Fall Detection Using Single Tri-Axial Accelerometer

Suleman Belal Kazi, Sherjeel Sikander, Sadia Yousafzai, Dr. Suleman Mazhar
Ghulam Ishaq Khan Institute of Engineering Sciences and Technology
Topi, Swabi, Pakistan

Abstract — This paper describes a mobile phone based fall detection algorithm using the phone’s built-in accelerometer which can detect falls with a high degree of accuracy. The fall detection application developed for phone can then notify predefined guardians or emergency contacts along with sending the victim’s GPS coordinates displayed on a map, providing timely medical help for possible injury. The algorithm has been tested, and the results are also included in this paper.

Keywords — accelerometer, mobile phone, fall detection

I. INTRODUCTION

Falls contribute greatly to accidental injuries worldwide. Each year around 37.3 million victims of falls require medical attention due to physical injuries [1]. Roughly 10% to 20% of falls can cause fractures [2]. The elderly are a demographic particularly vulnerable to falls. 30% people with ages above 65 and 50% people with ages above 80 fall each year [3]. Hospitalizations due to falls in older people are five times more likely than young people [4]. The number of elderly people in the world is expected to reach 2 billion by 2050 [5].

There are different approaches for fall detection. The use of multiple accelerometers [6], wearable sensors [7], cameras [8] or vibration based detectors [9]. Among these, accelerometers are very suitable for the detection of falls. Studies have also been done on accelerometer data sets for human activity recognition [10-11]. The tri-axial accelerometer present in the mobile phone used has three axes: X, Y and Z. Each axis returns the acceleration in the direction of the axis in terms of the acceleration due to gravity (1g = 9.8 m/s2). The three axes are shown in Fig. 1.

A cell phone is a ubiquitous device that people have with them all the time. New smart phones being produced have built in accelerometers. Thus cell phones can prove to be good devices for fall detection since no additional hardware is necessary. Moreover a cell phone serves as an ideal device to automatically notify contacts in case of a fall being detected. Some algorithms have been developed to use cell phones in this manner [12-14]. In this paper we present a computationally low cost algorithm that can detect falls in real time.

II. SYSTEM ARCHITECTURE

Initially the fall detection algorithm was developed and tested on a Nintendo Wii Remote [15] (a gaming console controller with a built in accelerometer) connected with a C# desktop application for ease of debugging and graph plotting. It was later ported to a Windows 8 Phone device with added features of locating the victim through GPS coordinates and sending help alerts through email. It was tested on two models, a Nokia Lumia 920 (1.5 GHz Processor, 1 GB RAM) and Nokia Lumia 620 (1 GHz Processor, 512 MB RAM), which demonstrates that our algorithm can be run not only on high end consumer cell phones but also relatively less powerful ones.

The application was developed using Microsoft’s Windows Phone Software Development Kit (SDK), which is required for development on the Windows 8 Phone platform. The signal processing was done in C# with the layout designed in XAML. The Math .NET open source library was used for digital filtering.

III. METHODOLOGY

The fall detection algorithm operates in a series of steps. In the usual user position for our mobile application, the positive Y-axis points vertically downward with the X and Z axes perpendicular to it. Hence while stationary, the Y axis reads a value of 1g whereas the X and Z axes read a value of 0g. The steps included for fall detection as proposed by our own model in Fig. 2 are as follows:

1. Readings from the accelerometer’s axes are stored at a rate of 100 Hz. A rate of 100 Hz ensures sufficient sampling rate. This data consists of a three dimensional acceleration vector \( \mathbf{A} \) with readings from the X, Y and Z axes as its components \( A_x, A_y, \) and \( A_z \) respectively.

2. Each second (i.e. for 100 samples of data), a high pass filter is applied to the accelerometer data vector \( \mathbf{A} \) yielding filtered values \( A_f \) and a low pass filter is also applied yielding values \( A_l \). The high pass filtered values correspond to acceleration values due to the movement of the cell phone (and hence the user), whereas the \( A_l \) values correspond to

![Fig. 1. Cartesian co-ordinate system of the accelerometer](image-url)
acceleration due to gravity. The filters are Finite Impulse Response (FIR) [16] online, stable and causal filters implemented in the Math.NET library. FIR filters are based on Fourier series with both the high pass and low pass filters of the sixth order. After multiple test results, the high pass filter is set to a cutoff of 2 Hz, and the low pass is set to a cutoff of 1.5 Hz for best fall detection results.

5. If $|A_h|_1$ is greater than $E_{th}$ the algorithm waits for three seconds. The three seconds allow for the transient acceleration changes due to the fall to be over. After three seconds, the algorithm checks the orientation of the accelerometer for determining the posture of the user. This helps separating high energy activities like running and jumping from actual falls. The only angle of interest is the angle that the accelerometer makes with the vertical, assuming that the Y axis was initially vertical. The angle is calculated from 100 samples of $A_i$ using trigonometric function as:

$$\theta_i = \cos^{-1}\left(\frac{A_{iyi}^2}{\sqrt{A_{ixi}^2 + A_{iyi}^2 + A_{izi}^2}}\right)$$

6. Max-wins voting is used to determine the user orientation. Each value of $\theta_i$ is compared with a certain empirically determined angle threshold, $\theta_{th}$. If $\theta_i$ is greater than the threshold, it counts as a win, and if it is less it counts as a loss. If the number of wins is greater than the number of losses, this means the user is not standing, and has fallen. In this case the algorithm is ready to proceed to the next step.

7. In practice, there are many falls in which the user might not be hurt, and the fall might be minor so that the user gets up and starts walking again. In this case the alarm does not need to be triggered by the algorithm. Hence in order to prevent a false positive fall from registering, the algorithm again checks the value of $|A_h|_1$ and of $\theta_i$ for five seconds. If in those five seconds the value of $|A_h|_1$ exceeds a certain empirically determined threshold $E_{m}$ (i.e. the user is again engaged in a high energy activity) and the value of $\theta_i$ indicates that the user is standing as described above, the alarm is cancelled and a fall is not detected.

8. At this stage we are almost certain that a fall has occurred. However there is still one false positive that remains to be accounted for, i.e. if a user accidentally drops his phone. To account for this we added a five second ‘grace period’ for the user in which the phone first beeps and shows a pop up notifying the user that an alarm has been sounded. The user can then cancel the alarm if he wishes. If the alarm is not cancelled within a pre-specified time, the phone will sound an on-spot alarm along with sending the persons GPS coordinates with a message to predefined people.

IV. RESULTS

Fig. 3 to Fig. 7 are the results of the algorithm steps applied to the raw accelerometer data (obtained using C#) as shown in Fig. 2. The blue, red and green lines in Fig. 3 to Fig. 5 indicate real time values of X, Y and Z axis respectively. The accelerometer (phone) is attached to the subject’s waist using a belt, with the Y-Axis pointing downwards.

The algorithm was tested with three test subjects wearing the cell phone at the waist (clipped to the belt) and falling in a variety of positions. The results are summarized in Table I.
Table I. Experimental Results

<table>
<thead>
<tr>
<th>FALL TESTS</th>
<th>NUMBER OF TESTS</th>
<th>CORRECT</th>
<th>INCORRECT</th>
<th>ACCURACY (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall forward/backward</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>Fall sideways</td>
<td>20</td>
<td>19</td>
<td>1</td>
<td>95</td>
</tr>
<tr>
<td>TESTS FOR FALSE POSITIVES</td>
<td>NUMBER OF TESTS</td>
<td>CORRECT</td>
<td>INCORRECT</td>
<td>ACCURACY (%)</td>
</tr>
<tr>
<td>Stumbling and getting up</td>
<td>20</td>
<td>18</td>
<td>2</td>
<td>85.5</td>
</tr>
<tr>
<td>Jumping up and down</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>Dropping Phone</td>
<td>20</td>
<td>3</td>
<td>17</td>
<td>15</td>
</tr>
<tr>
<td>Sitting Down Suddenly</td>
<td>20</td>
<td>17</td>
<td>3</td>
<td>85</td>
</tr>
<tr>
<td>Jumping and laying on bed</td>
<td>20</td>
<td>2</td>
<td>18</td>
<td>10</td>
</tr>
</tbody>
</table>

As can be seen in Table I, the algorithm has a good detection rate for actual falls, and has low false negatives except in the case of jumping to a lying position in a bed or dropping the phone. This is expected since the scenario is extremely similar to a fall and hence is not handled that well. However even in this scenario, the application provides a grace period of 5 seconds, in which the subject is able to cancel the alarm in each case of a false positive.

Fig. 3 shows the raw data collected during an event of fall. The graphs are plotted using C# desktop application on runtime along with the collection of raw data. Fig. 4 and Fig. 5 demonstrate the low frequency and high frequency components of raw data as shown in Fig. 3 respectively, where low frequencies correspond to gravity and high frequencies
correspond to user movement. Furthermore, Fig. 6 highlights the change of angle with the vertical due to post fall orientation of the user.

The results showcase high degree of accuracy in detection of falls. Although some false positives are inevitable due to dropping of phone or simulating a fall by jumping on a bed; still in practical cases the algorithm demonstrates an accuracy of more than 85%.

CONCLUSION

The strategy to use a cell phone as a portable fall detector shows great potential. With growing number of cell phone users, the application can easily be utilized by families of elderly people to stay alert regarding any possible physical injury caused by falls. The proposed solution is simple, using only low computational cost signal processing techniques through built in accelerometer of the phone. By placing a phone at waist of the user, the mobile application can detect and alert specified contacts in case of a fall. It not only provides portability and efficiency but also significantly reduce the costs related with healthcare.

In the future more portable sensors (e.g. body temperature, glucose, blood pressure etc.) can be integrated within such a system to provide a complete telehealth experience. The fall detection algorithm itself can be modified to detect traffic accidents, which mathematically is similar to energy impact of falls. Using a similar algorithm as proposed in Fig. 2, the mobile application can be made to detect a traffic accident by placing the phone in car of the user.

ACKNOWLEDGMENT

We are thankful for the contribution of the National ICT R&D Fund of Pakistan and BISAMIL Lab GIKI who provided financial support for this initiative, and are grateful to the participants who volunteered to test the fall detection algorithm.

REFERENCES


