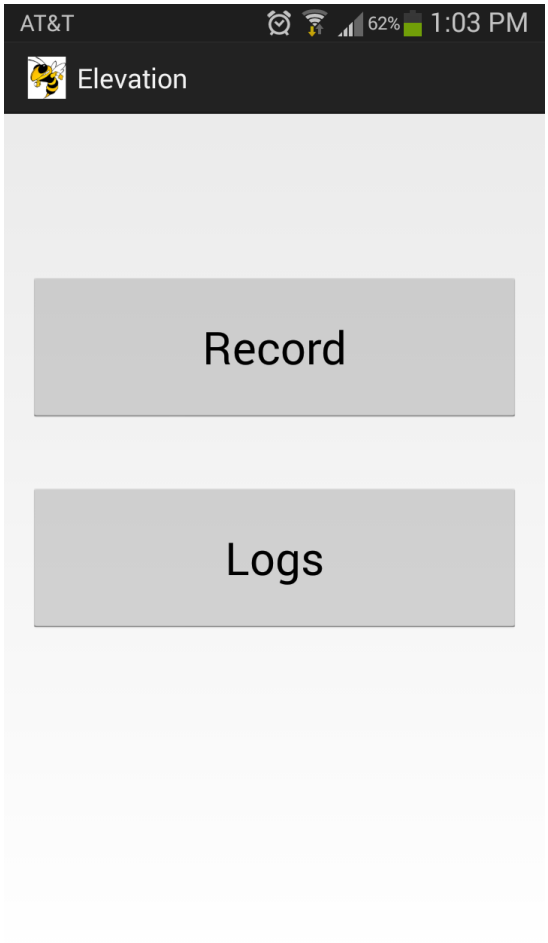
Vibrations also result in changes in the readings of accelerometer. And hence, the phone has to be steady for accurate readings.

The sensor values persist even after apparent change in the direction and magnitude of the force being applied on the phone. That is, if the direction of motion of the phone is reversed, the sensors values won't toggle sign immediately.

Higher Sampling Rate doesn't necessarily result in higher accuracy due to the above mentioned persistence of readings. A slight delay in reading the sensor helps in stabilizing the values.

Elevation Android App

The Elevation Android App is simply implements the algorithm described above and adds a method of determining ascent and descent for long distances using GPS Coordinates. The Architecture and Design of the app is explained using the following screenshots:



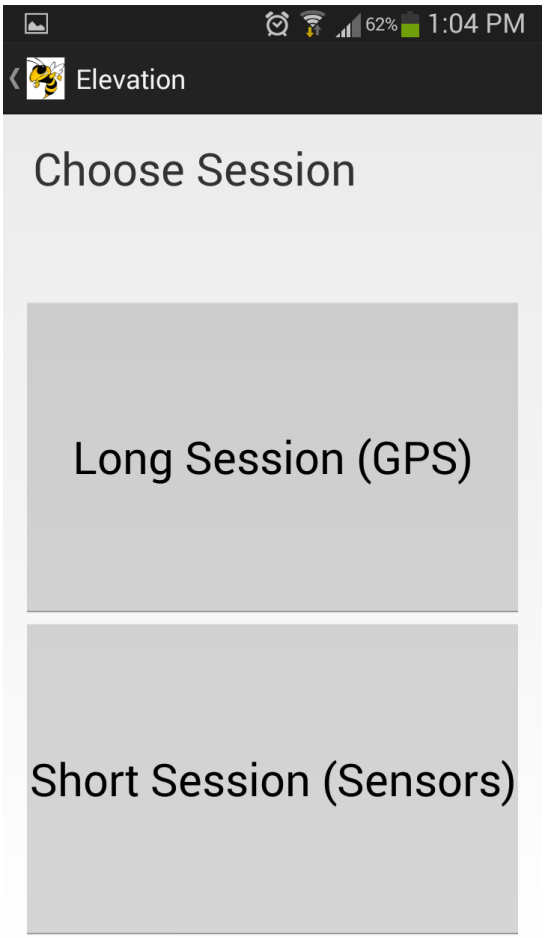


Fig 1: Start Screen

Fig 2: Choose Session

The Long Session is for longer duration sessions where more distance is covered, most likely on a car and hence, GPS coordinates are gathered using Location Detection features of the Android phone described in the image below.

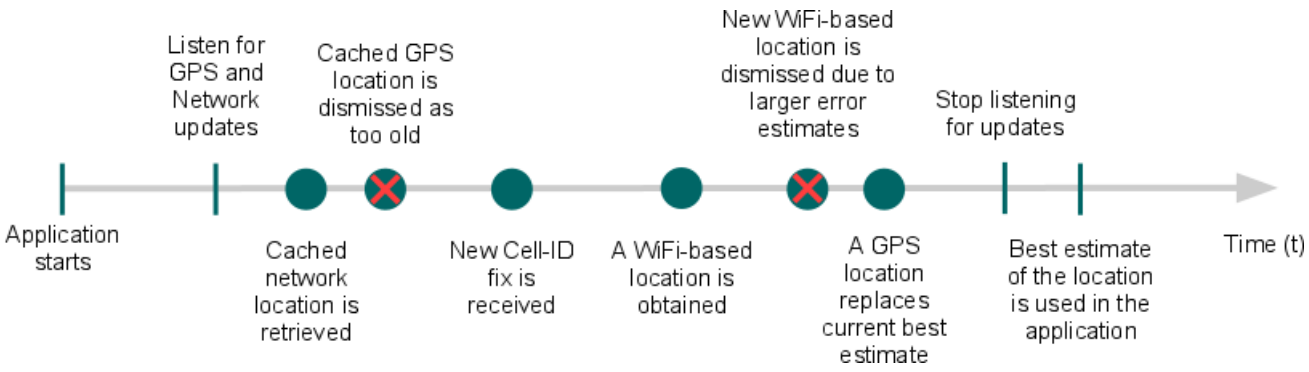


Fig 3: Location Detection on Android Device [12]

The Google Elevation API is used to compute the elevation of multiple points along the path (sampling rate of 30 seconds) and using these data points, the ascent and descent of the user during the session are calculated.

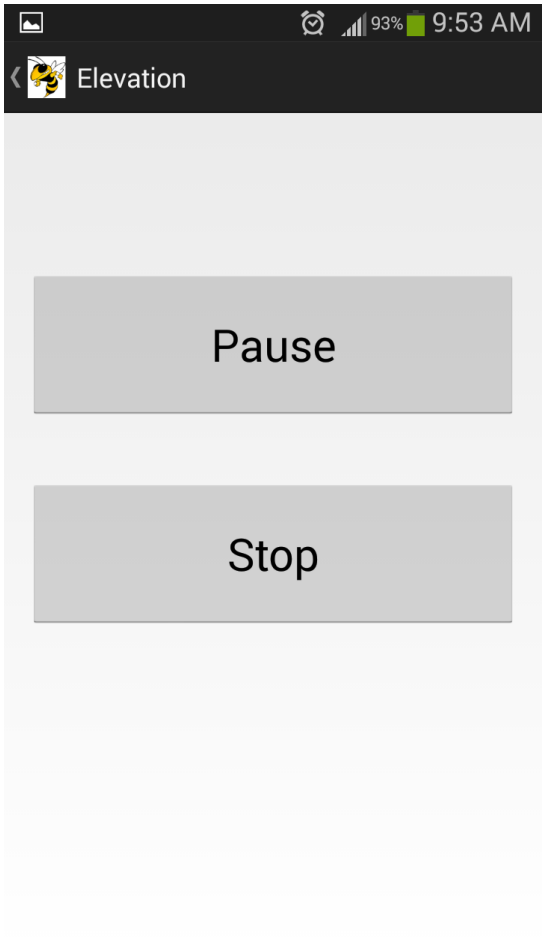


Fig 4: Data being Gathered

Fig 5: Results after Pause/Stop

When the activity is paused, you can resume it by hitting "Back" button that all Android Phones have. When the activity is Stopped, the data is saved on the phone memory and can be viewed by going into the Logs section from the Start Screen.

The Logs section shows the previous session information and the app can be exited from Start Screen by pressing the Exit button that shows up with Settings button of the Phone.

Fig 6: Logs Section

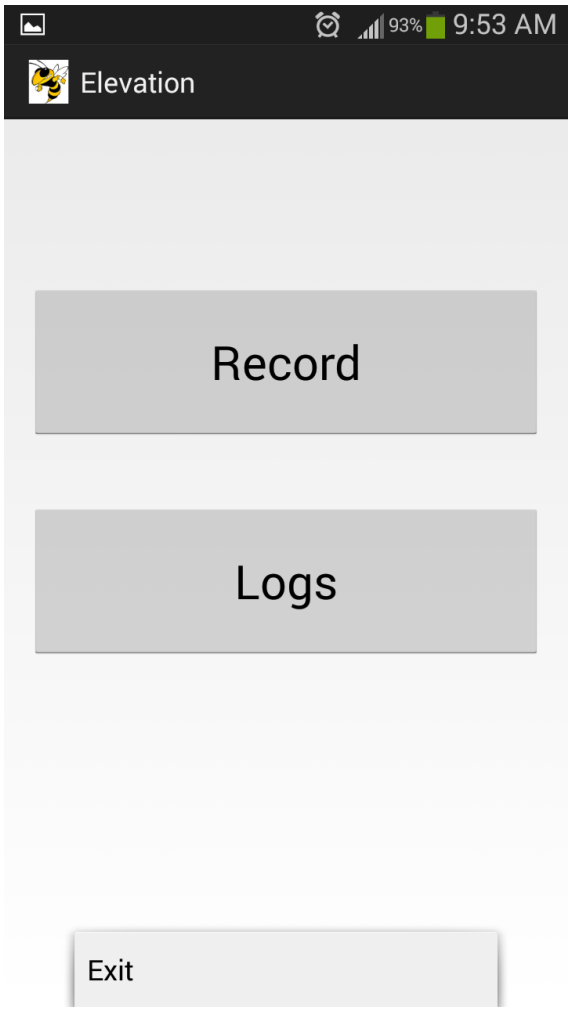


Fig 7: Exit Button (Settings Button)

Evaluation

The Android App was extensively evaluated on Samsung Galaxy S3, primarily by riding up and down in Elevators. The App was also tested by

- keeping it still on a table
- walking on a flat surface (floor)
- going up and down stairs

When the phone was kept flat on a table, the values of distance, ascent and descent were all zeros as expected in spite of non-zero acceleration readings from the sensor.

While walking on a flat surface, both ascent and descent values were registered, but totaling to near zero. That is the net elevation achieved by the walking activity is near zero, which is accurate.

Walking up and down stairs resulted in very erratic readings and had to be discarded.

The readings obtained from going up and down on an Elevator between 1st and 6th floors of Library (approximately 3 meters per floor) are plotted below.

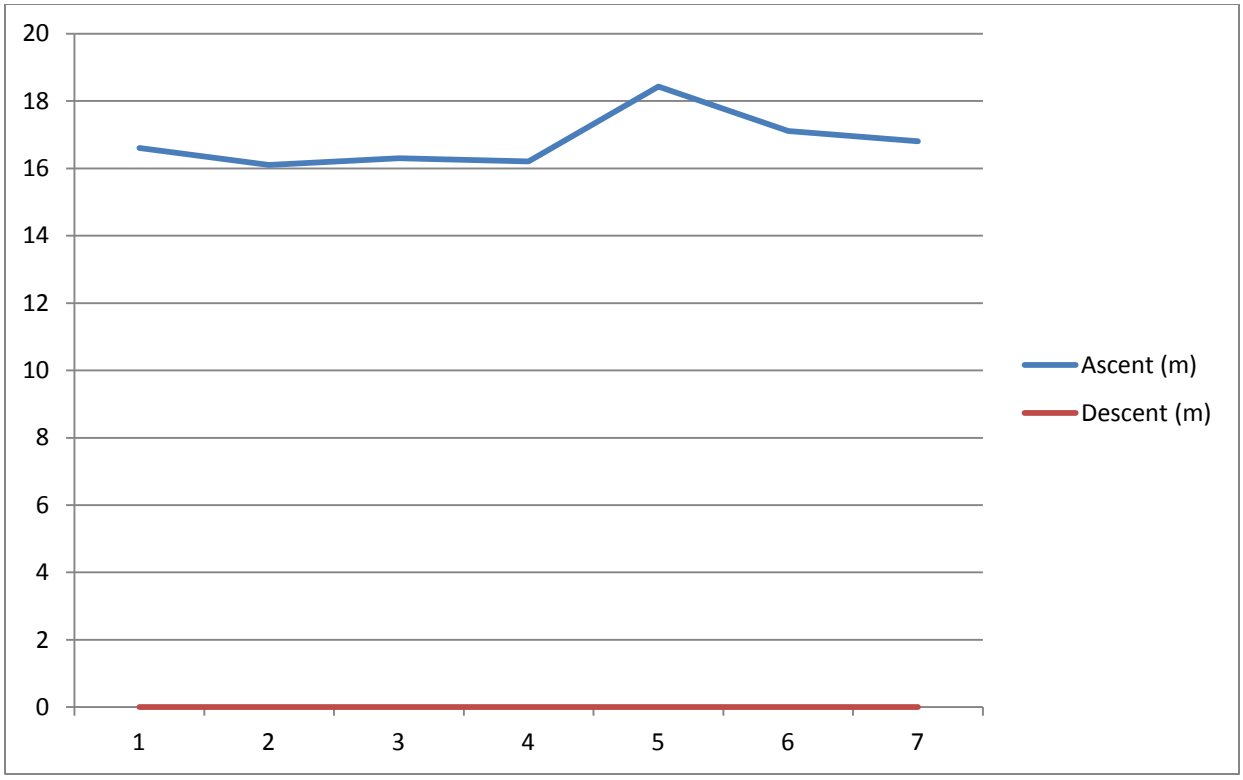


Fig 8: Going from 1st floor to 6th floor (18 m)

The error rate for readings while going up is relatively small at 7%. The app is able to determine that the motion is all the way upwards and not registering any descent. Whereas the graph below shows that while going down, the error rate is at nearly 22% and it also registers some ascent. This is most likely due to the sudden jerks that occur during the start and stop phases of the elevator. Upwards motion is

smoother in older elevators (the elevators in Library are old) and hence low error rate while going up.

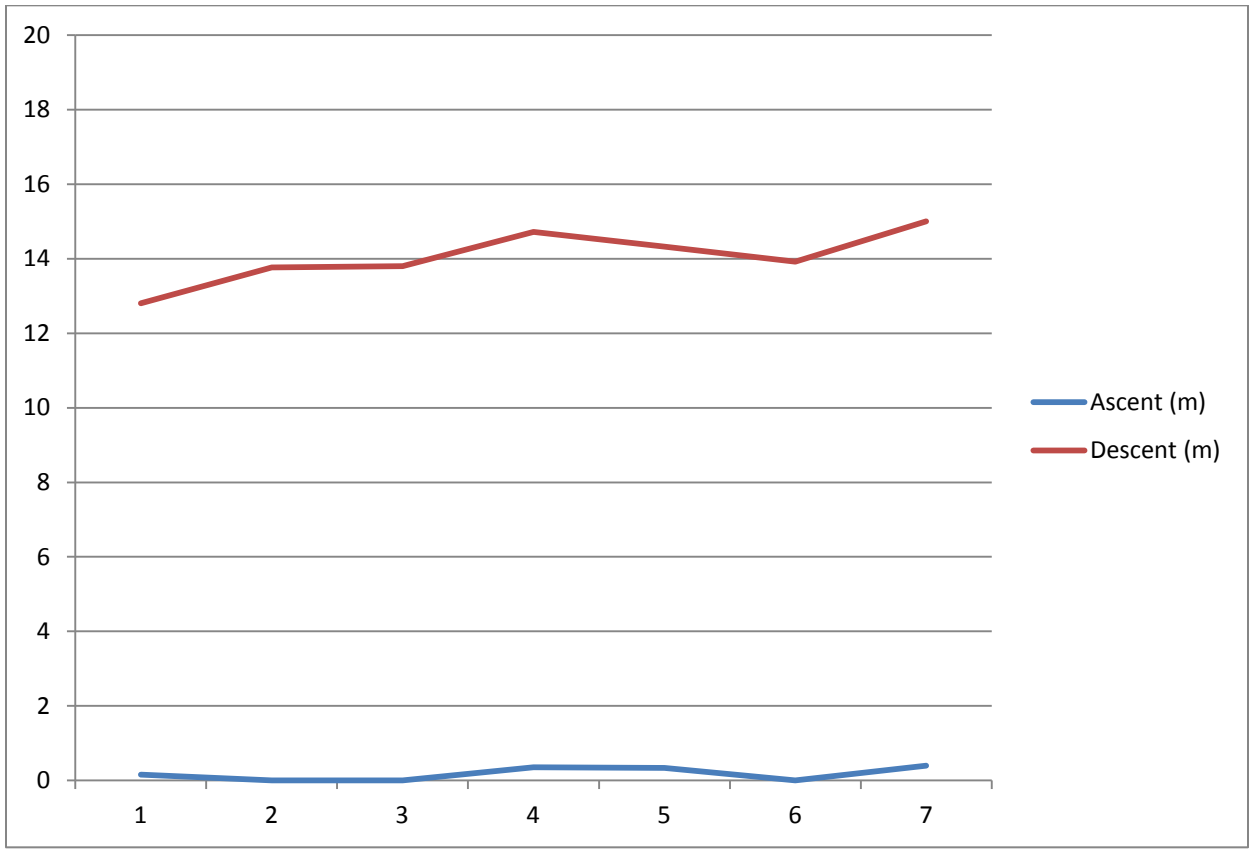


Fig 9: Going from 6th floor to 1st floor (18 m)

Related Work

Google's Project Tango (https://www.google.com/atap/projecttango/) is making a Smartphone specially designed for making the most of the micro-sensors on board. They are also developing APIs for leveraging these micro-sensors including the camera to provide a depth/distance perception aspect to the phone.

The Accelerometer and Gyroscope sensors have been used for a variety of gaming and other applications that involve motion detection. They have also been used in Pedometer apps like Runtastic.

Lessons Learned

Micro-sensors of a smart phone are a treasure trove of information but they are not yet ready to be utilized reliably. Accelerometer can be utilized to a good accuracy for very limited applications. In our case, we were able to make it work for simple vertical motion in a steady, stable environment as that of an elevator.

Higher Sampling Rate doesn't necessarily improve accuracy and orientation of the phone greatly affects the readings.

Different phones have differently calibrated sensors and hence it is next to impossible to make the App portable unless a calibration tool is integrated into the app to customize the filters for each phone and each usage scenario.

Future Work

The Android App could be further developed to provide social media access wherein a user can post the workout statistics directly to social media like Facebook, Twitter and Google+. The App can be integrated with Google Maps and use GPS location information to provide a route map of the hike/jog.

One extension is to gather data from as many phones and as many usage scenarios as possible and use data analytics and machine learning techniques to enhance the accuracy of the filters.

References

[1] Android Location Strategies.
http://developer.android.com/guide/topics/location/strategies.html

Appendix

GitHub Repo: https://github.com/smanchem/Elevation