Smartphone-based Fall Detection Proposal, 6/4/2011

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.

2. 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, with near 100% sensitivity and specificity, 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.

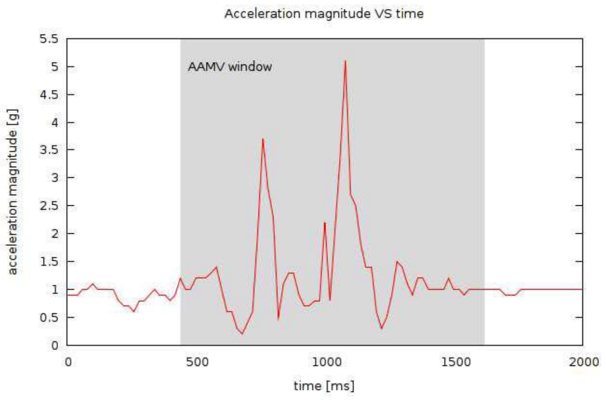
3. Related Work:

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 upon impact there is a sharp spike in the person's acceleration magnitude.

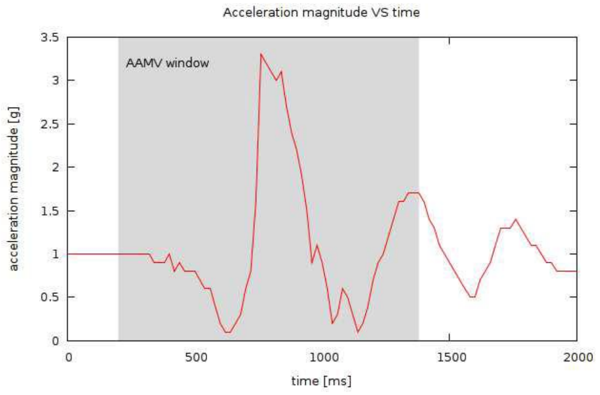
A commonly used approach to detect falls is to use two pre-determined threshold values to test the RSS values. The first threshold is the lower fall threshold (LFT), which is set to some value between 1g and 0g, and the second threshold is the upper fall threshold (UFT), which is set to an empirically 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.

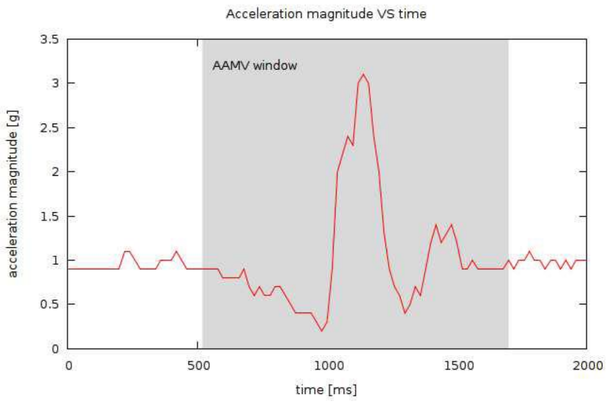
Abbate et al. [2] proposed an interesting and unique method for decreasing false positives. They noted that there are some normal activities of daily living (ADLs) that exhibit kinetic peaks similar to those that are present in falls [2]. Therefore, when using the threshold technique, which is a very common method of detecting a fall, these ADLs are likely to register false positives. Note that in each of the graphs in the figure below, the acceleration magnitude drops significantly below 1g, and thus typically below the LFT, and then later spikes above 3g, which is approximately the UFT. Thus, when detecting a fall based solely on thresholds, all of these scenarios would likely be classified as falls.



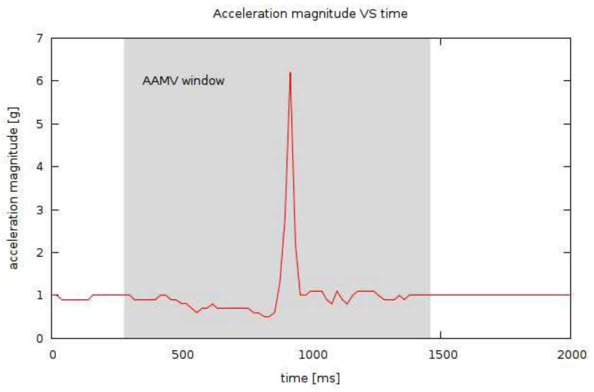
(a) Real Fall



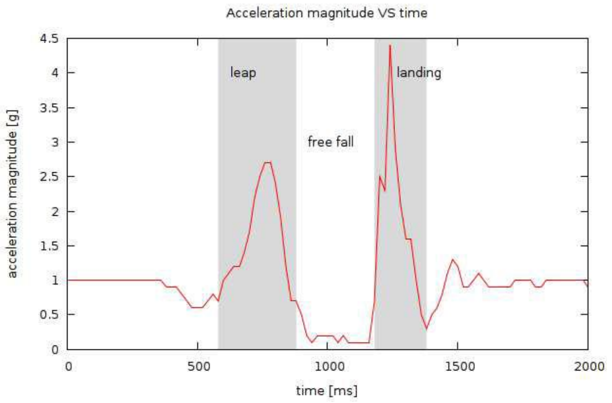
(b) Class A (sitting/lying on an elastic surface)



(c) Class A (sitting/lying on a soft surface)



(d) Class B (sitting quickly on a medium/hard surface)



(e) Class C (jumping)

Figure 3.1 From [2].

Abbate et al. defined some algorithms that can be used to filter out these fall-like ADLs so that they are not misclassified as falls.

$$AAMV = \sum_{i \in W} \frac{|acc_{i+1} - acc_i|}{number\ of\ samples\ in\ W}$$

Equation 3.1 From [2].

Falls exhibit quick and numerous acceleration changes while class A ADLs exhibit slow variations and class B ADLs exhibit few variations [2]. Because of this fact, Abbate et al. were able to use a measure called the average acceleration magnitude variation index to differentiate between class A and B ADLs and real falls. The value of the AAMV will be proportional to both the speed with which the acceleration magnitude changes and to the number of peaks present in the window, and so the AAMV should be higher for real falls than for class A and B ADLs [2].

- $FFI > 100ms$

- $FFAAM < 0.5g$

Equation 3.2 From [2].

Class C ADLs, on the other hand, have comparable AAMV values to real falls and instead are filtered based on their unique, defined shape, which consists of a leap, free fall, and landing [2]. Abbate et al. observed that Class C ADLs have a longer free fall interval (FFI) and smaller free fall average acceleration magnitude (FFAAM) than do real falls, and they empirically determined threshold values for the FFI and FFAAM that can be used to filter out the class C ADLs [2].

4. Proposed Approach and Present Status:

Initially, our algorithm consisted of the basic threshold approach combined with a test to check for inactivity, as was used in [1]. However, this resulted in low specificity values, with too many false positives to be useful. After researching additional methods for increasing our specificity, we chose to utilize the algorithms from [2] to filter out fall-like ADLs. Our hope is that by combining the approaches of these different systems, we will find improved specificity without compromising sensitivity.

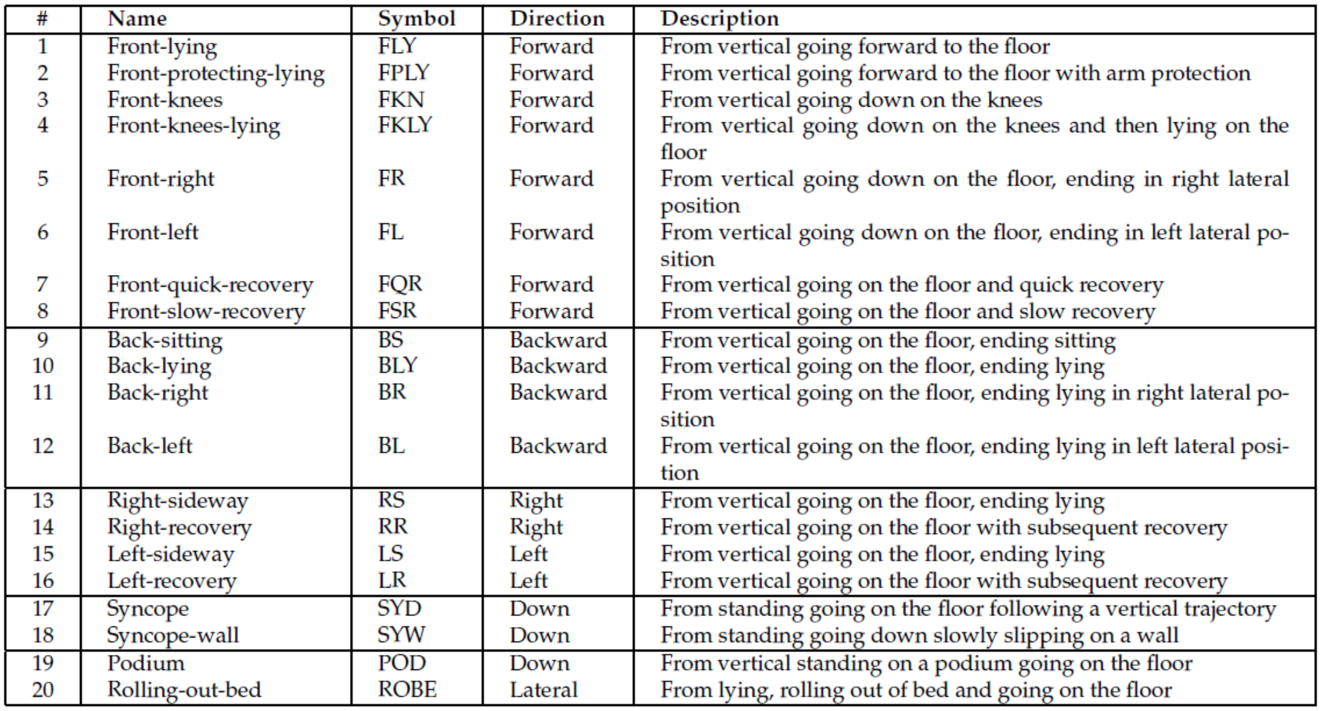
We chose to use mobile phones as our fall detection device. They are ubiquitous, rather inexpensive, and easy to “wear.” Furthermore, a mobile phone is equipped to call for help if necessary, even if the user is unable to interact with it, perhaps as a result of a fall. Additionally, we chose to use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

We currently have a working Android app that has been programmed with an implementation of our algorithm in Java, complete with the addition of the new algorithms we decided to implement from [2]. We originally designed our app using the source code that Gonzales provided [1]. The 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.

One thing that must be considered is the sampling rate to be used—it determines how frequently we take readings from the accelerometer. Four options for the sampling rate are provided by the Android API: NORMAL, UI, GAME, and FASTEST. These provide a sampling rate of 200ms, 60ms, 20ms, and 10ms respectively. Currently our app uses a sampling rate of 200ms because a slower sampling rate preserves the battery life. Although, we do not know how much or how little impact a faster sampling rate might have on the battery life. It will be necessary to experiment with different sampling rates in order to observe the effect it has on the battery life. Likewise, it is unknown how much or how little effect the frequency of the sampling rate has on the overall accuracy of the algorithm. However, the algorithms from [2] that we recently added to our system in order to increase our specificity were designed with a 20ms sampling rate in mind, and in order for them to work properly, it may be necessary to use a faster sampling rate. If possible, it would be beneficial to do some additional research in order to determine the effect a faster or slower sampling rate has on the algorithm's accuracy.

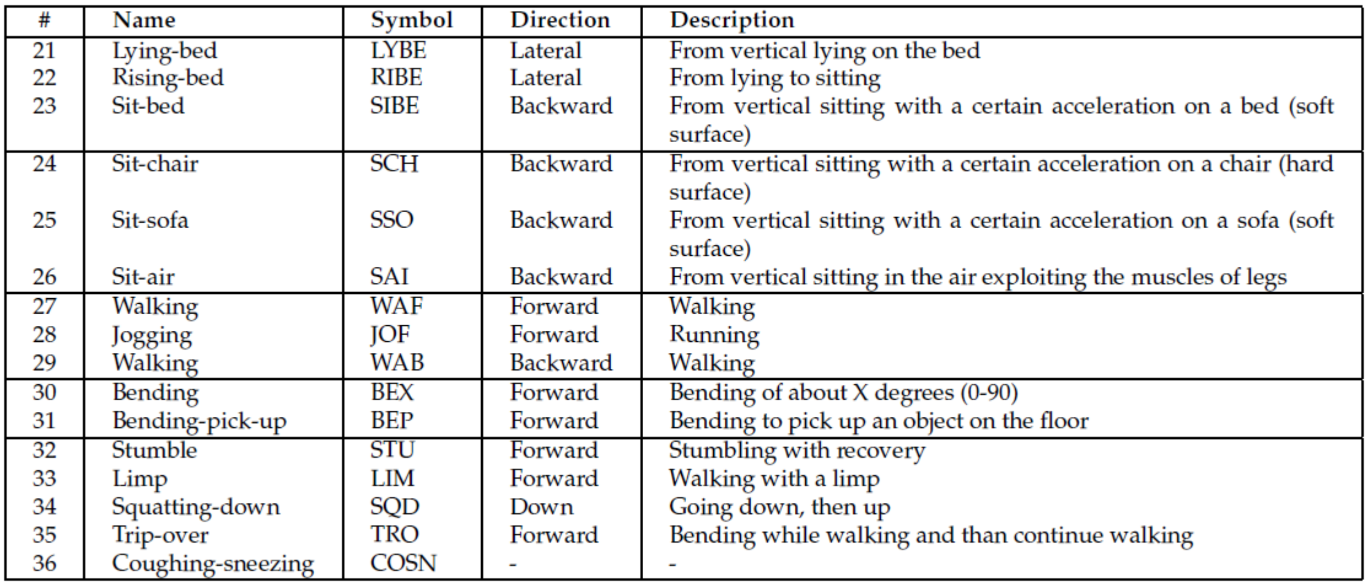
5. Evaluation:

As we have already noted earlier, there are several normal ADLs that exhibit kinetic peaks similar to those that are present in falls and that may register false positives [2]. When testing the fall detection system, it is therefore essential to simulate many different activities in addition to falls. Abbate et al. [3] emphasized the importance of a standard set of trials for researchers to use in order to facilitate comparison of various fall detection systems. They included a table that listed common trial scenarios that they concluded ought to be tested. It is indeed crucial for different fall detection systems to be able to be compared. And therefore, we intend to use the listing of scenarios from [3] to construct our tests.



#	Name	Symbol	Direction	Description
1	Front-lying	FLY	Forward	From vertical going forward to the floor
2	Front-protecting-lying	FPLY	Forward	From vertical going forward to the floor with arm protection
3	Front-knees	FKN	Forward	From vertical going down on the knees
4	Front-knees-lying	FKLY	Forward	From vertical going down on the knees and then lying on the floor
5	Front-right	FR	Forward	From vertical going down on the floor, ending in right lateral position
6	Front-left	FL	Forward	From vertical going down on the floor, ending in left lateral position
7	Front-quick-recovery	FQR	Forward	From vertical going on the floor and quick recovery
8	Front-slow-recovery	FSR	Forward	From vertical going on the floor and slow recovery
9	Back-sitting	BS	Backward	From vertical going on the floor, ending sitting
10	Back-lying	BLY	Backward	From vertical going on the floor, ending lying
11	Back-right	BR	Backward	From vertical going on the floor, ending lying in right lateral position
12	Back-left	BL	Backward	From vertical going on the floor, ending lying in left lateral position
13	Right-sideway	RS	Right	From vertical going on the floor, ending lying
14	Right-recovery	RR	Right	From vertical going on the floor with subsequent recovery
15	Left-sideway	LS	Left	From vertical going on the floor, ending lying
16	Left-recovery	LR	Left	From vertical going on the floor with subsequent recovery
17	Syncope	SYD	Down	From standing going on the floor following a vertical trajectory
18	Syncope-wall	SYW	Down	From standing going down slowly slipping on a wall
19	Podium	POD	Down	From vertical standing on a podium going on the floor
20	Rolling-out-bed	ROBE	Lateral	From lying, rolling out of bed and going on the floor

Table 5.1 Actions to be detected as falls. From [3].



#	Name	Symbol	Direction	Description
21	Lying-bed	LYBE	Lateral	From vertical lying on the bed
22	Rising-bed	RIBE	Lateral	From lying to sitting
23	Sit-bed	SIBE	Backward	From vertical sitting with a certain acceleration on a bed (soft surface)
24	Sit-chair	SCH	Backward	From vertical sitting with a certain acceleration on a chair (hard surface)
25	Sit-sofa	SSO	Backward	From vertical sitting with a certain acceleration on a sofa (soft surface)
26	Sit-air	SAI	Backward	From vertical sitting in the air exploiting the muscles of legs
27	Walking	WAF	Forward	Walking
28	Jogging	JOF	Forward	Running
29	Walking	WAB	Backward	Walking
30	Bending	BEX	Forward	Bending of about X degrees (0-90)
31	Bending-pick-up	BEP	Forward	Bending to pick up an object on the floor
32	Stumble	STU	Forward	Stumbling with recovery
33	Limp	LIM	Forward	Walking with a limp
34	Squatting-down	SQD	Down	Going down, then up
35	Trip-over	TRO	Forward	Bending while walking and than continue walking
36	Coughing-sneezing	COSN	-	-

Table 5.2 Activities that must not be detected as falls. From [3].

It is also important to note that threshold values are frequently determined based on simulated falls and as a result are not necessarily accurate for real-world falls [4]. Many fall detection systems have only been tested using simulated actions and falls. And when tested in real-world circumstances, they are often not as accurate as was claimed based on the simulations [4]. Thus, in evaluating the accuracy of a fall-detection algorithm, it is necessary to have some real-world testing in addition to using simulations.

Because our system is being developed as a free app for Android, we should be able to gather real-world data, involving both ADLs that set off false positives as well as real falls. This data could assist not only in assessing the accuracy of our current system, but also in improving the algorithms used.

6. Timeline:

Task 1	Review code.		Week 2 June 5
Task 2	Correct logic errors.		Week 2 June 5-7
Task 3	Test scenarios using GAME sampling rate.	Observe accuracy and battery lifetime.	Week 3-7/Week 3-4
Task 4	Test scenarios using UI sampling rate.	Observe accuracy and battery lifetime.	Week 3-7/Week 4-6
Task 5	Test scenarios using NORMAL sampling rate.	Observe accuracy and battery lifetime.	Week 3-7/Week 6-7
Task 6	Review results of diff. sampling rates.	Determine which sampling rate to use.	Week 8
Task 7	Real-world testing.	Release free app.	Week 8

7. 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. Vecchi. 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