Exploring the Accessor Design Pattern for Smartwatch Based Fall Detection

Andrew Polican[1], Yeahauy Wu[2], Brock Yarbrough[3], and Dr. Anne Ngu[3]

[1]Ira A Fulton School of Engineering, Arizona State University [2]College of Science and Technology, Temple University [3]College of Science and Engineering, Texas State University

Problem

Falls in the elderly cause ~26,000 deaths and $34 billion in medical costs annually.

Existing fall detection systems use expensive custom sensors.

The Accessor Design Pattern remains untested for activity detection on mobile platforms

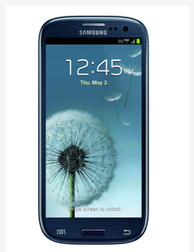
Fall detection requires low latency and high reliability for safety reasons.

Approach

We used the Microsoft Band smartwatch and an Android smartphone to detect falls.

In order to evaluate the Accessor framework we implemented two versions of our fall detection. One is a native Android application in Java, and the other is an implementation of the Accessor specification with Java and Javascript.

Both apps do fall detection locally and use Weka to run the detection model.
We also trained our own model for fall detection using Support Vector Machines (SVM).

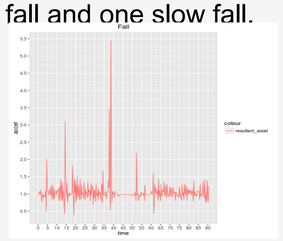


Procedure

Subjects were ages ranging from 19-29, all in good physical condition.

Accelerometer data was collected using an Android app and the Microsoft Band. Participants were told to perform at least 8 falls.

Four types of falls: front fall, back fall, right fall, left fall. For each type of fall there was one fast fall and one slow fall.



Features Used

Resultant Acceleration:

$$\sqrt{(A_x)^2+(A_y)^2+(A_z)^2}$$

CvFast:

$$CV_{Fast}(t)=\sqrt{(A_{x\_max}-A_{x\_min})^2+(A_{y\_max}-A_{y\_min})^2+(A_{z\_max}-A_{z\_min})^2}$$

Smax and Smin: maximum and minimum resultant acceleration in the sliding window.

All features except resultant acceleration were calculated using a 750 ms sliding window with 66% overlap.

Training and Testing

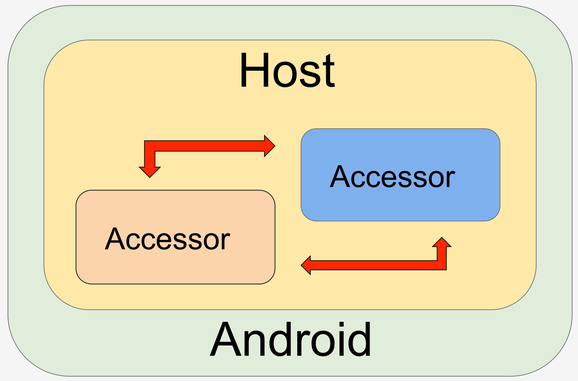
Due to sensitivity of SVM to unbalanced data, training and testing data consisted of 10% fall data and 90% non-fall data

Model was trained on one class SVM with RBF Kernel.

Training file consisted of 1938 samples while the testing set consisted of 568 samples

Accessors

Accessors are a design pattern created by the TerraSwarm Research Center.
They are self-contained Javascript objects that provide software interfaces to hardware and software services. Only their inputs and outputs are visible and they can be chained together into “Swarmlets” to create complex functionality. Swarmlets are executed by a host that is implemented to specific hardware. This enables the development of small, flexible programs that can be run on heterogeneous hardware.



Implementation

The Accessor host on Android was created using J2V8, a Java interface for the V8 Javascript engine. A simple accessor for the SVM model was written in Javascript, run with the host, and successfully predicted falls.

Weka was stripped of all GUI features so it could run on Android. An SVM model was created in Weka and added. For prediction the features calculated are written to a CSV file in groups. Each sample of the group is predicted. If 2-5 falls are predicted in a row then the final prediction is fall.

(We NSF for funding the REU (CNS-1358939) at Texas State University to perform this piece of work and the infrastructure provided by the NSF-CRI 1305302 award.)

Results

Confusion Matrix:

	Fall	Not Fall
Fall	511	3
Not Fall	35	20

Sensitivity (True Positive): 61.45%
Specificity (True Negative): 90.5%

Our Accessor host did not implement 100% of the official Accessor specifications due to time limitations. Despite this, we were able to develop a functional fall detection application that matched the functionality of the native Android application.

```
Num Instances:  110
Num Attributes: 5

   Name                 Type  Nom  Int Real     Missing      Unique  Dist
1 resultant             Num   0%   0% 100%     0 /  0%   110 100%   110
2 cvfast                Num   0%   0% 100%     0 /  0%   106 / 96%   108
3 smax                  Num   0%   0% 100%     0 /  0%    31 / 28%   61
4 smin                  Num   0%   0% 100%     0 /  0%    28 / 25%   60
5 outcome               Nom 100%   0%   0%     0 /  0%     0 /  0%    2


fall predict newinst: 1.026859,0.438893,1.026859,1.003515,fall
*****This was predicted: notfall
fall predict newinst: 1.012612,0.362756,1.026859,1.009937,notfall
*****This was predicted: notfall
fall predict newinst: 1.005841,0.322496,1.026859,1.005841,fall
*****This was predicted: notfall
fall predict newinst: 1.007313,0.228039,1.012612,1.005841,notfall
*****This was predicted: notfall
**********************: NoFall
```

Conclusions

Using commercial smartwatch sensors can yield acceptable fall detection results

The Accessor Design Pattern rivals native Android apps in both functionality and development time

Citations:
[1] S. Liu; W. Cheng. “Fall Detection with the Support Vector Machine during Scripted and Continuous Unscripted Activities”. *Sensors.* September 2012.
[2] P. Jantaraprim et. al.. “Fall Detection for the Elderly using a Support Vector Machine”. *International Journal of Soft Computing and Engineering (IJSCE)*, March 2012
[3] J. Guiry; P. van de Ven; J. Nelson. “Multi-Sensor Fusion for Enhanced Contextual Awareness of Everyday Activities with Ubiquitous Devices”. *Sensors.* March 2014,
[4] Elizabeth Latronico, et al"A Vision of Swarmlets". *IEEE Internet Computing, Special Issue on Building Internet of Things Software*, 19(2):20-29, March 2015.

This research supported by Texas State University and the NSF REU Program through REU award #135893