POTHOLE MAPPER

Updated Detail Design Document

Team Name: Droiders

Term:Spring 2014
Course: ECE 573 Software Engineering Concepts

Team Members:
Navin Chaganti
Email: nchaganti@email.arizona.edu
Graduate Student

Dhawal Srivatsava
Email: dhawalsrivastava@email.arizona.edu
Graduate Student

Executive Summary

An app like this can be used by the city transports for alerting presence of potholes and other kind of road hazards with more accuracy and reliability. The current road hazards reporting ways are through websites or phone calls, whereas this app detects the potholes automatically and plots them on the Google maps with exact coordinates. This app also gives the user the feature of manual verification of potholes such feature can be used by city transport agencies to have a more efficient road maintenance.

The application utilizes the smart phones accelerometer sensor to check the conditions of road in a moving vehicle. The app will be able to describe poor road conditions, road hazards such as potholes, speed bumps, etc. and all the different scenarios can be differentiated through different icons. The resulting data will be displayed as an overlay of colors (indicating road conditions) and icons indicating road hazards.

Main features:

1. Map of potholes,
2. Adding potholes,
3. Checking existing potholes,
4. Tracking changes for added, checked potholes or potholes nearby.

Table of Contents

Table of Figures

Project Overview

In this project we are developing a mobile application for detection of road surface hazards. Normal approach involves human inspection, however our approach uses the sensor data from the mobile devices. The data used for analysis is GPS and accelerometer data. This project uses processing of this data from sensors and plots the Potholes on the Map.

Trying to keep a check of huge network of road connections and maintenance of thousands of miles of roads is an enormous challenge for a government. Present solutions involves figuring the road qualities through websites and hotlines. But this kind of approach requires big amount of dedicated man power and financial investment. Our project aims at achieving the same target but with very little amount of manual and financial investment thus making it an interesting software product. Our project focuses on utilizing the accelerometer sensor in order to detect the road conditions and bumps automatically. This helps us to reduce dedicated man power. Also we are using the GPS sensor of smart phones to update a road hazards locations, giving very less financial and power investment for running of our application. Due to enormous amount of smart phone users, this application will lead to very wide coverage of road area.

Scope:
The project scope is to develop and design a mobile application on Android Platform that enables a user to detect Potholes on the road. The App will be running on a device lying on the dashboard of the car. The vibration signals detected by the accelerometer will be analyzed by the signal processing algorithm and finally all the detected potholes will be plotted on the Google Maps.

Problem Statement:
This project focuses on signal processing of the accelerometer data.

An Android application has to be designed which allows user to detect potholes while user is driving through a road. The application should also plot all the Potholes detected on the google Maps once the user stops detecting. Different Road Hazards need to be plotted as different events on the Map.

Requirements

Critical Requirements (B-Class Requirements)

B.1. Transfer function for all 5 algorithm be implemented on Matlab.

B.1.1. Algorithm-1 transfer function: $H_h(Z) = \left(\frac{1+a}{2}\right)\left(\frac{1-z^{-1}}{1-az^{-1}}\right)$

B.1.1.1. Matlab implements this transfer function successfully based on the parameters calculated. This mean if X-axis acceleration values is filtered through this transfer function, then output should not contain any window which contains either Low-Speed or Curb Hit flag.

B.1.2. Algorithm-2 transfer function: $H_v(Z) = \left(\frac{1+b}{2}\right)\left(\frac{1-z^{-1}}{1-bz^{-1}}\right)$

B.1.2.1. Matlab implements this transfer function successfully based on the parameters calculated. This mean if Z-axis acceleration values is filtered through this transfer function, then output should not contain any window which contains flags like either Smooth Road or Turn.

B.1.3. Algorithm-3 transfer function:

$$H_{z-axiz}(Z) = a_0 + a_1Z^{-1} + a_2Z^{-2} + a_3Z^{-3} \dots\dots (-a_{n-3})Z^{-(n-3)} + (-a_{n-2})Z^{-(n-2)} + (-a_{n-1})Z^{-(n-1)}$$

B.1.3.1. Matlab implements this transfer function successfully based on the parameters calculated. This mean if Z-axis acceleration values is filtered through this transfer function, then output should not contain any window which contains flags like Break and Phone Drop.

B.1.4. Algorithm-4 transfer function:

$$H_{x-axiz}(Z) = b_{p_z0} + b_{p_z1}Z^{-1} + b_{p_z2}Z^{-2} + b_{p_z3}Z^{-3} \dots\dots (-b_{p_zn-3})Z^{-(n-3)} + (-b_{p_zn-2})Z^{-(n-2)} + (-b_{p_zn-1})Z^{-(n-1)}$$

B.1.4.1. Matlab implements this transfer function successfully based on the parameters calculated. This mean if X-axis acceleration values is filtered through this transfer function, then output should not contain any window which contains flags like Speed bump and Joint Expansion/Railway Crossing.

B.1.5. Algorithm-5 implementation: Discarding peak with less than T_s*Speed. (T_s = 5)

B.1.5.1. Matlab implements this function successfully based on the parameters calculated. This mean if Z-axis acceleration values is processed through this function,

then output should not contain any window which contains flags like High speed trivial hit.

B.2. Transfer function for all 5 algorithm be implemented on smart phone. The results obtained from Algorithm-I, Algorithm-II, Algorithm-III, Algorithm-IV, Algorithm-V on smart phone should match respective results obtained through Matlab implementation in B.4.

B.3. Record GPS co-ordinates of detected events.

B.4. On detection of a road hazard, App gives user options to choose the type of "Anomaly" encountered.

B.5. When vehicle is stationary and experience a jerk, App does not record this event as pothole encounter.

Bells & Whistles (A-Class Requirements)

A.1 App automatically differentiates pothole encounters from other false targets with in the error bounds that will be established.

A.1.1. When vehicle stops short due to sudden breaks, App does not detect this scenario as pothole.

A.1.2. If user drops the phone while driving, the app does not consider this activity as a pothole encounter.

A.2. When vehicle is driven on long stretches of low quality road, the app should consider the whole stretch as a road hazard.

A.3. App differentiates pothole encountered based on their size as Small Medium Large.

A.4 During a call, the App goes on pause state and stops recording data.

A.5. GPS data recorded are plotted on the map.

A.6 Potholes detected are categorized based on current vehicle speed.

A.7. App supports a demo mode for user interaction

A.7.1. In demo mode, events are injected using pre-recorded data and application detects the event accurately as either speed bump, pothole or driving over a curb.

Application Analysis

State Model

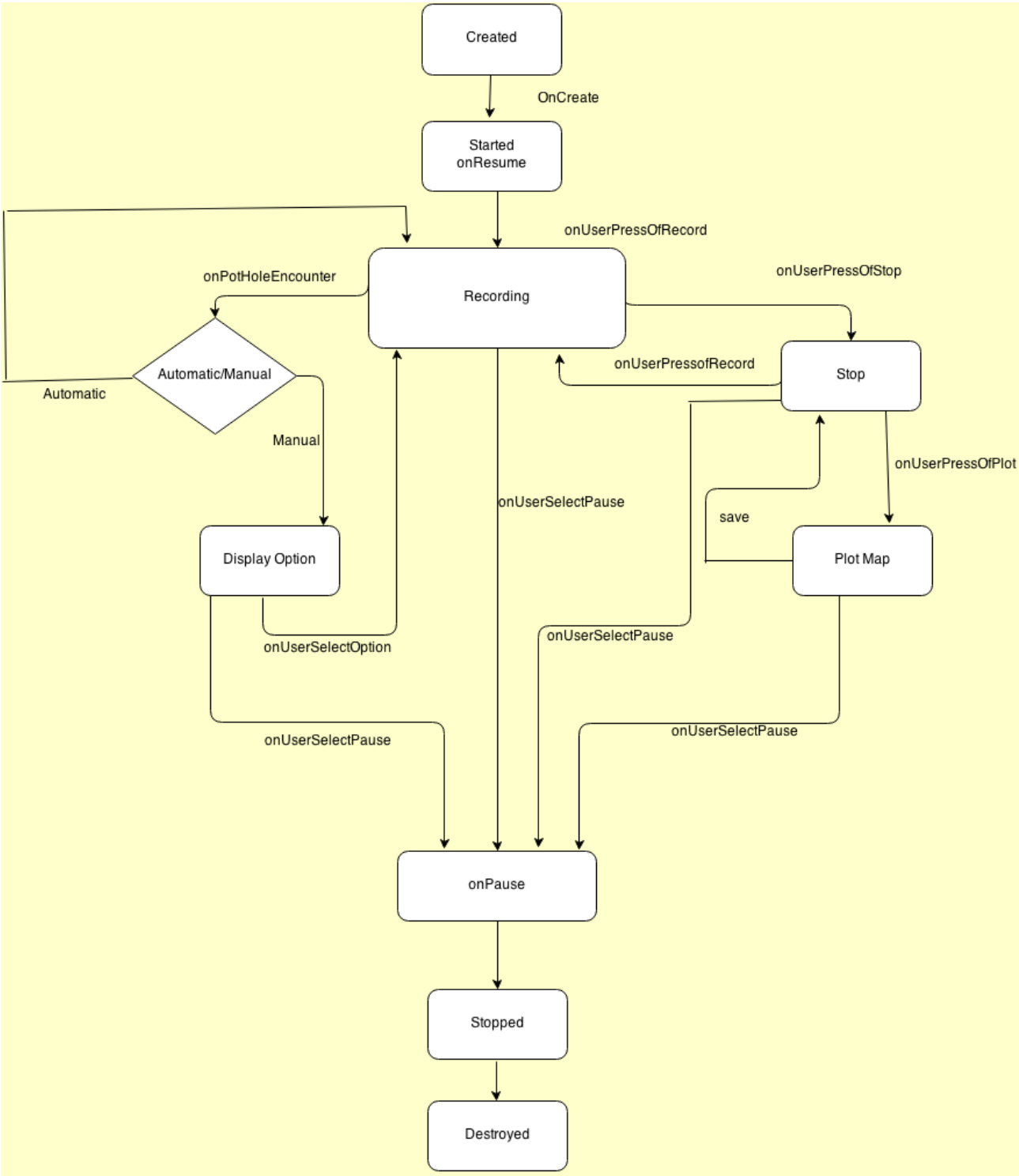

Figure 1: State Model Diagram

The intial Screen when the aplication is started. It describes the scenario when the data yet to be recorded.

App consists of two modes Automatic and Manual

This Automatic mode doesnot promt the user and plots the data as per the signal processing algorithms provided in the app.The user is not given the privilige to select the kind of road hazard in this scenario

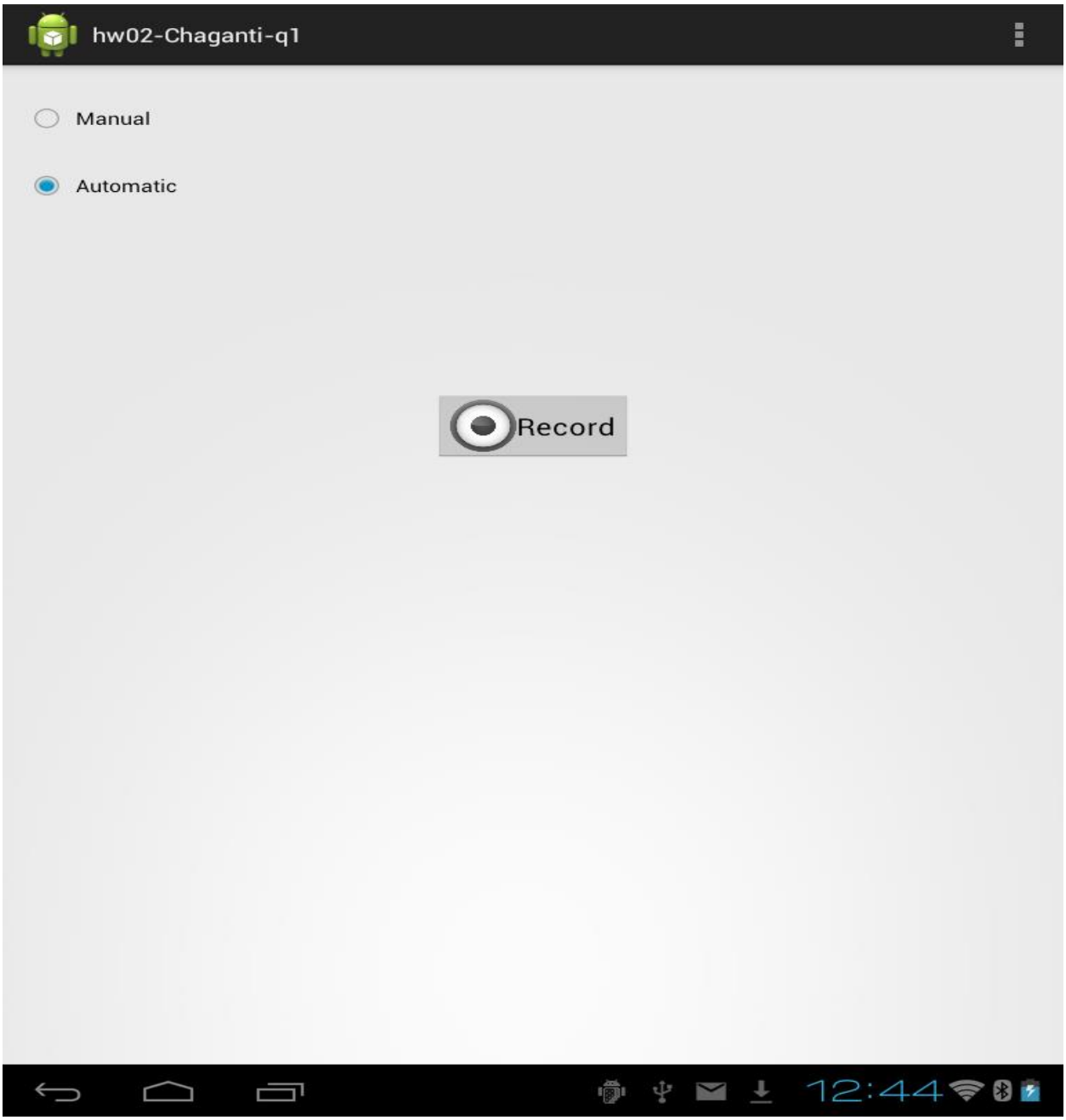

Figure 2: Start Screen

When the App is in the manual mode the detected event causes the app to open another activity which enable the user to specify the kind of road hazards.

This allow the user to record his experience while recording the data manually and this feature helps the person to manually verify the data with the automatic results provided in the automatic mode.

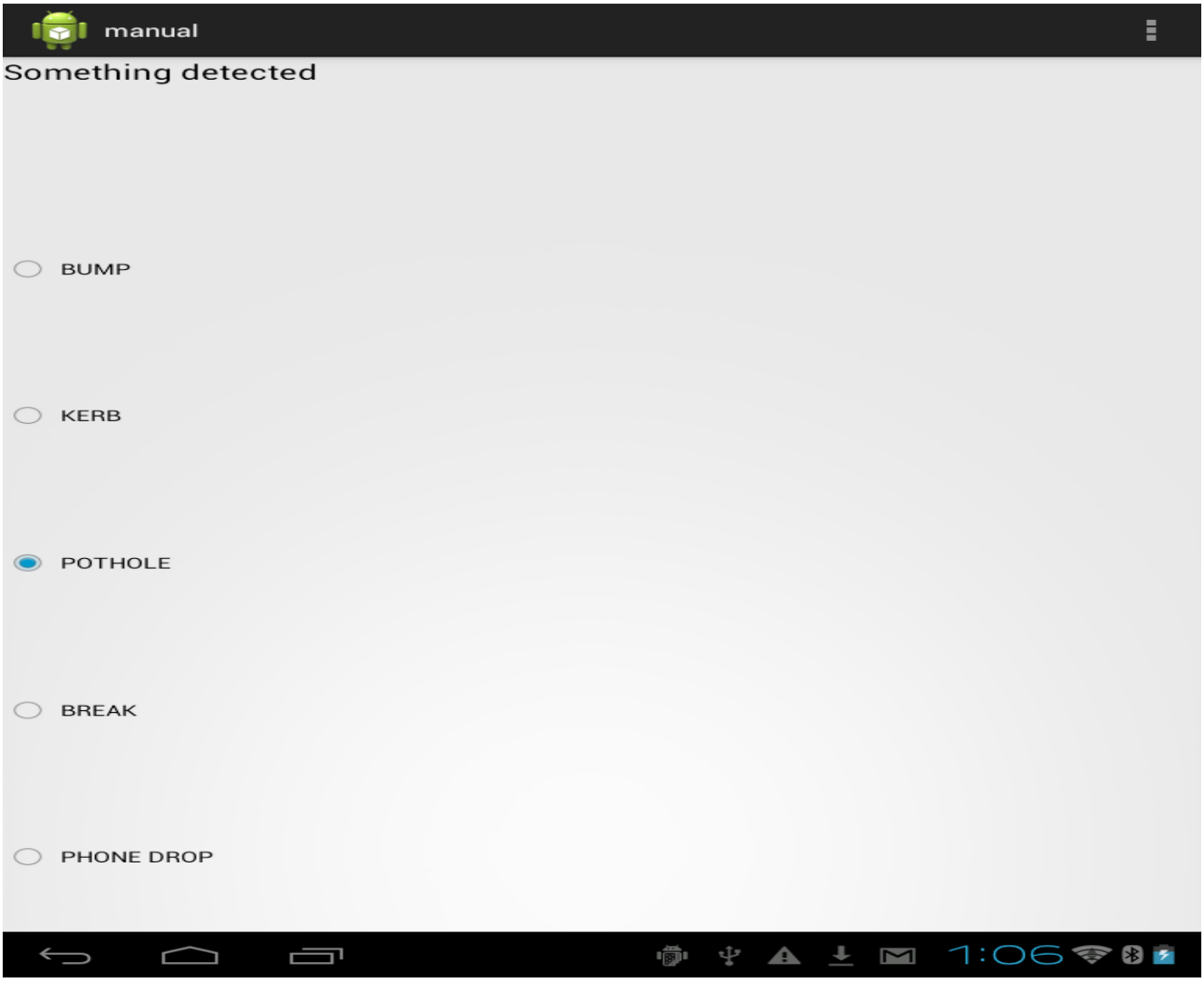

Figure 3: Manual Mode option activity

When the record button is selected the app starts recording the data and the 3 textview box indicate the X, Y and Z coordinates values sensed by the accelerometer.

The seconds indicate the amount of seconds the data is recorded

Stop button causes the record to be stopped.

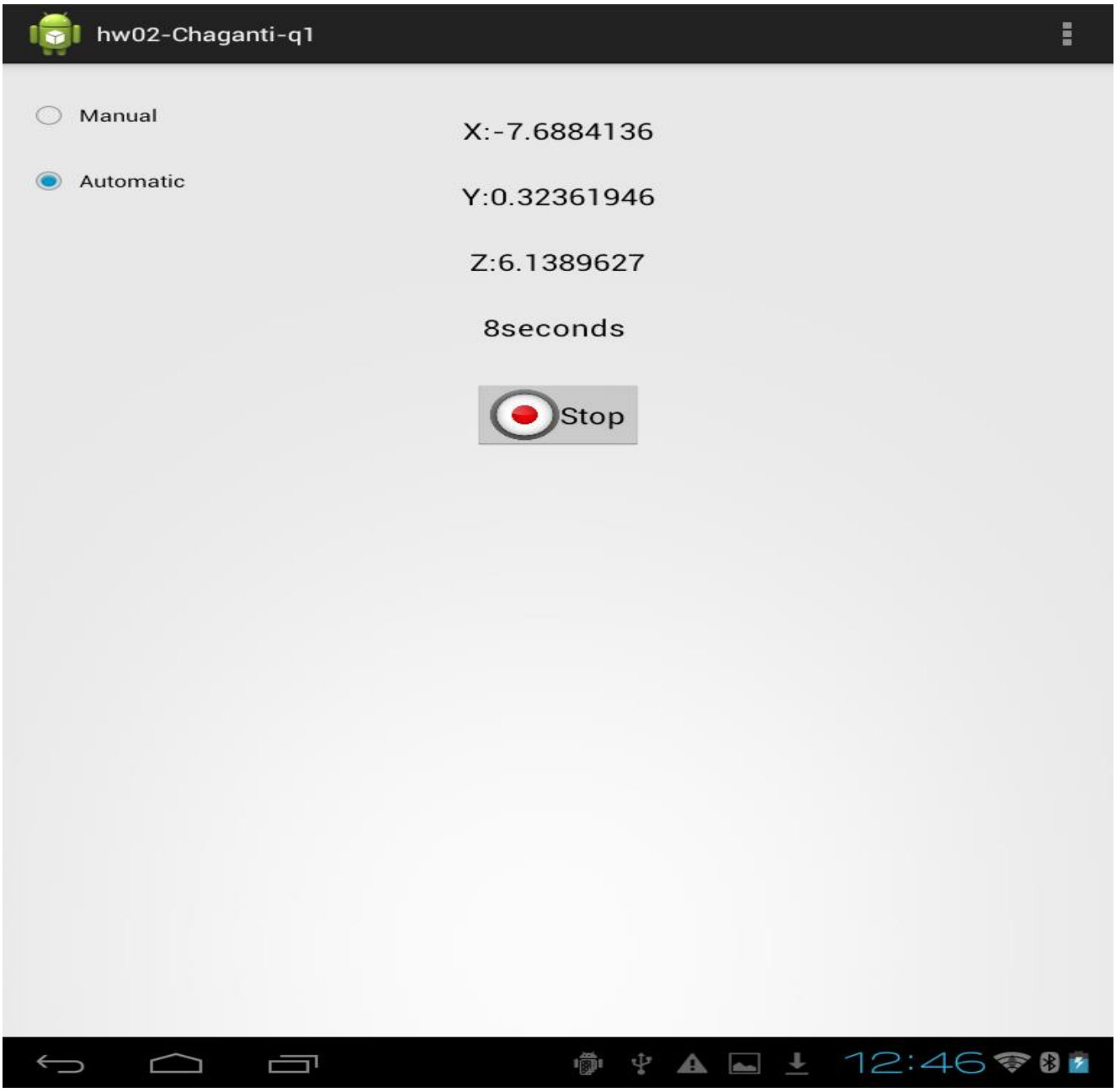

Figure 4: Recording Phase

After the recording of data the app changes the stop button back to record state

The app enables a button plot data which allows the user to check the recorded data on a map.

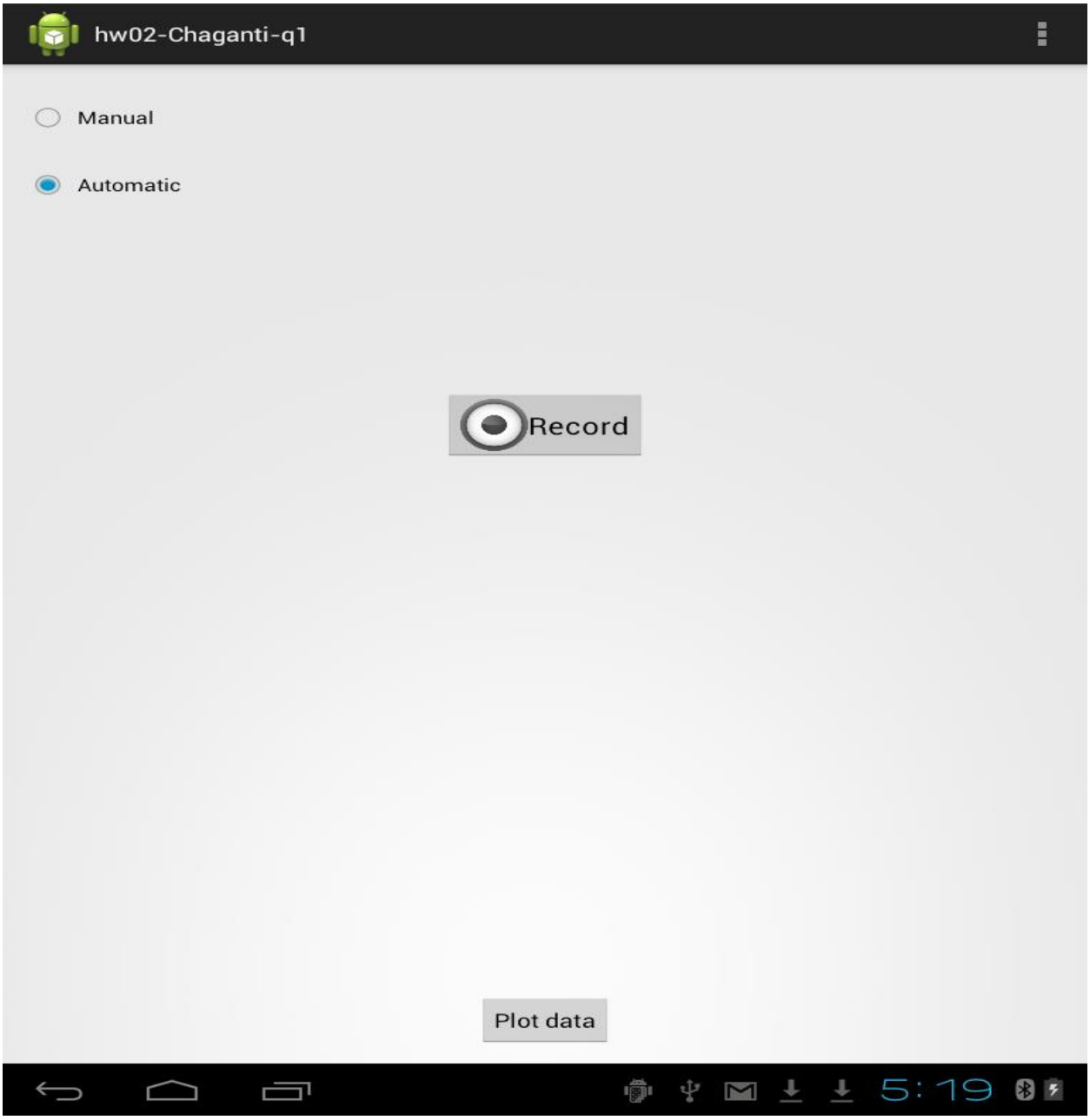

Figure 5: After Recording /Plotting mode

Domain Analysis

When user press the record button on the screen. The user interface class senses the tap on the record button and create an object of class accelerometer fetch. This accelerometer fetch class interacts with the smartphone's accelerometer interface and fetches the values of the three events from the accelerometer sensor. These three events corresponds to horizontal, lateral and vertical acceleration of the device under test. Once the fetching has starts then the Accelerometer Fetch class invokes object of GPS Fetch class to get the latitude and longitude co-ordinates of the location of the device under test. The GPS Fetch class also calculates the speed of the vehicle through the getSpeed() function. Till the application is in recording state, 5 data arrays containing Horizontal Acceleration, Vertical Acceleration, Latitude, Longitude and Speed samples) are stored simultaneously. After getting 100 or more samples, the SignalProcessor class object is invoked and in this object a sliding window of length 100 samples is applied on the Horizontal and Vertical Acceleration array to process the road anomaly, the computation of outputs from arrays are controlled by speed of the array. The computation of the pothole event is done in 5 sequential steps each involving an algorithm which is discussed in depth in next section. The Vertical Acceleration array is processed through several transfer functions for detection of a pothole (or similar) event and Horizontal Acceleration array is processed to validate that the detection is in fact a pothole. The input of each algorithm is the output from the previous algorithm and in turn is fed as an input to next algorithm. These five algorithm are obtained from a research paper [1] presented on the problem of pothole detection. If all 5 algorithm passes an event in the sliding window as a positive match for pothole, then the index value of the peak in that Vertical Acceleration array window is stored in a separate index array. When user press stop recording button then this index array is read by the MapPlotter object and corresponding latitude and longitude are sent to the plotMap() function of the object. The MapPlotter object plots the received latitude and longitude on the Google map API.

Match Filter: One of the prime implementation in our algorithm is that of Match filter. It is obtained by correlating a known signal (Pothole pattern), with an unknown signal (Data obtained from accelerometer sensor) to detect the presence of the template in the unknown signal. This is equivalent to convolving the unknown signal with a conjugated time-reversed version of the template. The matched filter is the optimal linear filter for maximizing the signal to noise ratio (SNR) in the presence of additive stochastic noise. Out threshold is decided based on the desired SNR.

Algorithms

Algorithms of Signal processing function:

The pothole event on the road is detected using these 5 sequential steps.

1. Algorithm-I: This algorithm checks the condition that the speed of the vehicle is above a bare minimum threshold. This algorithm makes sure that false detection like Closing of the door and parking curb hits are not marked as pothole events. This algorithm is targeted on the Horizontal Acceleration array.
   10 samples of array data is selected and are passed through a simple first order high-pass filter. The transfer function of the first order high pass filter is

$$H_h(Z) = \left(\frac{1+a}{2}\right)\left(\frac{1-z^{-1}}{1-az^{-1}}\right)$$

   The parameter value of 'a' is defined based on taking average of the readings obtained from different vehicle. This parameter value is directly related to the minimum speed value below which any pothole event is discarded.

2. Algorithm-II: This algorithm checks the condition that the data sets obtained from the accelerometer doesn't contain noise in it. We are targeting on high frequencies signal for detection of potholes. For our scenario, low frequencies signals will pose problems in detection hence we will filter our Vertical Acceleration data through a simple first order high-pass filter. The transfer function of the first order high pass filter is

$$H_v(Z) = \left(\frac{1+b}{2}\right)\left(\frac{1-z^{-1}}{1-bz^{-1}}\right)$$

   The parameter value of 'b' is defined based on taking average of the readings obtained from different vehicle. This parameter value is directly related to the maximum frequency obtained from the accelerometer if the vehicle is running on a smooth road.

   It should be noted that although both the algorithms (Algorithm-I and Algorithm-II) are employing a simple order high pass filter, they are targeting different data sets and also have different cutoff frequency.

3. Algorithm-III: In this algorithm the pattern of a pothole is searched in a window of 500 samples of data obtained from Vertical Acceleration array. An estimate of pothole signal is created and is used as the impulse response for the design of an FIR Match filter. Pothole pattern in the z-axis is a significant pattern for detection of road anomalies. This filter rejects all windows which doesn't have pothole pattern in it.

The filter input is obtained from the output of algorithm-II. The high passed signal is searched for the pothole pattern and if the convoluted output's expected value is above the threshold T_z then the sample is assumed to contain a pothole. The threshold T_z is calculated based on the data obtained from different vehicles. The transfer function of the Z axis FIR Match filter is:

$$H_{z-axiz}(Z) = a_0 + a_1Z^{-1} + a_2Z^{-2} + a_3Z^{-3} \dots\dots (-a_{n-3})Z^{-(n-3)} + (-a_{n-2})Z^{-(n-2)} + (-a_{n-1})Z^{-(n-1)}$$

The transfer function contains n samples. Here the parameters of the function a_0, $a_1 \dots a_{n-1}$ are obtained from a sample of pothole which was first detected manually and then analyzed for the pattern. An initial estimate for the number of samples could be 100. The output of the filter will give an estimate of the peak of the Z-axis (P_z) in the small n-size window frame.

4. Algorithm-IV: Acceleration along the X-axis also plays an important role in verifying the detection of the anomaly as a pothole. Road anomalies like speed bump, railway track and expansion joints impact both the side of car equally. However a typical pothole is assumed to impact only one side of the car. Hence the acceleration along the X-axis will be more in case of actual pothole hit as compared to the other anomalies. Also the acceleration will be dependent on the pothole type and thus depends on the peak P_z of the acceleration in Z-axis. Based on the P_z obtained from Algorithm-III a second match filter can be modeled with the sample length of 35 samples (in order to look in close proximity of the pothole detection). The sample chosen will be around the index of the detection of the peak in z-axis array. The transfer function of the X axis FIR Match filter is:

$$H_{x-axiz}(Z) = b_{p_z0} + b_{p_z1}Z^{-1} + b_{p_z2}Z^{-2} + b_{p_z3}Z^{-3} \dots\dots (-b_{p_zn-3})Z^{-(n-3)} + (-b_{p_zn-2})Z^{-(n-2)} + (-b_{p_zn-1})Z^{-(n-1)}$$

This filter rejects all windows which doesn't have X-axis acceleration pattern in it. The parameters of this transfer function depends on the P_z obtained from previous algorithm. And the n is the number of samples which is 35. A factor of T_x can be used to model 'b' parameters from P_z.

2. It is observed that at high speeds, high peak is observed in acceleration reading for even a small road anomaly. This algorithm ensures that the pattern for the pothole encounter which is searched for in the Z-axis reading is developed dynamically based on the speed of the vehicle. This filter rejects windows where although a pothole pattern is present the peak z acceleration is less than a predefined factor Ts times the speed of the vehicle.

   So the output obtained till algorithm-IV is looked for the peak in the window. If the peak is less than the T_s times the speed, then the window is discarded.

   From the previous implementation [1], T_s is initiated at 5 and T_x is initiated at 2.5.

Class Diagram

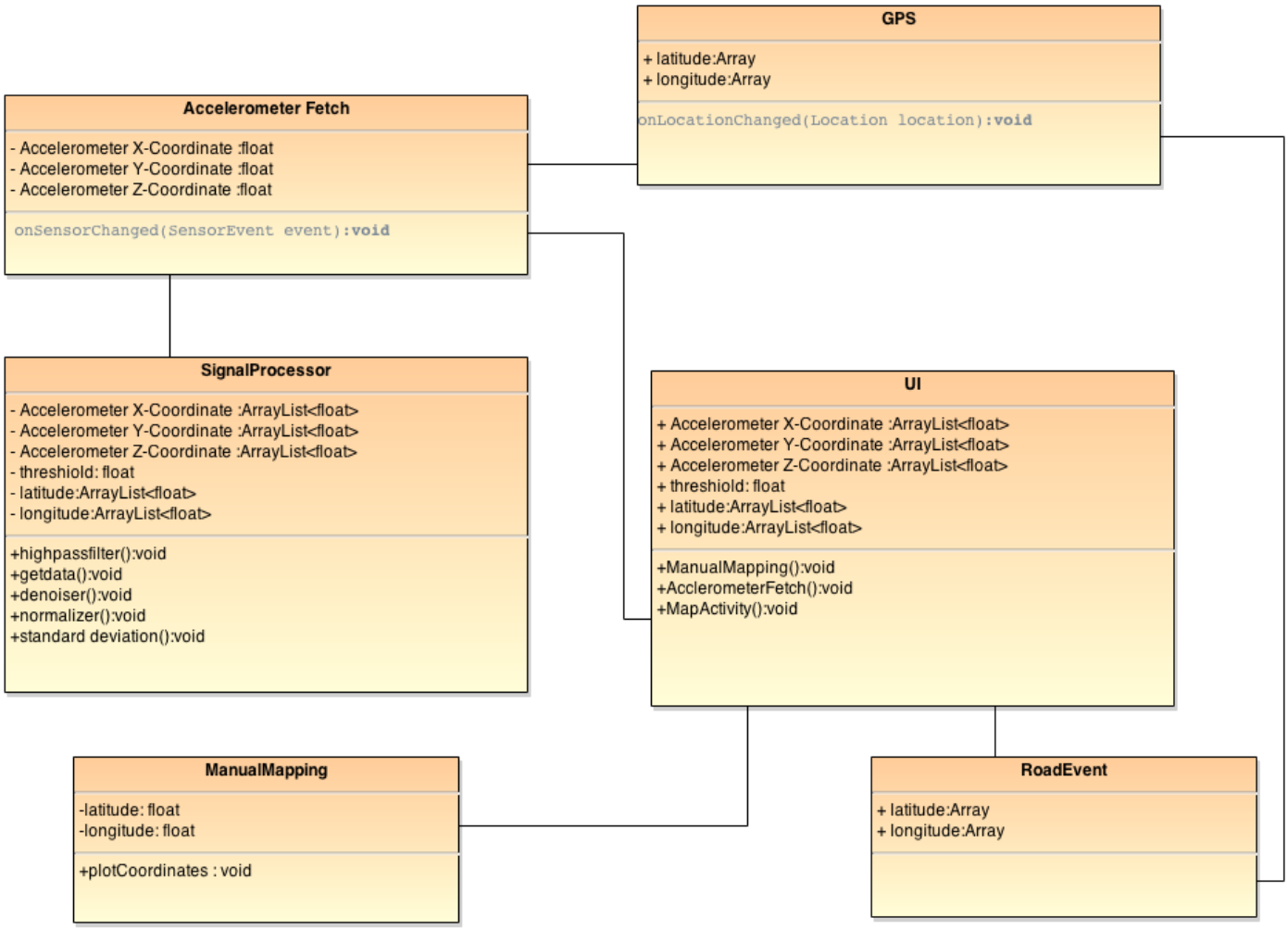

Figure 6. Class Diagram

Testing Strategy

Pre-Conditions

1. Switch on the device

2. The accelerometer sensor should be available on the device and should be properly functioning.

3. Enable the GPS receiver.

4. Install the app successfully

5. Switch on the app

6. Make sure the app is lying on its back with the screen facing the ceiling and is on a horizontal surface like car dashboard.

7. Press the record button.

Matlab Pre-Conditions

Implement Algorithm-I, Algorithm-II, Algorithm-III, Algorithm-IV and Algorithm-V on Matlab.

Test Cases

Test Case Id: Test.1.

Requirement id: B.1.1

Steps

1. Implement Algorithm-I on Matlab.
2. Filter X-Axis acceleration data through Algorithm-I transfer function.

Expected Results

1. X-axis acceleration values is filtered through this transfer function, then output should not contain any window which contains either Low-Speed or Curb Hit flag.

Test Case Id: Test.2.

Requirement id: B.1.2

Steps

1. Implement Algorithm-II on Matlab.
2. Filter Z-Axis acceleration data through Algorithm-II transfer function.

Expected Results

1. Z-axis acceleration values is filtered through this transfer function, then output should not contain any window which contains flags like either Smooth Road or Turn.

Test Case Id: Test.3.

Requirement id: B.1.3

Steps

1. Implement Algorithm-III on Matlab.
2. Filter Z-Axis acceleration data through Algorithm-III transfer function.

Expected Results

1. Z-axis acceleration values is filtered through this transfer function, then output should not contain any window which contains flags like Break and Phone Drop.

Test Case Id: Test.4.

Requirement id: B.1.4

Steps

3. Implement Algorithm-IV on Matlab.
4. Filter Z-Axis acceleration data through Algorithm-IV transfer function.

Expected Results

1. X-axis acceleration values is filtered through this transfer function, then output should not contain any window which contains flags like Speed bump and Joint Expansion/Railway Crossing.

Test Case Id: Test.5.

Requirement id: B.1.5

Steps

5. Implement Algorithm-V on Matlab.
6. Process Z-Axis acceleration data through Algorithm-V function.

Expected Results

1. Z-axis acceleration values is processed through this function, then output should not contain any window which contains flags like High speed trivial hit.

Test Case Id: Test.6.

Requirement id: B.2

Steps

7. Implement Algorithm-I, Algorithm-II, Algorithm-III, Algorithm-IV and Algorithm-V on Smartphone.
8. Filter X-Axis, Z-Axis acceleration data through function implemented on device.

Expected Results

1. Transfer function for all 5 algorithm be implemented on smart phone. The results obtained from Algorithm-I, Algorithm-II, Algorithm-III, Algorithm-IV, Algorithm-V on smart phone should match respective results obtained through Matlab implementation in B.4.

Test Case Id: Test.7.

Requirement id: B.1.1

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select 'Manual' radio button.
3. Keep the vehicle stationary.
4. Give vertical jerks to the device manually.

Expected Results

1. App should not detect the event as a pothole and should not give user list of possible options.

Test Case Id: Test.8.

Requirement id: B.1

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select 'Automatic' radio button.
3. Keep the vehicle stationary.
4. Give vertical jerks to the device manually.

Expected Results

1. App should not detect the event as a pothole.

Test Case Id: Test.9.

Requirement id: B.1

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select 'Automatic' radio button.
3. Drive the vehicle over a pot hole.

Expected Results

1. The accelerometer value should be processed through filter algorithms and output values should match the reference values from Matlab.

Test Case Id: Test.10.

Requirement id: B.1

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select 'Manual' radio button.
3. Drive the vehicle over a pot hole.

Expected Results

1. The accelerometer value should be processed through filter algorithms and output values should match the reference values from Matlab.

Test Case Id: Test.11.

Requirement id: B.1

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select 'Manual' radio button.
3. Drive the vehicle over a pot hole on a speed range of 20-30 mph.
4. Drive the vehicle over a pot hole on a speed range of 30-40 mph.
5. Drive the vehicle over a pot hole on a speed range of 40-50 mph.

Expected Results

1. The Pothole should be detected and should be categorized based on the speed for steps 3, 4, 5.

Test Case Id: Test.12.

Requirement id: B.1

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select ‘Automatic’ radio button.
3. Drive the vehicle over a pot hole

Expected Results

1. The Pothole should be detected and should be categorized based on the speed for steps 3, 4, 5 for Test.14.

Test Case Id: Test.13.

Requirement id: B.3

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select ‘Automatic’/'Manual' radio button.
3. Drive the vehicle over a pot hole

Expected Results

1. The location shown on the toast for the Pothole coordinates should be fetched from the GPS of the phone and the values should match to a reference application on the device.

Test Case Id: Test.14.

Requirement id: B.1

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select ‘Automatic’/'Manual' radio button.
3. Drive the vehicle over a pot hole

Expected Results

1. The data obtained after being processed from the SignalProcessor function should match the results obtained from the reference implementation on Matlab.

Test Case Id: Test.15.

Requirement id: A.3

Steps

1. Switch on the Demo Mode.
2. Locate the file where the recorded accelerometer data are present.
3. Set the velocity using the velocity control button
4. Press 'Start Demo’

Expected Results

1. The app should detect the potholes, speed bump, Curb etc. from the input accelerometer file.

Test Case Id: Test.16.

Requirement id: B.1

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select 'Automatic' radio button.
3. Drive the vehicle slowly in the parking lot over a curb.

Expected Results

1. App should not detect the event as a pothole hit.

Test Case Id: Test.17.

Requirement id: A.1.1

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select 'Automatic' radio button.
3. Drive the vehicle for some distance and apply sudden break.

Expected Results

1. App should not detect the event as a pothole hit.

Test Case Id: Test.18.

Requirement id: A.1.2

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select 'Automatic' radio button.
3. Drive the vehicle over a speed bump.

Expected Results

1. App should not detect the event as a pothole hit.

Test Case Id: Test.19.

Requirement id: A.1.2

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select 'Automatic' radio button.
3. Drive the vehicle over a distance.
4. While driving, drop the phone from dash board on car seat

Expected Results

1. App should not detect the event as a pothole hit.

Test Case Id: Test.20.

Requirement id: A.2

Steps

1. Make sure that the pre-conditions are satisfied.
2. Select 'Automatic' radio button.
3. Drive the vehicle over a stretch of low quality.

Expected Results

1. App should detect the starting co-ordinates and ending co-ordinates of the stretch and record the data as a road hazard.

Integration with Platform

The accelerometer uses the function public void onSensorChanged(SensorEvent event) which continuously track for the change in the sensor values and having frequency of game mode which is usually 20-50Hz values array contents depend type of sensor being monitored, as the linear acceleration is the major parameter which has to be sensed.

Sensor.TYPE_ACCELEROMETER:

SI units (m/s^2) are the values which are obtained

- values[0]: x-axis Acceleration minus Gx
- values[1]: y-axis Acceleration minus Gy
- values[2]: z-axis Acceleration minus Gz

Acceleration applied to the device (Device) is measured by the Sensor. Forces applied on the sensor are calculated by relation:

Device = - ∑Force / mass
Gravity influence the measured acceleration:

Device = -gravity - ∑Force / mass
Gravity = 9.81 m/s^2
When the device is sitting on a table

Similarly, when the device is in free-fall and therefore dangerously accelerating towards to ground at 9.81 m/s^2, its accelerometer reads a magnitude of 0 m/s^2.

The values generated by the accelerometer are compared with the reference values obtained from the Signal processing in Matlab. When the comparison is successful the GPS event is triggered

GPS:
The android phone uses the function public abstract void getLatitude() and getLongitude() to fetch the GPS data.

Orientation:
The phone should be lying horizontal on the dashboard of the Car and should be fixed firmly to the dashboard with glue or tape. The horizontal axis of the device should be parallel to the direction of the motion of the vehicle.

Calibration:
The app will be in the calibration mode for 30 seconds and will start when x and y axis coordinates show value 0.0 with a deviation of 0.05.

Task Allocation and Breakdown

Task allocation and breakdown

1. Class to be implemented: UI
   Project member: Navin Chaganti
   Requirements covered: B.1, B.2, B.3

2. Class to be implemented: ManualMapping
   Project member: Navin Chaganti
   Requirements covered: B.3

3. Class to be implemented:SignalProcessor
   Project member: Dhawal Srivastava
   Requirements covered: B.4, B.5, B.5,A.1, A.2

4. Class to be implemented: RoadEvent
   Project member: Navin Chaganti
   Requirements covered: B.4 , A.2,A.3

5. Class to be implemented:Acclerometer Fetch
   Project member: Dhawal Srivastava
   Requirements covered: B.1

6. Class to be implemented: GPS
   Project member: Navin Chaganti
   Requirements covered: B.3 ,A5

7. Implementation of Algorithms on Matlab
   Project member: Dhawal Srivastava
   Requirements covered: B.1

8. Implementation of Demo
   Project member: Navin Chaganti
   Requirements covered: A.7

Timeline For Completion

Tasks to be implemented by 03/01/14

- Raw Data Collection
- Implementation of Class User Interface.
- Implementation of Class Accelerometer Fetch

Task allocation number: 1, 5.
Requirements Covered: B.1, B.2, B.3,

Tasks to be implemented by 03/07/14

- Implementation of Class ManualMapping
- Implementation of Class GPS

Task allocation number: 2, 6
Requirements Covered : B.1, B.4

Tasks to be implemented by 3/10/2014

- Bugfixes for Alpha release
- Alpha release

Tasks to be implemented by 03/20/14

- Implementation of Class SignalProcessor

Task allocation number: 3
Requirements Covered : B.1, B.2, B.5,A.1, A.2

Task to be implemented by 03/27/14

- App testing for signalProcessor Class
- App testing for MapActivity Class
- Implementation of class RoadEvent

Task allocation number: 4
Requirements Covered : B.4 , A.2,A.3

Task to be implemented by 04/04/14

- Implementation of Demo Mode

Task allocation number: 8
Requirements Covered : A.6

Task to be implemented by 11/04/14

- App testing of Demo mode
- Implementation of Signal Processing Algorithm
- Bug fixes for Beta release

Task allocation number: 7
Requirements Covered : B.1

Task to be implemented by 18/04/14

- Beta Release

Global/Shared Task and Experience

Documentation Task Allocation:

Navin Chaganti:

- Worked on the Map based requirements
- GUI based Interface and Class description
- Testing Strategies for the UI
- Manual Mode of App
- Map and GPS activity of the App
- Detail Design Documentation of Map and GPS

Dhawal Srivastava:

- Studied and came up with the Signal Processing requirements
- Back end software Activity and its Class Description
- Analysis of Accelerometer data and implementation of Class
- Automatic mode of App
- Testing strategy for signal processing algorithms
- Detail Design Documentation of Accelerometer and Signal Processing Algorithms

Software Tasks Allocation.

Navin Chaganti:

Major Task allocated

- Implementing UI class
- Implementing GPS class
- Implementing ManualMapping Class
- Implementing RoadEvent Class

Justification: Previous Experience in Map Based apps.

Major focus on the UI of the App and Plotting the Coordinates detected on the MAP
Also responsible for the manual Mode of the app where the User chooses the Particular Road event to Plot on Map
Dhawal Srivastava:

Major Task allocated

- Implementation of Signal Processor Class
- Implementation of Accelerometer Fetch Class
- Implementation of algorithm on Matlab

Justification: Currently took class on DSP also pursuing thesis on signal processing and Image Processing.

Major Focus on Implementing the Back end Signal Processor software in order to automatically detect the particular Pothole and automatically storing them as a Road event in order to plot on the Map.

Reference

[1]. J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: using a mobile sensor network for road surface monitoring," in Proceeding of the 6th international conference on Mobile systems, applications, and services, ser. MobiSys '08. New York, NY, USA: ACM, 2008, pp. 29–39. [Online]. Available: http://doi.acm.org/10.1145/1378600.1378605