Smartphone-based Fall Detection, Final Paper

Ariel Keller

1. Introduction

Falls are the leading cause of admission and extended stay in a hospital among the elderly [3]. They are also the sixth cause of death for those over 65 [3]. As the current population ages, we expect to find ourselves with a higher percentage of elderly population than ever before [6]. In light of this, falls and fall detection are of immediate importance. We propose to develop a fall detection system to meet this need.

1.1 Definition

A fall begins with a period of free fall. Then, there is impact with the ground, and after the impact, a period of inactivity, which can range from a matter of seconds to a few hours [3]. During free fall, a person's RSS value falls below 1g, approaching 0g, and then upon impact there is a sharp spike in the person's acceleration magnitude.

The RSS is the root-sum-of-squares or acceleration magnitude and is calculated using the equation below.

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

Equation 1.4

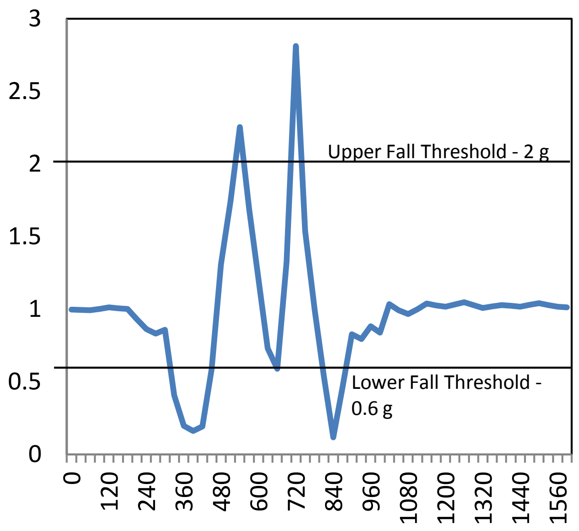

Figure 1 - A backwards fall with UFT and LFT indicated.

1.2 Evaluation parameters

There are two measures that are commonly used to evaluate and compare fall detection systems. These are sensitivity and specificity.

- **Sensitivity** is the system's ability to detect real falls.
- **Specificity** is the system's ability to detect only real falls, in other words, its ability to filter out false positives.

In order to calculate these, one must first tally up the true positives, false negatives, false positives, and true negatives resulting from the tests.

True positive (TP): a fall occurs and is correctly detected.

False negative (FN): a fall occurs and is not detected.

False positive (FP): a fall did not occur, but the system detected a fall.

True negative (TN): a fall did not occur, and the system did not detect a fall.

$$Sensitivity = \frac{TP}{TP + FN}$$

Equation 1.1

$$Specificity = \frac{TN}{TN + FP}$$

Equation 1.2

From a medical standpoint, it is most important to have a high sensitivity, and the specificity is secondary to sensitivity. Clearly, it is essential to detect any real falls that occur; it is dangerous for a real fall to remain undetected. While specificity is secondary, it is still important. It is inconvenient and impractical for a fall detection system to have frequent false positives; this would make for an unreliable system, and any alarms would likely not be taken as seriously, making the system less effective.

1.3 Motivations

As previously noted, there is high demand for a fall detection system. And, in fact, there has been a lot of research done recently on developing algorithms for fall detection, and many different systems have been developed to detect falls. However, many of them suffer from frequent false positives. Bourke et al. [5] studied several different algorithms, and even the best of the algorithms they studied, which was actually quite good, still had a rate of approximately 0.5 false positives per day.

Because of the demand for a fall detection device and the shortcomings of the systems currently available, it is necessary to research ways to decrease the amount of false positives detected, and hence increase the system's specificity, while still retaining high sensitivity.

Also, many systems use sensors whose sole purpose is to be used for fall detection, and they often use more than one accelerometer and require that the sensors be attached in specific locations in order to make it possible to determine the user's posture. This method allows for better specificity; however, it also makes the system inconvenient to use. We chose instead to implement our system on an android phone that is carried in the user's pocket. This is convenient for several reasons. Android phones are ubiquitous and relatively inexpensive. Also, since our system only requires data from the single accelerometer on a phone, the user would not need to wear multiple dedicated sensors. Finally, in our system, the phone need not be strapped on in locations to which it may be difficult to attach a device, such as the calf, thigh, or head as used in (**list source numbers). Rather, the phone can simply be placed in any pants pocket located near the waist. Additionally, there are no restrictions as to how the phone must be oriented in the pocket.

While these lax parameters about the placement and orientation of the phone make the system very convenient to use, they do introduce some additional difficulty in detecting a fall. Because we don't require any a priori information about the phone placement or orientation, we cannot assume anything about the sensor's orientation. This renders it useless for us to use the acceleration values for each individual axis. Instead, we use the acceleration magnitude. We calculate the acceleration magnitude using the root-sum-of-squares or RSS.

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

Equation 1.4

2. Related Work

A commonly used approach to detect falls is to use two pre-determined threshold values to test the RSS values. The first threshold is the lower fall threshold (LFT), which is set to some value between 1g and 0g, and the second threshold is the upper fall threshold (UFT), which is set to an empirically determined value, typically 3g or higher [2]. In order for the system to detect a fall, the RSS value must first fall below the LFT, indicating free fall. It must then rise above the UFT, indicating the impact. Some systems also check for a period of inactivity after testing for both thresholds, which helps to decrease the amount of false positives [1].

Another approach that is often combined with the threshold approach is that of using gyroscopes to measure angular velocity and determine posture information. That posture information is then used along with the accelerometer data to determine if there has been a fall [3]. However, gyroscopes use more energy than accelerometers, and therefore have a greater impact on battery life [3]. Clearly, battery life is a significant consideration when using a smartphone for fall detection. Thus, if it is possible to develop an accurate fall detection algorithm without the use of gyroscope information, we would prefer to do so.

Abbate et al. [2] proposed an interesting and unique method for decreasing false positives. They noted that there are some normal activities of daily living (ADLs) that exhibit kinetic peaks similar to those that are present in falls [2]. Therefore, when using the threshold technique, which is a very common method of detecting a fall, these ADLs are likely to register false positives. Abbate et al. used some properties of the different classes of ADLs to distinguish them from falls. In so doing, they were able to define algorithms to filter out these fall-like ADLs so that they are not incorrectly classified as falls.

3. Methodology

For our algorithm, we chose to use the basic threshold approach with a test to check for inactivity, as was used in [1] and to utilize the algorithms from [2] to filter out fall-like ADLs. We hoped that by combining the approaches of these different systems, we would find improved specificity without compromising sensitivity.

We chose to use mobile phones as our fall detection device since they are ubiquitous, rather inexpensive, and easy to "wear." Furthermore, a mobile phone is equipped to call for help if necessary, even if the user is unable to interact with it, perhaps as a result of a fall. Additionally, we chose to use the Android platform because it is open-source and written in Java.

The Android application is straight-forward to use. Once inside the app, the user presses the "Start Monitoring" button to start the service and may then leave the app and continue to use their phone without stopping the service (in order to stop, the service must be explicitly stopped by the user). The service has an event listener registered with the

SensorManager so that whenever a change in acceleration is detected, the onSensorChanged() method is called. If a fall is detected, the system starts an alarm which starts a 30 second countdown and asks the user if they fell. The user may respond with "Yes, I fell." or "No, I did not fall." If the user selects "Yes, I fell," or if the countdown runs out, the alert class notifies the emergency contact. If the user selects "No, I did not fall," an alert is not issued to the emergency contact.

We used some of the source code that Gonzales provided in [1] in designing our app, with some of our own edits made to the code and the algorithms. We edited the inactivity detection so that it only runs when there has been a possible and so that the inactivity has to be detected within 5 seconds after the potential fall. These edits were intended to reduce wasted computations and to ensure that we only detect inactivity directly after a possible fall; any inactivity that does not occur directly after a possible fall is irrelevant to our algorithm.

Having discovered that most other systems use higher UFTs than the 1.6g used in [1], we tested three different UFTs: the original 1.6g, 2g, and 2.5g. From our testing, we concluded that the 2g threshold yielded the best results and consequently used that UFT for our algorithm.

We also implemented and incorporated in our system the filtering algorithms used in [2] to distinguish certain ADLs from falls. This novel approach used in [2] was very unique and had performed well in tests. We hoped that by combining these filtering algorithms with our edited form of the algorithms from [1], we would find increased specificity without compromising sensitivity.

We chose to use the UI sampling rate, which is 60 ms, or 16 2/3 Hz. We also implemented linear interpolation to calculate a sample directly between any two neighboring ones; the interpolation effectively yields a 30 ms sampling rate with the smoothness of the 60 ms rate. Using the UI sampling rate with linear interpolation yielded a rather smooth graph without a lot of noise, and yet it was fast enough that the peaks and troughs of falls and other activities were all present.

To see how the UI sampling rate compares to both a slower and a faster rate, observe the figures below. The 20 ms (GAME) sampling rate yields too much noise in the readings; the RSS values seem to jump around inexplicably, especially in the free fall interval. The 200 ms (NORMAL) sampling misses some of the peaks and/or troughs of falls and activities performed. The 60 ms (UI) sampling rate is fast enough that the peaks and troughs of falls and activities are present and slow enough that there's not a lot of superfluous noise.

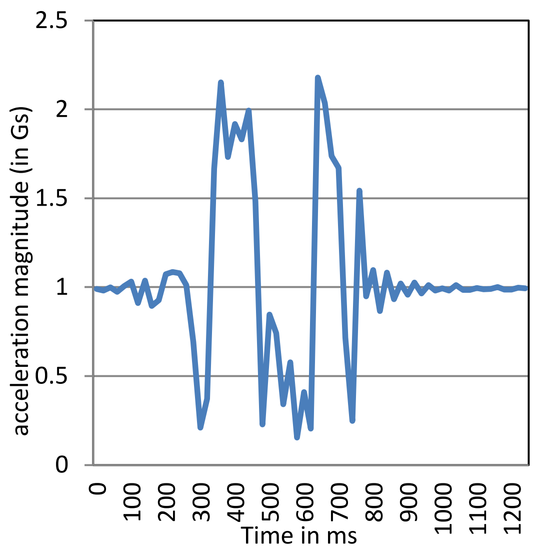

Figure 2-Jumping with GAME (20 ms) sampling rate

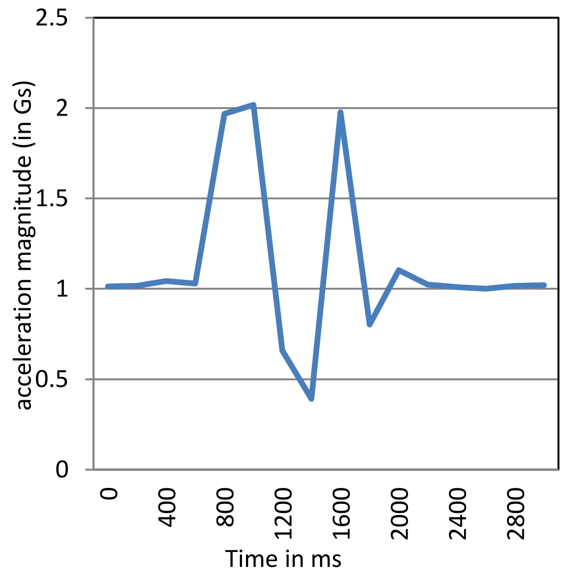

Figure 3-Jumping with NORMAL (200 ms) sampling rate

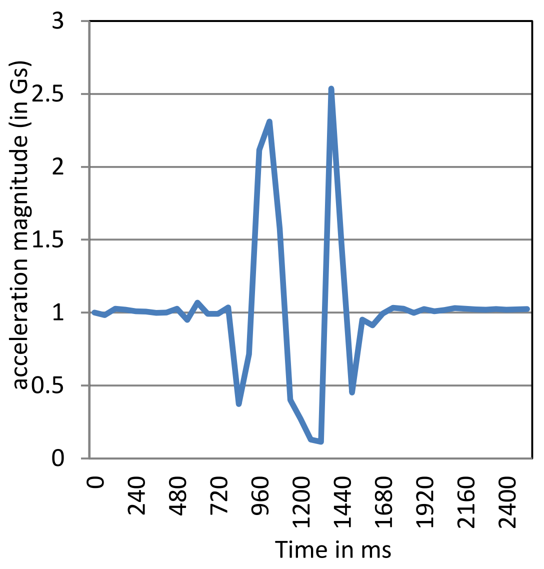

Figure 4-Jumping with UI (60 ms) sampling rate

4. Testing.

For our testing, we used a dummy that we got from the fire department as well as a volunteer who performed some of the less dangerous falls so that we could compare results from a human to the results we received with the dummy.

The dummy helped quite a bit with testing. He is used by the fire department for their training programs. He is 6'1" and weighs 180 pounds. He was designed to have the height and weight of an average adult male. For some types of falls, such as straight forward, using a dummy instead of a volunteer can provide more realistic results since a volunteer may be worried to injure himself and therefore fall somewhat cautiously.

However, there are limits to the effectiveness of testing with a dummy. Although he approximates a human's weight distribution, his joints are very different, especially his knees. Some of the falls did not yield very useful results when tested with the dummy. The sideways falls and rolling-out-of-bed in particular did not work very well. Often, the dummy's legs would end up twisting and crumpling, resulting in some bad data. For this reason, we ended up using a volunteer performing falls to analyze the accuracy of the system for detecting sideways falls and falls from rolling-out-of-bed.

After testing both left and right sideways falls with the phone in each of the left and right pockets, we discovered that sideways falls are successfully detected only if the phone is in the pocket opposite to the side to be fallen on. In other words, left falls are detected if the phone is in the right pocket, and right falls are detected if the phone is in the left pocket.

Gonzales et al noted having some similar issues with one of the phones they used for testing in [1]. After breaking one phone during testing, they had to get another phone to use, and they noted that the new phone gave extremely low sensitivity for lateral falls towards the side where the phone was located. [1] Gonzales et al proposed that their phone may have had some internal flaw that cause it not to respond properly when there was direct impact to it. [1]

This explanation seems consistent with the problems we encountered as well. In fact, not only did we have trouble with any lateral falls that involved impact with the phone, but we also had problems with forward and backward falls that ended in direct impact to the phone. Our system could not detect forward falls when the phone was in the front pocket and could not detect backward falls when the phone was in the back pocket. However, if the phone was in the back pocket, the forward falls were detected quite well, and if the phone was in the front pocket, the backward falls were detected quite well.

Clearly, this issue had a severe negative impact on the sensitivity of the system since it meant that many of the falls that we tested were not detected. Our overall sensitivity was 64.2%. However, if we exclude the results of falls that ended with direct impact on the phone and then calculate the sensitivity, it is much higher—92.2%.

We also tested many different activities in addition to falls with our volunteer so that we could assess our program's specificity. We tested activities of daily living (ADLs) that we knew might register false positives, or simply that are very common and therefore necessary to test for thoroughness. We referred to the list of ADLs in [3] when compiling our list of activities to test. And we ended up testing the following ADLs: bending, bending-pick-up, jogging, jumping, lying-bed, rising-bed, sit-bed, sit-chair, sit-sofa, sit-air, squatting-down, walking, and walking backwards. Admittedly, our tests were limited—although we tested a great variety of ADLs, we had only one volunteer to do so. However, with our limited tests, all ADLs except jogging and one instance of sit-bed were consistently filtered out. Jogging presented more of a challenge because it looks very similar to a fall.

Some of the time, the inactivity detection would filter out jogging, and some of the time it would not. We discovered that the inactivity detection filters out jogging if one jogs and comes to a gradual stop because the tester is still active and moving after the final peak produced by jogging. See figure 5. Notice how the peaks get gradually smaller as the activity finishes. The activity depicted in this figure was filtered out and did not set off a false alarm.

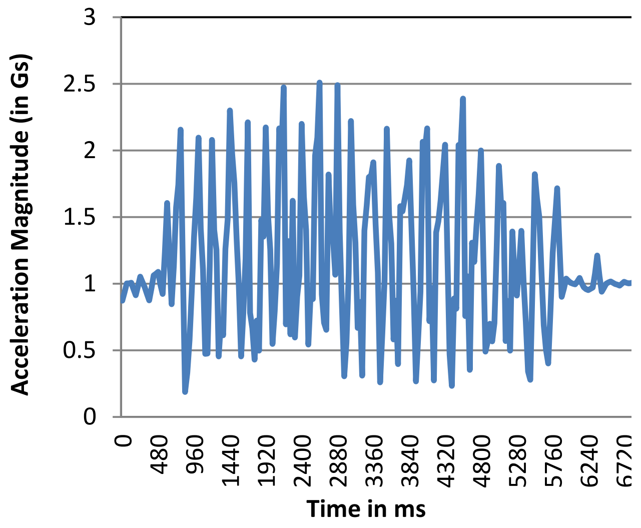

Figure 5-Jogging with a gradual stop

But if one jogs and stops abruptly, then the inactivity detection does not filter out the jogging because the tester is inactive and the final peak produced by the jogging looks, for all intents and purposes, like a fall. See figure 6. Notice that the peaks do not gradually get smaller, but instead we have a sudden stop in activity right after a large peak. The activity depicted in this figure was not filtered, but rather set off a false alarm.

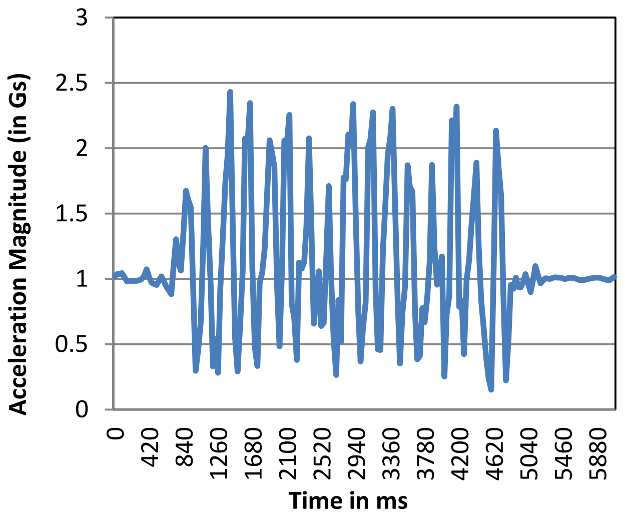

Figure 6-Jogging with an abrupt stop.

Even considering the trouble that jogging presented, our overall specificity was still 94.2%.

5. Conclusion/Future Work

One thing that ought to be researched in more depth in the future is the issue with particular models of phones not responding well to impact. It is significant to find out how common this problem is, and whether it is specific to particular phones or models, also, whether it is only an issue in older phones, or if newer phones have this problem too. Depending on the extent of this problem, it may actually not be feasible to use a phone for fall detection. However, since only one of the papers we read mentioned this problem, we would suppose, and hope, that this is not a very common problem.

Much more testing is also needed. It is difficult to test a fall detection system because real falls are often quite different from simulated ones. Also, it is hard to find volunteers to help with fall testing because simulating falls can be uncomfortable and there is always the possibility of injury. However, some real-world testing and data may be attained once our app is ready to be released. Once that happens, we will be able to have tests of regular daily activities, many ADLs and possibly eventually, though regrettably, some real falls.

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

[3] S. Abbate, M. Avventui, P. Corsini, J. Light, and A. Vecchio. Monitoring of Human Movement 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, p. 1-20.

[4] Bagalà F, Becker C, Cappello A, Chiari L, Aminian K, et al. (2012) Evaluation of Accelerometer-Based Fall Detection Algorithms on Real-World Falls. PLoS ONE 7(5): e37062. doi:10.1371/journal.pone.0037062

[5] A.K. Bourke, P. van de Ven, M. Gamble, R. O'Connor, K. Murphy, E. Bogan, E. McQuade, P. Finucane, G. ÓLaighin, J. Nelson. Evaluation of waist-mounted tri-axial accelerometer based fall-detection algorithms during scripted and continuous unscripted activities. Journal of Biomechanics - 16 November 2010 (Vol. 43, Issue 15, Pages 3051-3057, DOI: 10.1016/j.jbiomech.2010.07.005)

[6] 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

[7] Gazmend Bajrami. Activity Identification for Gait Recognition Using Mobile Devices. Master's thesis, Gjovik University College, 2011.

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