A Proposal to use Dynamic Thresholding Techniques and Machine Learning Algorithms to Improve Existing Fall-Detection Systems.

Kevin Ringstaff
University of Tennessee at Chattanooga
ringstaff.k@gmail.com

Abstract

A dynamic threshold based fall detection algorithm using individual characteristics and feedback is proposed. Falling is one of the most significant causes of death and injury for elderly citizens, and rapid responses to a fall can dramatically increase the likelihood of a recovery. Data is gathered using a tri-axial accelerometer on a mobile phone and then analyzed to detect a fall. In the event of a false positive, the user is prompted to select the ADL that they were participating in at the time. This data is uploaded to a central server where machine learning algorithms are used to analyze and modify the thresholds and send the new thresholds back to the individual users.

1. Introduction

Falls affect over one in every three elderly people [14], [15]. Falls currently reason for injury death and extended hospital stays for elderly people. This is especially relevant because of the impending influx of the baby boomer generation [5]. As more and more people retire, accidental falls will continue to increase, and will have the ability to overwhelm our medical system. This is why an automated fall detection system that is accurate, cheap, and easy to use is in such high demand.

Detection of a fall is performed by analyzing and classifying Activities of Daily Living (ADL). ADLs are normal activities that are frequently performed throughout the day, such as standing, sitting, lying down, walking, and running. One problem that many early fall detection systems suffered from was a lack of a set of guidelines amongst different research teams. The authors of Abate et al. (2007) defined a fall as "Unintentionally coming to the ground or some lower level and other than as a consequence of sustaining a violent blow, loss of consciousness, sudden onset of paralysis as in stroke or an epileptic seizure." [3].

They go on to divide falls into two types: Accidental falls, those caused by environmental factors, and non-accidental falls which are caused by biological factors.

Although there are a number of appropriate devices that are able to be used in fall detection, mobile phones were chosen in this study for a variety of reasons. They are relatively inexpensive, programmable, and widely available which greatly aids in testing purposes. Another feature is their ability to call for help in the event that anything should happen, even if the user is unconscious and unable to do so themselves. Additionally, mobile phones are extremely portable in contrast to other detection devices; most people are confortable "wearing" their phone and they usually have it on them at all times.

2. Materials and Methods

2.1. Hardware and Software

The application is designed for and tested on the Droid Charge by Samsung. It has a 1GHz processer, 2GB of internal Memory, 512MB of RAM, and a 4.3-inch display. It runs on Verizon's 4G LTE network [12]. The Droid has a single tri-axial accelerometer sensor which is generally measured from waist level as the phone is often stored in the pocket.

We chose to develop for the Android platform because it is open source, written in Java, and because it provides a robust coding environment that is well documented and familiar to use. We decided to develop the application for Android SDK 10, primarily because it is the most widely installed SDK version as of April 26, 2012 [11]. Furthermore, because all future SDK releases are guaranteed to support legacy applications, our program will continue to be supported by the vast majority of Android devices because older phones will be gradually phased by newer devices.

2.1. Application Design

In order to have the most impact, the application must be easy and intuitive to use. Because our primary audience is the elderly population, we designed the UI such that it specifically caters to people with poor eye-sight and unsteady hands by making the buttons as large and as easy to read as possible. Additionally, in the event that a fall actually happens, there is no user interaction required for the phone to call for help.

One major concern in implementing a fall detection algorithm on a mobile phone is battery life. Since the algorithm has to be constantly running to be effective, we designed the program to start and stop a background service [1], [2]. A service is an application that constantly runs in the background. It is able to use less system resources primarily because it does not require the use of the screen or user input. Once the service is started, it runs in the background until it is explicitly stopped by the user.

We initially designed our application and algorithms based on the source code that was provided in the master's thesis by Gonzales [1]. Gonzales provided three algorithms with varying thresholds, and we chose Algorithm 1 as a starting point because he concluded that it had performed the best [1].

The overall program is simple: The user opens the application and presses the 'Start Monitoring' button to start the fall detection service. The service then begins and the user is free to exit the application without stopping the service. The service registers an event listener with the SensorManager which calls the onSensorChanged() method whenever an acceleration change is detected.

The Android API provides four options for the sampling rate of the sensor: Normal, UI, GAME, and FASTEST which provide an effective sampling rate of approximately 200ms, 60ms, 20ms, and 10ms respectfully. Since the algorithm was initially designed with storage considerations in mind, it only keeps track of the current and immediately previous accelerometer values. Additionally, a lower sampling rate is beneficial to preserving the battery life of the device. Thus, we decided that a sampling rate of 200ms would be sufficient for this purpose.

However, a lower sampling rate does not necessarily mean poor performance. Essentially, all body movements have a frequency measured below 20Hz, and there have been other studies which suggest even lower sampling rates are acceptable [13]. Future research is required to determine how faster or slower sampling rates affect the overall accuracy of the algorithm.

3. Fall Detection

The most common method of detecting a fall is by using thresholds. Thresholds are basically an upper or lower bound that the acceleration values are compared against [3]. There is an upper fall threshold (UFT) and a lower fall threshold (LFT). These values are typically set by analyzing multiple simulated falls for the average upper and lower bounds and calculating the root-sum-of-squares (RSS) signal from the accelerometer data, equation 3.1 [4].

$$RSS = \sqrt{x^2 + y^2 + z^2}$$

Equation 3.1

The UFT is the upper magnitude that measures the force of the impact, generally in G values, equation 3.2. The LFT is the lower magnitude that measures when a body is in free-fall. A person at rest has a RSS value of 1G. As a person falls, the RSS falls from 1G towards 0G. When they hit the ground, the RSS rapidly increases.

$$1G = 9.81\frac{m}{s^2}$$

Equation 3.2

A fall must start with a period of free fall, which causes the acceleration amplitude to drop significantly below the LFT threshold which is typically set somewhere below 1G [4]. There is a sharp spike in the graph when the fall stops. The acceleration amplitude then rapidly increases above the UFT threshold, which is typically set at or above 3G [2].

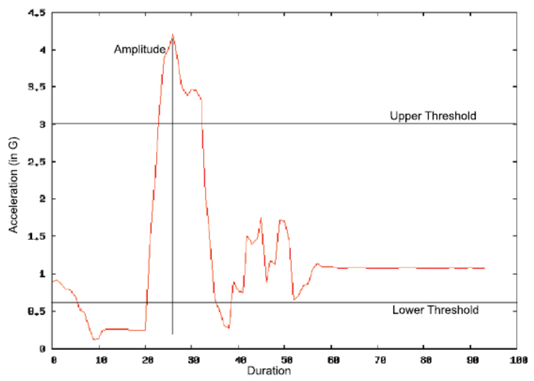

Figure 3.1 – UFT and LFT. From [2].

After both the LFT and UFT thresholds have been exceeded, in that order, the detection algorithm checks for a period of inactivity. This is referred to as the "long lie" [7]. The rationale behind this is that a person is likely to be immobile for a short time while they recover. Many systems incorporate this method because it can greatly reduce the occurrence of false positives [1], [2], [7], [9]. The long lie is only a matter of seconds, usually 3-5, although some systems use 20 seconds or more [13]. A fall is detected when the LFT and UFT have been exceeded and the system detects a long lie.

3.1. False Positives

One problem in using static thresholds is that the system detects many false positives. A false positive is an activity that is incorrectly classified as a fall. There are many ADLs that are routinely misclassified; for example, jumping, sitting down quickly, and laying down quickly on a bed. These ADLs are mistaken for real falls because they are characterized by kinematic peaks similar to those of real falls [7].

There have been numerous studies and techniques that were developed to increase the specificity of the system [7]. One such method that, arguably, has the most potential is using dynamic thresholds. Dynamic thresholds allow the system to modify UFT and LFT by specifically tailoring the thresholds to an individual user based on a number of characteristics. For example, there are studies that query the user for their weight, height, and average activity levels and then use that data to modify the thresholds from their original value. The thresholds for an active user would therefore be less sensitive than those from a non-active user based on the mentality that an active user would produce higher RSS values [2].

However, this approach, while an improvement, still suffers from some of the same issues. A more ideal solution would be to implement a real-time algorithm to dynamically update the thresholds based on user activity. There are quite a few studies that have either suggested this or attempted it [1], [2], [5], [8], [13]. There are a number of algorithms specifically suited to an adaptive learning environment which have been used to construct a model of typical fall patterns. Support Vector Machines (SVM), Random Forest, Bagging, and Naïve Bayes have been successfully implemented and have been shown to increase the specificity of the algorithm on a closed dataset to about 95% [8].

There are a number of problems in implementing those machine learning methods. One problem is the significant amount of calculations and storage space that is required. Another issue is with the actual data that was used to build the classifiers. There are a number of problems that have been reported with volunteers simulating falls in a lab setting. Most of the studies used young subjects who were both willing and able to simulate falling in a controlled environment [4]. The subjects were instructed not to try and break their fall, which is contrary to elderly people who would naturally try to avoid the fall which would result in a lower overall UFT value. Additionally, because the subjects used crash pads, the recorded UFT values would be lower than if the fall was on a harder surface.

4. Proposal

We propose a similar system to [1] and [2] which will start with an initial fixed threshold, calculated based on provided user information such as height, weight, and average activity level. Afterwards, the system will start the fall detection service to run in the background with the newly calculated thresholds. The system will undoubtedly still produce false positives, and when it does the user will be presented with a screen asking if they have fallen. If they respond that it was not a fall, then the system will locally save all of the data that was acquired leading up to the false positive classification. Later, when the phone is connected via USB to a computer or a power supply, that data will be uploaded to our server. From there, we can run the more complex classification algorithms to further refine the system.

Additionally, when the user indicates that a false positive has occurred, they will be presented with a drop-down menu that has a list of typical ADLs, such as sitting down on a hard chair, jumping, and running as well as a space to type in the activity if it is not in the list. Using this additional information will help us further analyze the data in hope of not only preventing false positives, but it will aid in classifying ADLs in general.

Since this will be developed in the Android environment and freely downloadable, we should theoretically be able to gather data from a large amount of people. Eventually, the inevitable will happen and we will get some data from actual falls. While tragic, this real data will be extremely helpful in aiding the development of a fully functional classification system.

5. Conclusion

We hope that by utilizing these machine learning methods and user feedback to dynamically adjust the thresholds for specific users that it will reduce the false positive rate dramatically. Additionally, we hope to incorporate the user feedback into a larger database where we can further analyze data collected from real users of the system.

6. References

[1] I. J. D. Gonzales. Fall detection using a smartphone. Master's thesis, Gjovik University College, 2010-2011. Available in http://www.stud.hig.no/~090285/falldetection.pdf.

[2] F. Sposaro and G. Tyson. iFall: An Android Application for Fall Monitoring and Response. In Conf Proc IEEE Eng Med Biol Soc, 2009, pp. 1:6119-22.

[3] S. Abbate, M. Avvenuti, P. Corsini, J. Light, and A. Vecchi. Monitoring of Human Movements for Fall Detection and Activities Recognition in Elderly Care Using Wireless Sensor Network : a Survey, Wireless Sensor Networks: Application-Centric Design, Merrett, G.V., Tan & Y.K., 2010, pp. 1-20.

[4] A. K. Bourke, J. V. O'Brien, and G. M. Lyons. Evaluation of a threshold based tri-axial accelerometer fall detection algorithm. *Gait & posture*, 2007, pp. 26(2):194–9.

[5] B. Brown. An Acceleration Based Fall Detector: Development, Experimentation, and Analysis. Summer Undergraduate Program in Engineering Research at Berkeley (SUPERB), University of California, Berkeley, 2005. http://www.eecs.berkeley.edu/~eklund/projects/Reports/GarrettFinalPaper.pdf

[6] Q. Li, J.A. Stankovic, M.A. Hanson, A.T. Barth, J. Lach, and G. Zhou. Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information. BSN'09, 2009, pp. 138–143.

[7] S. Abbate, M. Avvenuti, G. Cola, P. Corsini, J. Light, and A. Vecchio. Recognition of False Alarms in Fall Detection Systems. CCNC, IEEE, Italy, 2011, pp. 23-28.

[8] M. Luštrek, B. Kaluža, Fall Detection and Activity Recognition with Machine Learning, Informatica, 2009, pp. 33 2.

[9] A. K. Bourke and G. M. Lyons. A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. *Medical engineering & physics*, 2008, pp. 30(1):84–90.

[10] Google Inc. Android. http://developer.android.com.

[11] AppBrain. http://www.appbrain.com/stats/top-android-sdk-versions.

[12] Android Central: Droid Charge Specifications. Mobile Nations. http://www.androidcentral.com/droid-charge-specs.

[13] D.M. Karantonis, M.R. Narayanan, M. Mathie, N.H. Lovell, B.G. Celler. Implementation of a Real-Time Human Movement Classifier Using a Triaxial Accelerometer for Ambulatory Monitoring. IEEE Trans Inf Technol Biomed, 2006, pp. 10:156–167.

[14] Lord SR, Ward JA, Williams P, Anstey KJ. An epidemiological study of falls in older community-dwelling women: the Randwick falls and fractures study. Aust J Public Health, 1993, pp. 17(3):240–5.

[15] Tinetti ME, Speechley M, Ginter SF. Risk factors for falls among elderly persons living in the community. N Engl J Med 1988, pp. 319(26):1701–7.