Detecting User Activities using the Accelerometer on Android Smartphones

Sauvik Das — Georgia Institute of Technology
LaToya Green — University of Houston
Beatrice Perez — University of Puerto Rico, Mayaguez
Michael Murphy — Franklin W. Olin College of Engineering

Supervisor: Dr.Adrian Perrig

July 30, 2010

Contents

1 Abstract

The purpose of this study is to identify whether smartphones pose a security threat to the user. The accelerometer and other sensors within the device can be used without the users consent. Our intent in this is to show that the accelerometer can be used to obtain sensitive information about the user. Using the magnitude of the accelerometer data we found that we could identify general activities preformed by the user, and even have the phone learn new activities. Multiple approaches were implemented to attempt to find the best results. With individual calibration we obtained accuracy of 93%, which could be improved with future work.

2 Introduction

Smartphone security is becoming increasingly important as consumers rely more and more on their smartphones to store personal information. With the development of smartphone operating systems, such as Apple's iOs, Google's Android and Blakcberry's RIM operating systems on the rise, it is imperative that security measures are in place to protect the privacy of the user. This is especially true for Google's Android platform, being that it has an open development environment.

What is Android?

The Android platform is an open platform for mobile devices consisting of an operating system, applications and middleware[1]. Android gives users the opportunity to build and publish their own applications by providing an open development environment. Android treats all applications (native and third-party) as equals[2]. Therefore, having such an open development environment requires security measures to be taken in order to protect the integrity of the Android platform and the privacy of its users.

Android Security and Permissions

The Android Platform takes advantage of several mechanisms designed to protect the privacy and security of Android users, as well as the operating system. These methods include the Android security architecture, application certificates, and application permissions. The purpose of the Android security architecture is to prevent applications from being able to automatically perform operations that could jeopardize the security of other applications, the operating system or the user. Certificates are used to identify the author of a specific application and to prevent users from installing fraudulent software on their devices. Android will not install an application that has not been signed with a certificate. Therefore, the origin of all published applications is traceable.

Android security permissions are handled by the AndroidManifest.xml file present within all application files. When a user downloads an application onto their device, they are automatically notified of the permissions the application has access to. This informs the user of what type of information an application is able to collect from the device as well as the hardware the application can use.

The AndroidManifest.xml file takes care of both software and hardware permissions. But while Android does require permissions for the use of hardware devices such as the camera and vibrator, it does not require permissions to be set in place for the use of any available sensors, including the accelerometer, orientation, and gyroscope sensors[3]. But we have found that these sensors, when used alongside other tools such as the internet and GPS, can also pose as security threat to the user. And it is possible for an application to collect user information from these sensors without the user's knowledge.

Accelerometers

The accelerometer in Android phones measures the acceleration of the device on the x (lateral), y (longitudinal), and z (vertical) axes. Accelerometers can be used to detect movement and the rate of change of the speed of movement. As stated above, the use of accelerometers in Android applications does not require the application to have permission to use it. Therefore, it is possible for an application to collect a user's accelerometer data without the user's knowledge. With accelerometer data and the use of a server to collect the information, it is a fairly simple task for someone to gain a user's personal information, their location, or to figure out what a user is doing or typing.

3 Background Information

Accelerometers have been used for a variety of uses throughout the world today, from medical to research, from car performance to robotics. However, with the advent of the iPhone and Android, accelerometers are much more commonplace in the world of today. The most commonly used accelerometer within our phones is the LIS311DLH, at 3x3x1 millimeters, it is a tiny, low power, high performance linear accelerometer. It senses the forces of acceleration in the X Y and Z planes to a precision of six decimal places. In order to investigate the possible link between cyber security and physical security, we set out to see how these sensors in smart phones, specifically this accelerometer could be used. In our research we wish to see if this device can turn what seems to be a harmless application on the Android Marketplace into a potential spying device, able to detect what actions the user is taking at the moment.

4 Related Work

Many groups have developed sensor recording and processing devices, such as the eWatch, a wearable sensor that is intended for use of monitoring elderly and sick, identify non-responsiveness and keep taps on their position[5]. A Japanese company, KDDI, has developed something similar to this, a system to keep track of employee's actions throughout the day, and send it back to a central mother server. This device can detect such activities such as walking, climbing stairs, or even cleaning. However, little thought from the company has gone into the potential security violations[6]. Many other universities have explored activity recognition using video instead of accelerometer data[7]. While this is not exactly related, it also explores the idea of using a system we don't have on our minds as being a security threat, in fact, the opposite, a security tool - cameras - as something to invade privacy. Indeed, some universities also attempted activity recognition using cell phones as well[8]. Other systems have been suggested by the world today, but not implemented, such as a path tracking device using only accelerometer data, or the use of accelerometer to transmit data other than the data of acceleration. Such phones could then be used as tracking devices for the creators, able to see their user's daily lives through accelerometer data.

Current location tracking systems offer high accuracy using GPS and other such methods; however, those require consent from the user. In this paper we attempt to show such things as location and activity can be detected and recorded without the knowledge of the user, using simply the accelerometer within their phone.

5 Methodology

5.1 Background

The data received from the accelerometer was in the form of a three-valued vector of floating point numbers that represented the individual accelerations of the smartphone device in the X, Y, and Z axes subtracted by the gravity vector $\mathbb{G}$. The acceleration values were recorded in meters per second squared($\frac{m}{s^2}$). Thus, layed flat on a level surface, the expected reading of the accelerometer would be approximately **[0,0,-9.81]**. The process of converting these acceleration vectors into known activities occured in four broad phases: *data acquisition,signal processing*, *feature extraction* and *classification*.

During the *preprocessing* stage, each sampled acceleration vector was combined into a single magnitude; in the *noise reduction* stage, the received signal was linearized and smoothed to reduce noise; in the *feature extraction* stage, features were extracted for each sample window of a predetermined number of *512* samples; in the *classification* stage, unknown patterns of data were classified via feature comparison of known patterns of data, using the Nearest Neighbor and Naive Bayes classifiers. Two versions of the classification stage were implemented, one *online*–which was done on the fly as data was being received–and one *offline*–which was done on a server after all relevant data had been collected.

Prior to implementation, we had decided on attempting to classify the following five activities:

- **Phone Detached**: The user is not currently holding the phone
- **Idle**: The user is holding the phone in an idle state
- **Walking**: The user is walking
- **Running**: The user is running
- **Jumping**: The user is jumping

. Having achieved significant accuracy in classifying the aforementioned activities, two additional activities were added to the list:

- **Descending Stairs**: The user is descending stairs
- **Climbing Stairs**: The user is climbing stairs

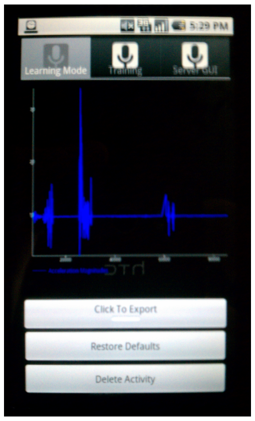

Figure 1: The data acquisition application running on the Motorola Droid

. Following reasonable success with these additions, users would be allowed to enter customized gesture patterns.

5.2 Data Acquisition

The device used was the Motorola Droid. To acquire the accelerometer data, we created an application for the Droid using the Android SDK. Training data was acquired by having users perform an instructed task (e.g., walking, running, jumping) while the data was either being exported to a server or written to the Droid's SD card. The application also graphed the sampled acceleration data, allowing for a real-time preview of the data. Figure 1 shows a screenshot of the application running on the Droid.

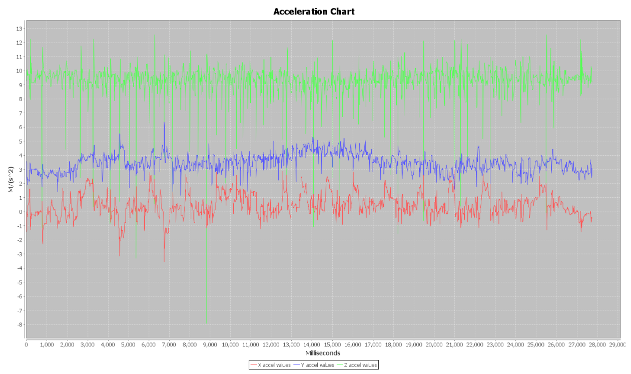

Figure 2: Raw accelerometer data in the X,Y and Z. Horizontal axis units are in milliseconds, and vertical axis units are in $\frac{m}{s^2}$

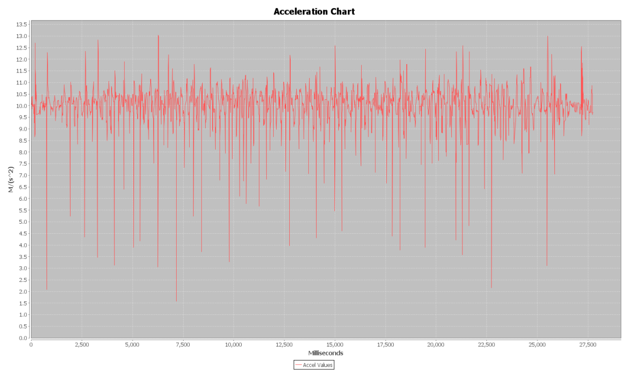

Figure 3: Merged acceleration magnitude data. Horizontal axis units are in milliseconds, and vertical axis units are in $\frac{m}{s^2}$

5.3 Signal Processing

5.3.1 Preprocessing

The first step taken was to merge the three-dimensional input signal into one acceleration magnitude. The original signal can be seen in Figure 2. We found the magnitude of the acceleration vector by taking the euclidean magnitude of the three individual values, that is:

$$\bar{a} = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

The signal run through equation (1) can be seen in Figure 3. The merger was done to simplify feature extraction, because the overall theme in the acceleration pattern was deemed to be sufficient for the recognition of our original set of activities, none of which required distinction of directional accelerations. However, features from individual acceleration axes may be important in determining activities where directional information is relevant, such as differentiating between martial arts motions.

5.3.2 Noise Reduction

There were two primary sources of noise in the received signal. The first was irregular sampling rates and the second was the noise inherent in discrete physical sampling of a continous function.

The accelerometer was likely sampled irregularly because of the Android framework's implementation of the sampling mechanism. The Android API offers four abstract sampling rates for its accelerometer sensor (listed from fastest to slowest): *Fastest*, *Game*, *Normal*, and *UI*. The actual physical sampling capabilities of the acclerometers likely vary from device to device, so these sampling rate options are used more as guidelines than as actual physical sampling rates. Another reason is because of how application on the Android framework receives acceleration readings. The Android API only allows the acquisition of an acceleration sample on an ”onSensorChanged()” event, which fires whenever the Android OS determines that the accelerometer values have changed. As such, an acceleration sampling application cannot force a reading of the accelerometer at predetermined intervals. Combined with the fact that Android OS likely has varying loads of activity from moment to moment, the ”onSensorChanged()” is not scheduled firmly, resulting in irregularly sampled values. To regularize sampled values, the holes in the signal were interpolated via a process called data linearization.

Outside of irregular sampling rates, additional noise was periodically introduced to the signal just from the nature of the activity patterns performed. For example, a slight change in the orientation of the phone is quite common even in an Idle position, but can result in an anomalous peak in the resultant signal. This type of noise was handled by running the signal through a 5-point smoothing algorithm.

5.3.3 Linearization

Data Linearization was the process used to handle the irregular sampling of the received acceleration signal. The process involved choosing a desired regular sampling rate, and interpolating all the holes in the data via linear interpolation. The linearized value would be calculated by finding the closest sampled data point before the desired sampling time, and the closest sampled data point after the desired sampling time, and then interpolating what the value of the desired sample time would have been. One significant problem in the linearization process was to determine an ideal desired sampling rate.

In order to ensure that not too much data was calculated via interpolation - which could have resulted in a false recreation of the original signal - a program was created which determined the mean and minimum differences in time (in milliseconds) of subsequent readings. The arithmetic average of these two values was then calculated and analyzed for multiple datasets in order to provide us with a general guideline of what might be an ideal sampling rate. It was determined that approximately 1 sample every 8 milliseconds, or 125 samples per second was a good sampling rate for the data.

A signal before and after data linearization can be seen in Figures 4 and 5, respectively. Notice that while the X axis has almost doubled in length, the key features of the signal are exactly preserved. This is expected because data linearization is simply meant to fill in the holes of a sampled signal, and should not significantly alter the signal itself. The X domain is doubled because we used the arithmetic average of the minimum and mean sampling time differences, which in this case corresponded to about half the mean difference.

5.3.4 Smoothing

The linearized signal was then run through a 5-point smoothing algorithm to reduce

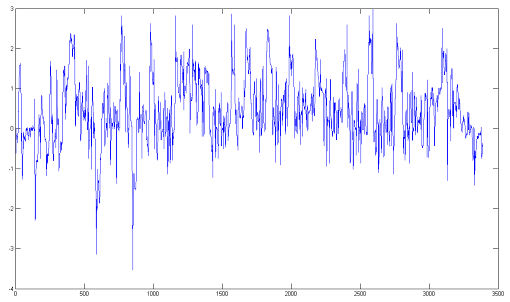
Figure 4: An acceleration signal before data linearization. X-axis is the sample number, Y-axis is in $\frac{m}{s^2}$

any additional noise. This additional noise could have come from any of a number of sources (e.g., idle orientation shifts, screen taps, bumps on the road while walking). The 5-point smoothing algorithm calculates each point to be the average of its four nearest neighbors, the two nearest before and the two nearest after. The 5-point smooth was chosen so that spikes with an observable, steady progression would be preserved while anomalous, sudden spikes would be eliminated.

We tried running the signal through a 5-point smooth between one and five iterations, with three iterations yielding the most accuracy. This is because over-smoothing caused subtle distinctions between similar, but different signals - such as the Idle versus Phone Detached signal - to disappear.

Figure 6 shows the signal shown in Figure 5 after smoothing. Notice how in Figure 5, the signal is very jaggedy and discrete - a key indication of a noisy signal. In Figure 6, the signal is more smooth and continous.

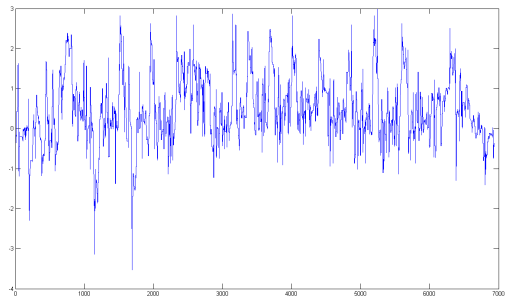
Figure 5: An acceleration signal after data linearization. X-axis is the sample number, Y-axis is in $\frac{m}{s^2}$. Notice that the X-domain is significantly longer in this image, though the shape of the curve is approximately the same.

Revisiting data linearization, Figure 7 shows the same signal smoothed, but not first linearized. Notice how the signals in Figures 5 and 6 are significantly different. Thus, it was determined that after smoothing, the differences between linearized and non-linearized signals are noticable and significant.

5.4 Feature Extraction

Feature Extraction was the process used to extract the key elements from a processed signal which made the signal distinct. For example, how could one represent the signals in Figure 8 such that a classification algorithm could later distinguish between the three signals? As humans, it is simple for us to make the distinction visuall; but machine algorithms require concrete statistics - features.

The following features were chosen based

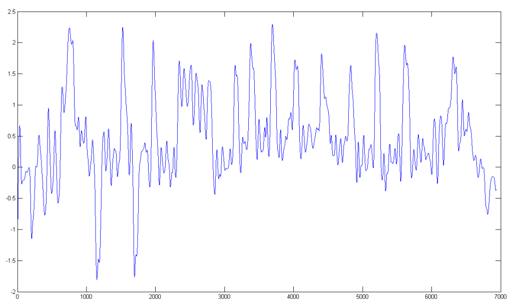

Figure 6: Smoothed acceleration signal. Compare this signal to the signal in Figure 5.

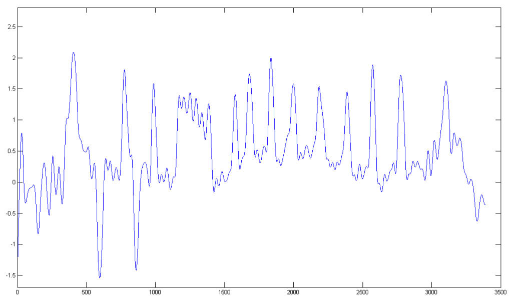

Figure 7: Smoothed, but not linearized acceleration signal. Notice how this signal is significantly different from the one in Figure 6.

on our own discussion and the recommendations of previous accelerometer based activity recognition research:

1. **Fundamental Frequencies**: The fundamental frequencies of the signal. These were found from the Fourier Transformation of the signal over the sample window. The final value was the average of the three dominant frequencies of the signal.

2. **Average Acceleration**: The arithmetic average of the acceleration values in the sample window.

3. **Maximum Amplitude**: The maximum value of the signal in the window.

4. **Minimum Amplitude**: The minimum value of the signal in the window.

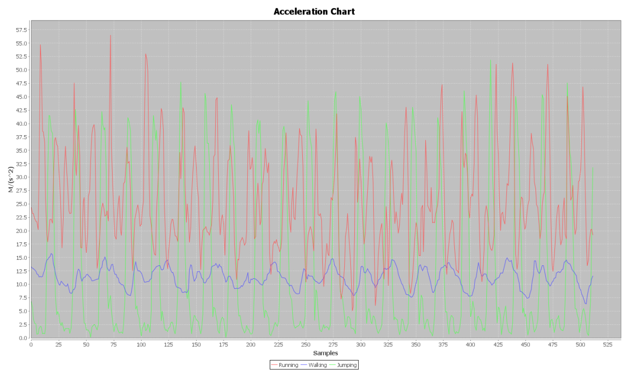

Figure 8: A thirty second-spread of Walking, Running and Jumping accleration signals.

Note that a sample window of 512 samples was used. 512 samples were chosen based on the recommendation of prior papers, because it was deemed sufficient for activity recognition and because 512 is a 2^n number, which is ideal for the Fast Fourier Transformation algorithm.

5.5 Classification

During the classification stage, we labeled unknown patterns of data based on the knowledge acquired from known patterns of data. For the online implementation of the Activity Recognition application, we implemented a 1-Nearest Neighbor classifier on the Motorola Droid. The 1-Nearest

Neighbor algorithm is a simple algorithm which matched unknown patterns of data with a known pattern of data based on the euclidean distance between the two feature vectors. The known pattern of data with the lowest euclidean distance from the unknown pattern of data was determined to be the best match. For the offline classification, we used the Naive Bayes classifier provided by the Weka 3 data mining software. The Naive Bayes classifier works on the premise that the absence of a single feature does not necessarily disqualify a known pattern of data from being the correct match. Despite this assumption, Naive Bayes has been shown to be a very effective classifier. Unfortunately, Weka 3 could not be run on the Droid because of Java Mobile Edition's limited functionality.

6 Results

We found that using the nearest neighbor classifier the program could predict patterns or activities with 93% accuracy after it had been calibrated for a particular user. We found that for people of the same height the phone could predict activities even if the subject had not gone through the training process. A different classifier, for example Naives Bayes could complete the predictions for a bigger population after a minimum amount of samples had been collected. Individual gestures were recognized with as much accuracy as activities but once again machine learning had to come first.

One of the limitations of our program is that for the phone to make accurate predictions, the handset would have to be in the same place as it was when it was in the training mode for example if during the training mode the phone was being held in the hand, then for more accurate predictions, the phone would have to be in the hand of the person that was using it. Having it be on the pocket or the purse could make the classifier label an activity incorrectly.

7 Conclusions and Future Work

The research shows that sensors in the android platform could constitute a violation of privacy for the users. Still to be done on the activity recognition implementation would be to use a better classifier that would predict for a greater amount of people without having to go through the training process. The accelerometer proved to be a useful tool in identifying activities based on your phone's movements and other uses of the accelerometer could potentially include its use in detecting key presses and even text without the knowledge of the user. Another interesting application would be the transfer of information between phones based on the matching acceleration values between them. Overall we found that the safeguards in place are not enough and have flaws that must be corrected in future updates to the platform.

8 References

[1] ”Android FAQ - What Is Google Android?” Android News - Android Google Phone Forums. Web. 20 July 2010. ⟨*http : //www.talkandroid.com/google − android − faq/*⟩.

[2] Meier, Reto. Professional Android 2 Application Development. Indianapolis, IN: Wiley, 2010. Print.

[3] ”Security and Permissions.” Android Developers. Web. 20 July 2010. ⟨*http : //developer.android.com/guide/topics/security/security.html*⟩.

[4] ”LIS331DLH MEMS digital output motion sensor ultra low-power high performance 3-axes ”nano” accelerometer.” Web, 28 July 2010. ⟨*http : //www.st.com/stonline/products/literature/ds/*15094*.pdf*⟩

[5] Uwe Maurer, Anthony Rowe, Asim Smailagic, and Daniel Siewiorek. Location and Activity Recognition Using eWatch: A Wearable Sensor Platform. 2008. Print.

[6] ”Mobile that allows bosses to snoop on staff developed ” BBC NEWS - Technology. Web. 27 July 2010. ⟨*http : //news.bbc.co.uk/*2*/hi/technology/*8559683*.stm/*⟩.

[7] Weiyao Lin, Ming-Ting Sun, Radha Poovandran, and Zhengyou Zhang. ”Human Activity Recognition for Video Surveillance.” Web 29 July 2010 ⟨*http : //www.ee.washington.edu/research/nsl/papers/iscas−*08*.pdf*⟩.

[8] Nishkam Ravi and Nikhil Dandekar and Preetham Mysore and Michael L. Littman. Activity Recognition from Accelerometer Data. Department of Computer Science Rutgers University

[9] Bao, L., and Intille, S. S. 2004. Activity recognition from userannotated acceleration data. In Proceceedings of the 2nd International Conference on Pervasive Computing.