Fall Detection with Posture recognition on Android Smart phone

Technical Report

December 2012

Abstract:

In the present scenario, there has been increase in the population of the elderly due to availability of better medical facilities. By 2050, 2 billion people will be aged 60 and over. This represents both challenges and opportunities. Many of the old lose their ability to live independently because of limited mobility, frailty or other physical or mental health problems. Many require some form of long-term care, which can include home nursing, community care and assisted living, residential care and long stays in hospitals. Accidents due to fall can prove fatal if unattended. Proactive monitoring of patient becomes a key responsibility in helping patients to continue their daily activities and helps people to keep track of their loved ones. The proposed fall detection mechanism uses the sensors available in a typical smart phone to find out whether an accident has occurred.

https://github.com/BharadwajS/Fall-detection-in-Android

Table of Contents

Introduction

A typical android smart phone is loaded with a variety of sensors. Accelerometer, magnetometer, proximity sensor, light sensors are some of the sensors available which can be used to detect postures and potentially detect events like fall. The test bed phone used for this project is Huawei Ascend y200 .It runs on an 800MHz Cortex-A5 processor. It runs on Android 2.3.6 platform. It has all the sensors mentioned above. It has 165 MB of RAM for the user to install applications. So the application shouldn't occupy much memory so that it doesn't slow down the phone.

Accelerometer used is LIS3DH 3-axis accelerometer which can measure linear acceleration in x, y, z mutually perpendicular axes. The phone is assumed to be in particular position for the algorithm to compute postures. Typically the phone assumed to be in the waist pocket region. The algorithm is tuned to this particular position and thus all computation is based on this assumption.

Algorithm

There are two modules in the algorithm,

- Posture recognition
- Fall detection

For both modules we read the ax, ay, az values from the accelerometer. Android 2.3.6 supports different time delays for reading sensor.

- SENSOR_DELAY_NORMAL - 200,000 us delay
- SENSOR_DELAY_GAME - 20,000 us delay
- SENSOR_DELAY_UI - 60,000 us delay
- SENSOR_DELAY_FASTEST - 0 us delay

We have used SENSOR_DELAY_UI which sets the sampling rate as 16.67 Hz. Then we compute the L2 norm of the ax, ay, az. This is used by posture recognition and fall detection module [1].

In the Posture recognition module, the user postures are classified into three basic postures: sitting, standing and walking. The values of "ay" is applied a threshold to find out the orientation. Using the rate of variation of L2 norm w.r.t mean acceleration due to gravity ($9.8\ m/s^2$) we classify the data into walking or just transition between states [2].

The fall detection module searches for particular pattern in the signal. Figure 1 represents a typical pattern in L2 norm during a fall event

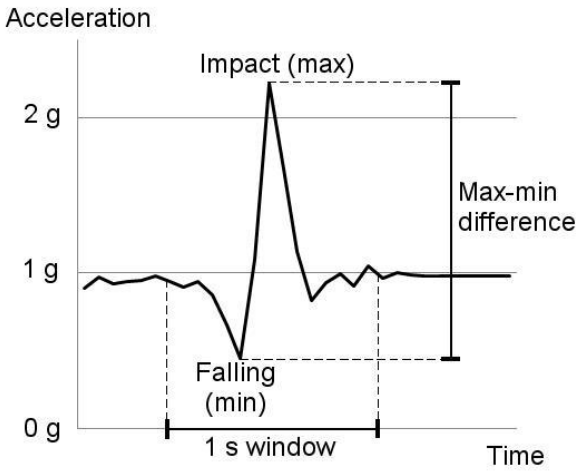


Figure 1: Fall pattern

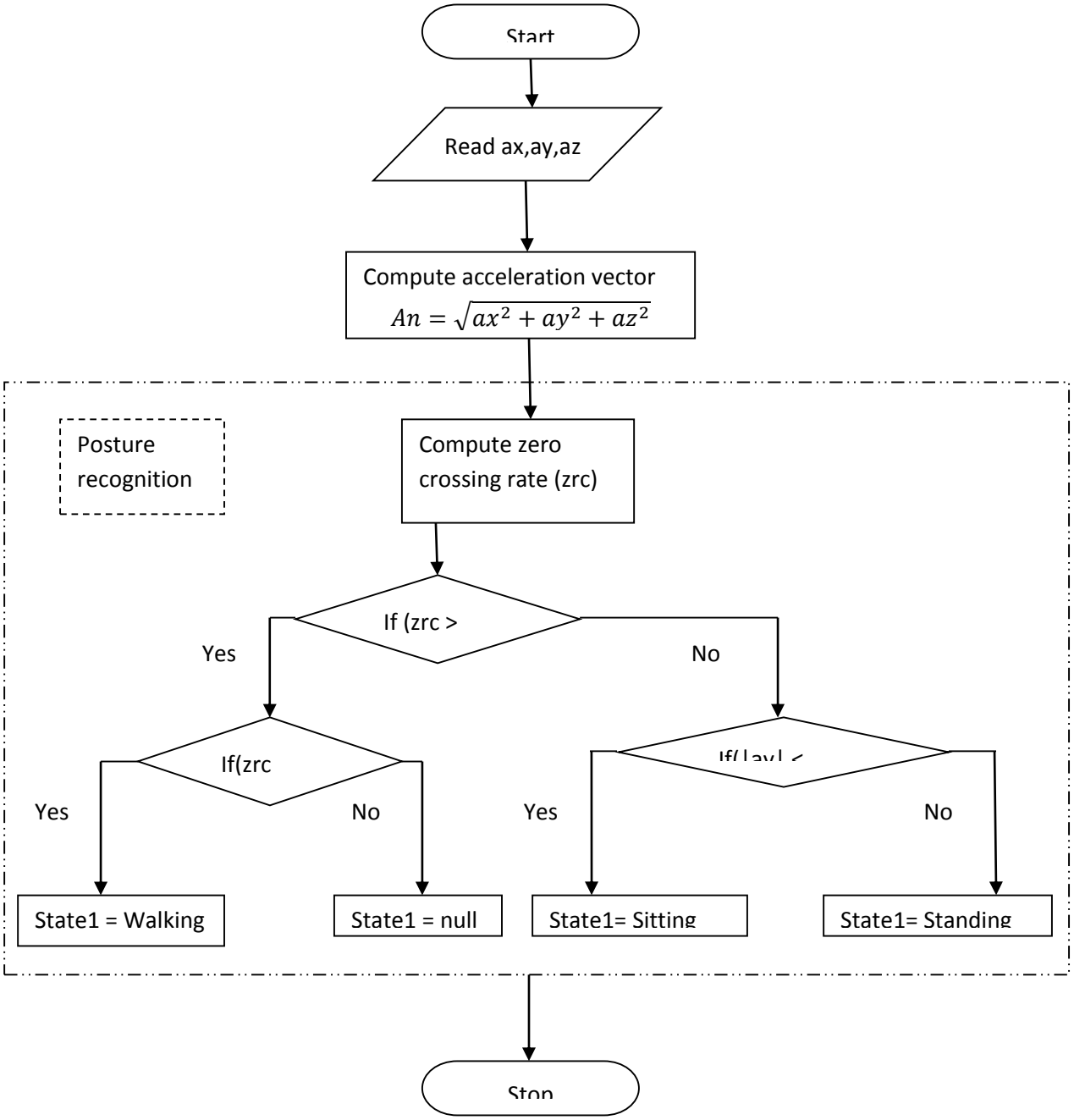
If the difference between consecutive minima and maxima is greater the 2g, the output is decided as a fall.

The final decision on event of a fall is based output of both posture recognition and fall decision module. When a fall is detected, decision from posture recognition module tells us whether it's a false alarm. If the state is still walking, the fall event decision can be discarded. Table 1 summarizes the final decision.

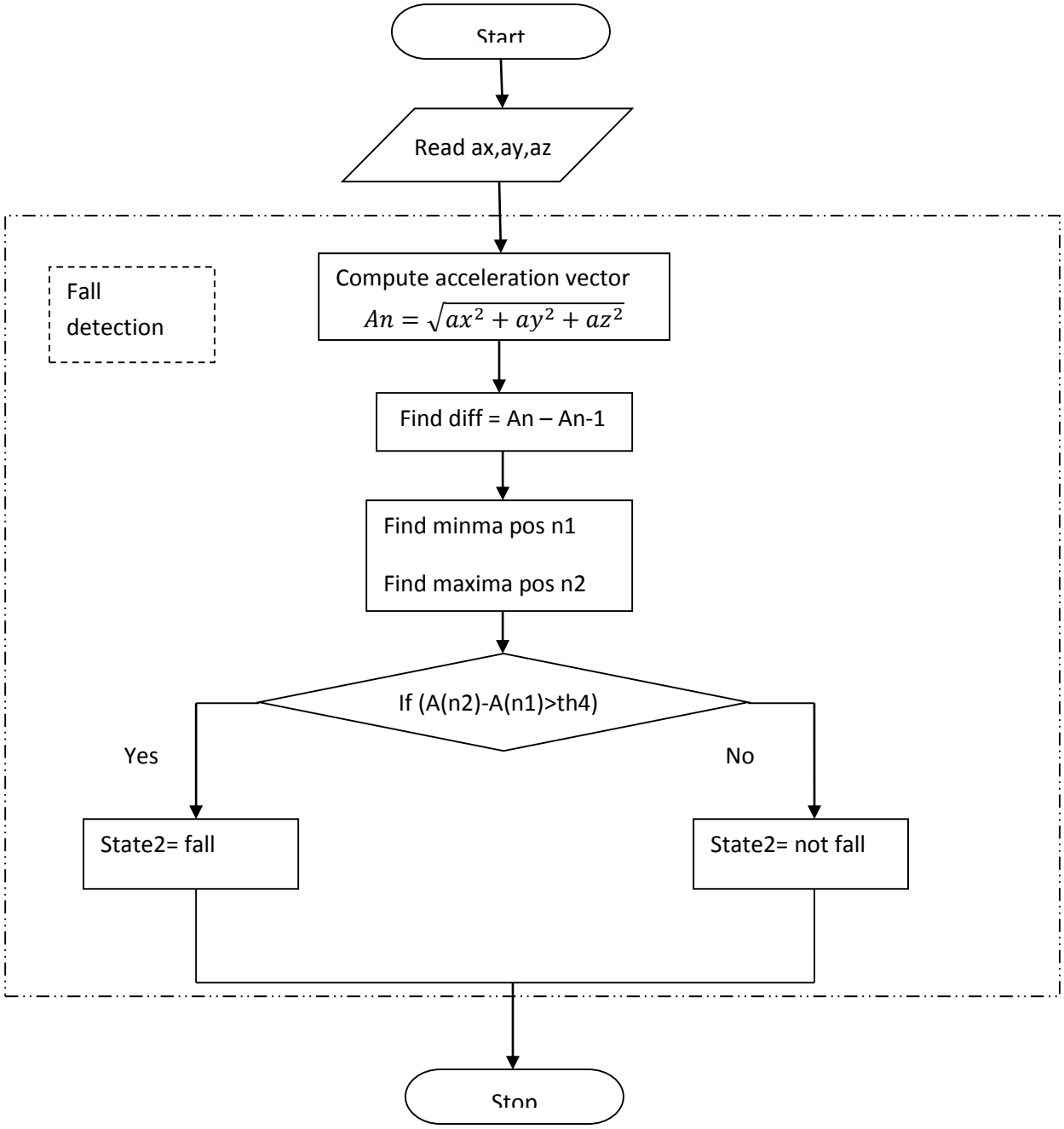
Fall detection	Posture recognition	Decision on occurrence of fall
Fall	Walking	False
Fall	Standing	False
Fall	Sitting	True
Fall	Null state	True

Table 1: Decision on fall

Flow Chart – Posture Recognition



Flowchart – Fall detection



Analysis of the Algorithm

The data from the Accelerometer is stored in buffer of suitable length. In our case, it's fixed as 35 samples. As the sampling rate is fixed as 16.67 Hz the memory of the buffer corresponds to 2.1 sec. Neglecting the processing delay and process preemption, the decision on an event is made after this delay.

Lowering the window size, will affect the decision by the posture recognition algorithm. The reason behind this is that, the zero crossing rate requires a constant oscillation for a particular amount of time to figure out between states of walking and momentary jitter (no state).

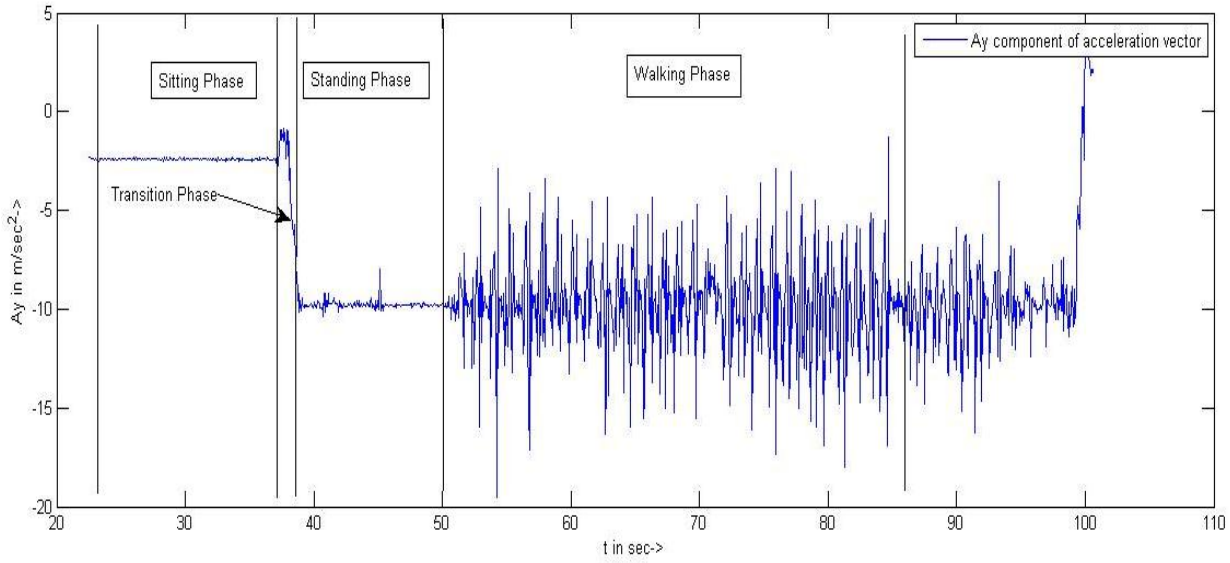


Figure 2: Ay component of Acceleration vector

The transitions and slight variations are regarded as null state, a dummy state where no specific posture can be shown correspondence to. To characterize a state between null and walking to thresholds are used on the zero crossing rate. Zero crossing rate is computed by taking window's worth of L2 norm of acceleration vector samples.

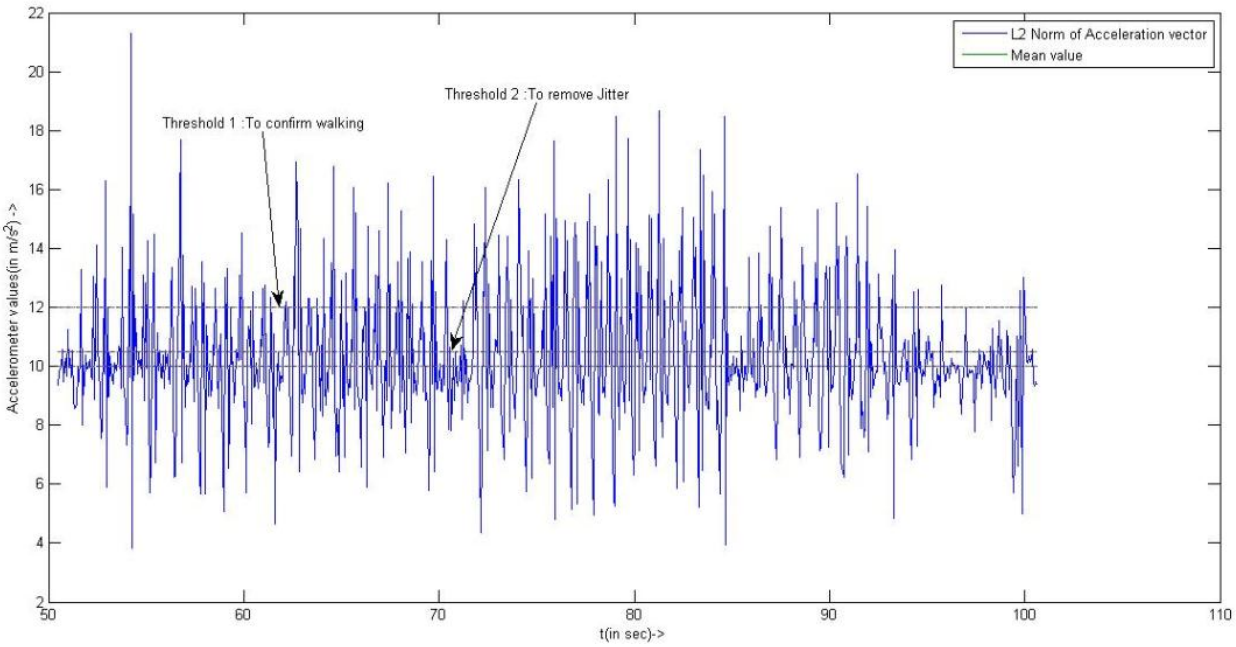


Figure 3: Zero crossing rate

The trend has to greater than Threshold1 to be recorded as walking. Else the variations are discarded. Threshold 2 marks the lower threshold based on which slight variations are removed in the start of the algorithm. Therefore, Threshold1 is the upper threshold and Threshold2 is the lower. The necessity of lower threshold is that transitions between states will be marked as a null state based on lower threshold. We measure the rate of variations of acceleration vector w.r.t $10\ m/s^2$ and we store it as zero crossing rate of acceleration vector. Figure 4 shows output of Posture recognition augmenting the Ay component.

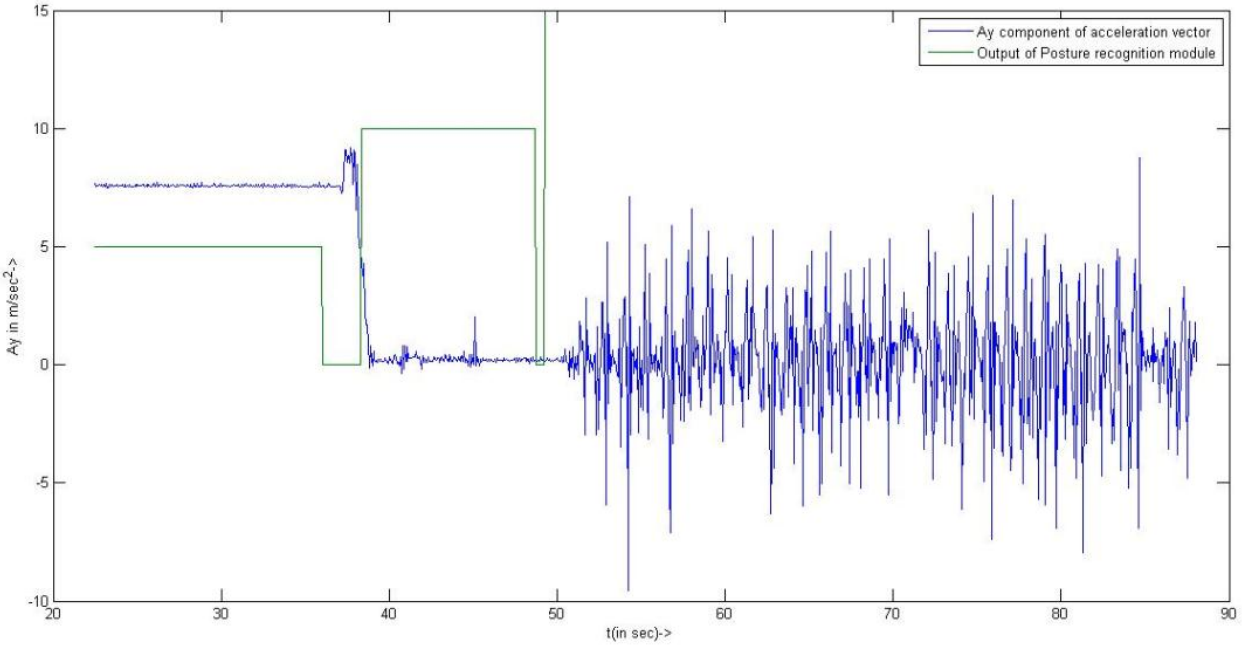


Figure 4: Posture recognition module

Table 2 gives what the values of Posture recognition module means.

State	Value
Sitting	5
Standing	10
Walking	15
Null	0

Table 2: Values for different states

In the Fall detection algorithm, we search for a particular pulse in the L2 norm of acceleration vector. Figure 1 illustrates typical pulse pattern during a fall. The difference between minima and maxima triggers an alarm.

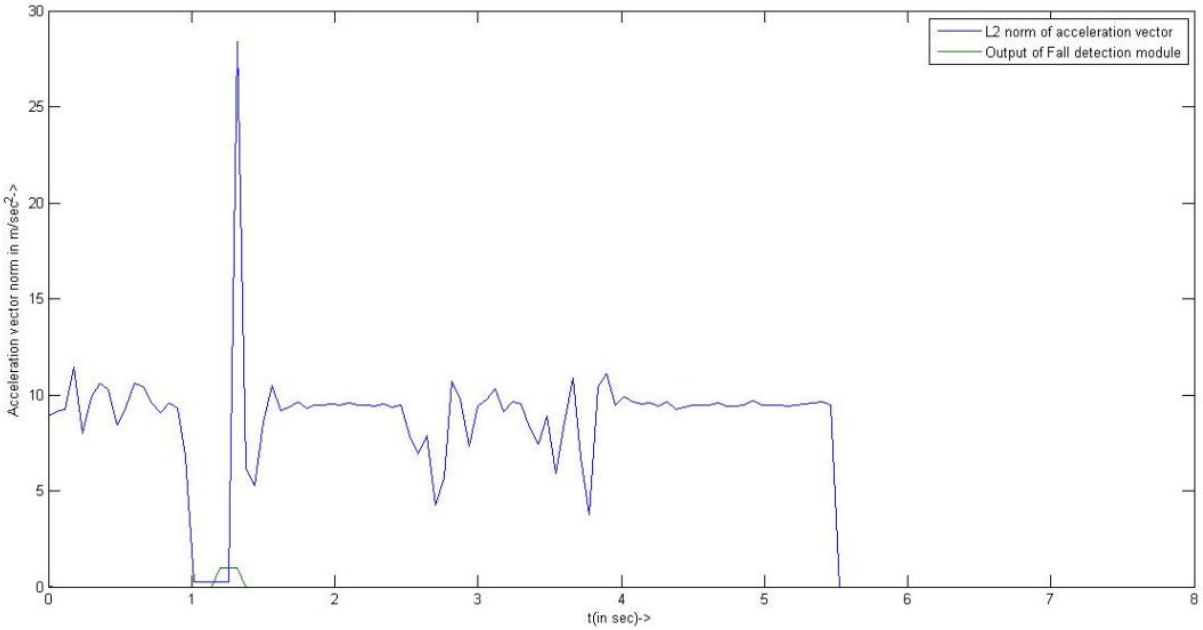


Figure 5: Fall detection module

Near 1 sec in the Figure 5, we can see output of Fall detection module is going to 1. The minima-maxima difference is greater than $2 * 9.8\ m/s^2$ at this position.

Results

For measurements, 20 people were asked to perform following activities in the same order: sitting, standing, and walking. Based on these readings, the efficiency of the algorithm is estimated. The results are tabulated in Table 3.

Probability of missed detection	1-(17/20) = 0.15	This corresponds to the percentage of states that are recognized out of order by posture recognition module
Probability of false alarm initial	10/20 = 0.5	False alarm from only Fall detection module
Probability of false alarm final	0.1	False alarm from combination of both posture recognition and fall detection module

Table 3: Results

Conclusion

With a simple threshold based approach, we were able to achieve detection of 85% accuracy in detecting different postures. Fall detection module is augmented with posture recognition to reduce false alarm rate to 10%.

References

[1] M. L. M. G. Hristijan Gjoreski, "Accelerometer Placement for Posture Recognition and Fall Detection," in *Seventh International Conference on Intelligent Environments*, 2011.

[2] G. I. o. T. Sauvik Das, U. o. H. LaToya Green, U. o. P. R. M. Beatrice Perez and F. W. O. C. o. E. Michael Murphy, "Detecting User Activities using the Accelerometer on Android Smartphones," July 30, 2010.

.