Road Conditions

Therese Horey
therese.horey.57@my.csun.edu

ABSTRACT

Information on road conditions is useful to know as it has a wide range of applications, from vehicular and pedestrian safety issues to infrastructure maintenance. The Android app that is developed seeks to gather road condition data while the user is jogging. The approach taken in this paper relies upon the use of the phone's accelerometer and filtering algorithms. The detected road anomalies and relevant data are stored in CSV files in the phone's internal storage. Testing was completed to confirm the accuracy of the results and moderate success has been achieved.

1. INTRODUCTION

Good roads are essential for a modern society. Knowledge of road conditions is important since vehicles and people use roads every day and poor quality roads can cause many safety issues. Gaining data on road conditions is an ongoing task with different approaches. Most work on this subject deals with gathering data from cars; however, people also frequently use roads. Many joggers take the road and bring their Smartphone along with them. This project therefore aims to create an app that can determine road conditions while the user is jogging.

This app uses the phone's accelerometer sensor to measure the acceleration in the y-direction, which is vertical to the user. A high-pass filter is applied to the raw data to remove the force of gravity so that the user's movements can be more closely monitored. This filtered data is then compared to a set threshold to find road anomalies, which includes bumps, potholes, or steep changes in road elevation. The threshold level is determined after much testing to find the right balance of bump detection and minimal false positives. The results are recorded into two CSV files for each jog. When the user stops the data recording with the app, useful information about the jog is displayed, such as: time passed, distance traveled, average road quality, and average speed.

2. RELATED WORK

It is difficult to find published work on projects relating to the analysis of road conditions while the user is jogging. The works found that were most similar to this project all dealt with determining road conditions from a car. The first such work comes from the University of Calabria in Italy and aims to monitor road surface quality from Smartphones fixed in certain positions on a car. [1] These phones used the accelerometer and GPS to test for the existence and locations of potholes and bumps. The accelerometers were oriented through the use of Euler angles and filters were applied to the data to clean the signal for analysis. A Low Frequency Filter was used first, followed by a Speed Filter, and last a Small Peaks Filter. After testing, this project was found to accurately detect bumps in the road, however only had a 65% pothole detection rate.

Another project that dealt with evaluating road conditions with an accelerometer was carried out by the University of Latvia. [2] In this case, the raw accelerometer data was passed through four algorithms that would detect potholes based on event values exceeding a specified threshold level. One of these algorithms made use of the mean and standard deviation of the data for bump detection. This project was tested with various results, ranging from a detection rate of 47% to 100%, depending on the type of bump.

These related works are relevant to a jogging road condition app since the data dealt with is highly variable, speed has to be taken into account, and the app must work with different unique users. False positives and how they can be reduced is also an issue.

3. SOFTWARE INTERFACE

The system includes the user and the phone. As a jogging app, the user interaction with the app will be minimal. The user should start the app, place the phone in an armband to secure it for data collection while jogging, then remove the phone and stop the data collection after exercising. At this point some data regarding the recent jog will be displayed for the user and the user may decide to view the jog results with a Google map. The app will also save a results file and a sensor data file that can be opened and examined using Excel onto the phone's internal storage. These files can be viewed anytime the phone is plugged into a computer. Specific examples and screenshots are given below.

Figure 1. Start screen.

Figure 2. Countdown.

Figure 3. Stats are displayed when data collection is stopped.

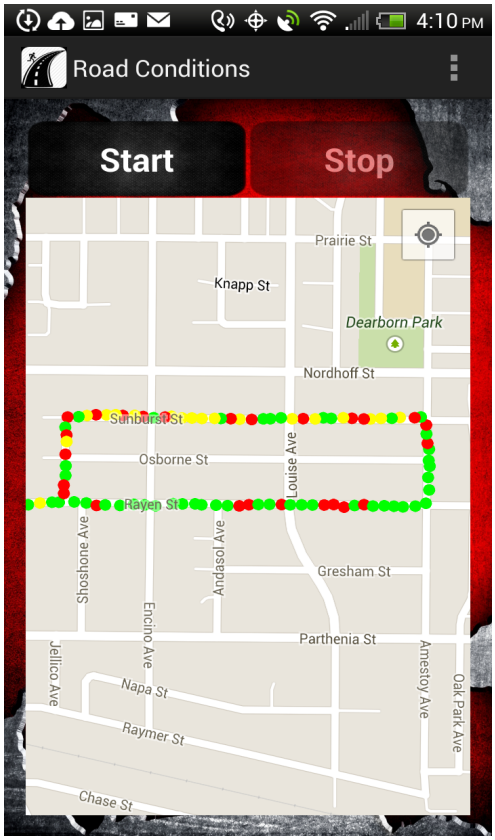

Figure 4. Map is displayed after user presses View Results.

4. SYSTEM ARCHITECTURE

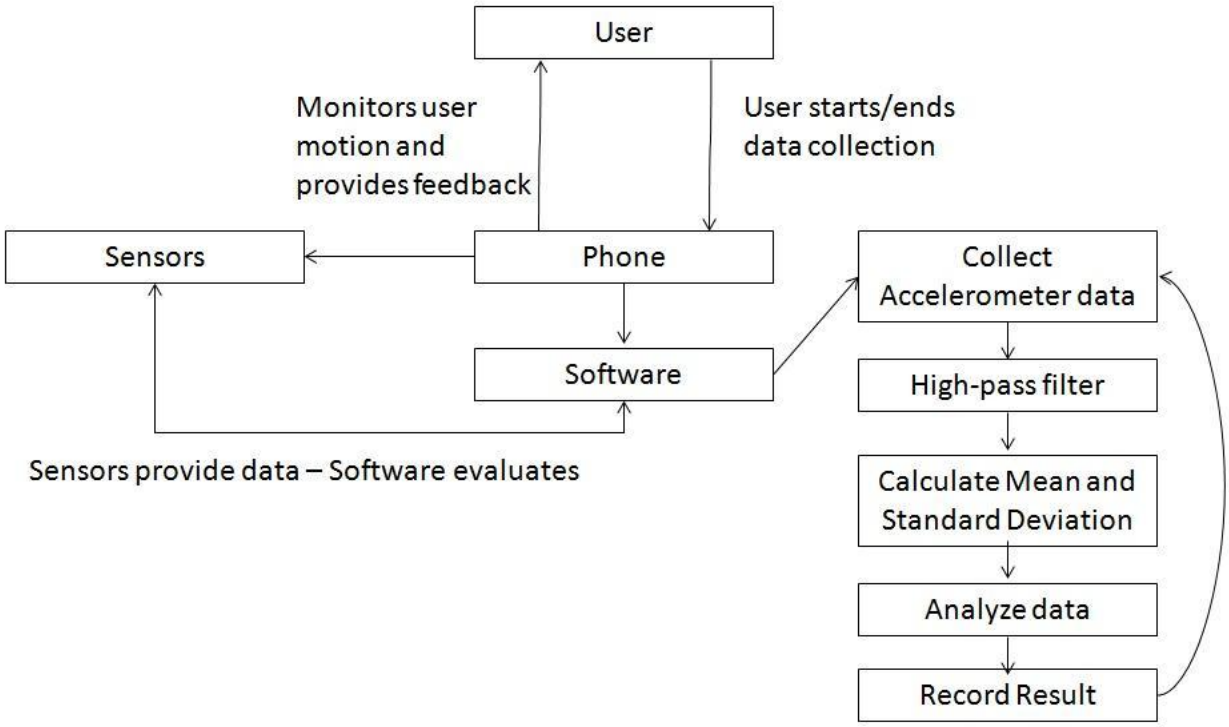

Figure 5. High level system design.

Once the user starts the app and after an initial countdown, the accelerometer begins reporting data at its fastest speed - for the HTC One, this is approximately 99 Hz. A high-pass filter is then applied to this raw data to remove the force of gravity. The filtered data is then used for calculating the mean and standard deviation. These calculations, along with the filtered data, are used for analysis to find any anomalies that might indicate bumps or other undesirable road conditions. The verdict from the analysis and the relevant data is recorded into two CSV files that can be accessed using Excel from the phone's internal storage. This process repeats until manually stopped by the user. A representation of this is given by Figure 5.

5. APPROACH

Gathering the accelerometer data and applying a filter to it is the simple part. The data analysis is where things get tricky. The first step was to choose which axis to focus on: x, y, or z. In all the related work on this subject, the axis chosen was the one that measured the force of acceleration perpendicular to the ground. Since the phone is to be strapped to the user's upper arm in a vertical position, the axis that needs to be examined in this case is the y axis. This can be confirmed by Figure 6.

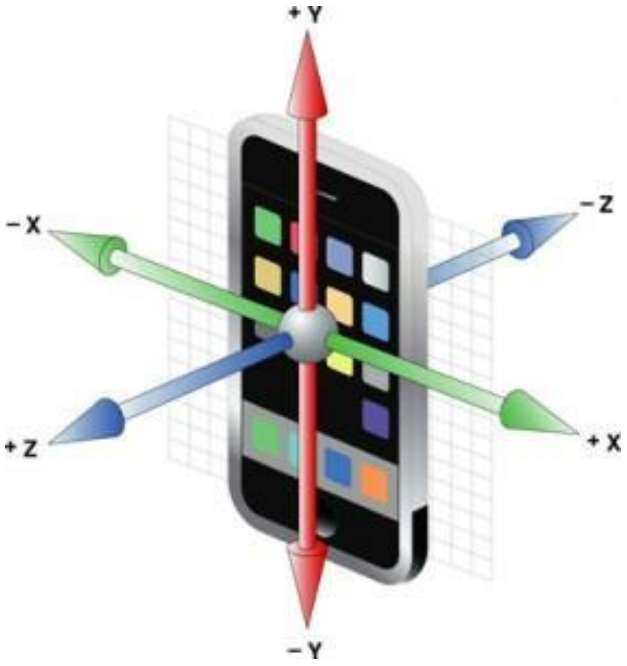

Figure 6. Phone axes.

Once the filtered data is obtained from the y axis, two calculations are made with the data. 1 – the population mean and 2 - a sample standard deviation. The formulas used are given in Figure 7.

$$\overline{X} = \frac{\Sigma X}{n} \qquad SD = \sqrt{\frac{\sum (x - \bar{x})^2}{N - 1}}$$

Figure 7. The sample mean and standard deviation formulas.

The mean is updated whenever new data is gathered; however, the standard deviation is only updated once every 10 seconds from a sample size of 100. This ensures that the individual jogging style of the user or changes in stride length are taken into account and not reported as false positives.

Once the mean and standard deviation are established, all new filtered data is compared to the mean plus or minus the standard deviation multiplied by a constant. [mean ± (standard deviation * constant)] This create a threshold that, when exceeded, registers as a bump. Every 10 seconds the program evaluates the road conditions by looking at the bump count. The results are added to a CSV file with relevant data, and the bump count is reset to 0.

Time (s)	Mean (m/s^2)	Standard Deviation (m/s^2)	Confidence	Condition
20.176	0.056673	4.805788	97.70%	AVERAGE
30.176	0.010596	5.211949	97.70%	AVERAGE
40.176	0.014574	5.295028	97.70%	HIGH

It required a good deal of testing to find a suitable constant to use for the determination of the threshold. As is known, standard deviation encompasses two-thirds of the data so the first number that seemed appropriate for the constant was 1.65. This would increase the threshold to cover 90% of the data, and anything outside of that range would be considered a bump. This threshold proved to be too low; causing so many false positives that even a good quality road would have a high bump count. The constant was eventually increased to 2.33 – a 98% threshold. At this point the app was not sensitive enough and would only pick up very large road anomalies. The constant was slowly scaled back and when it reached 2.28 – a 97.7% threshold – it became sensitive enough to capture bumps without a high rate of false positives. Figure 8 illustrates this process.

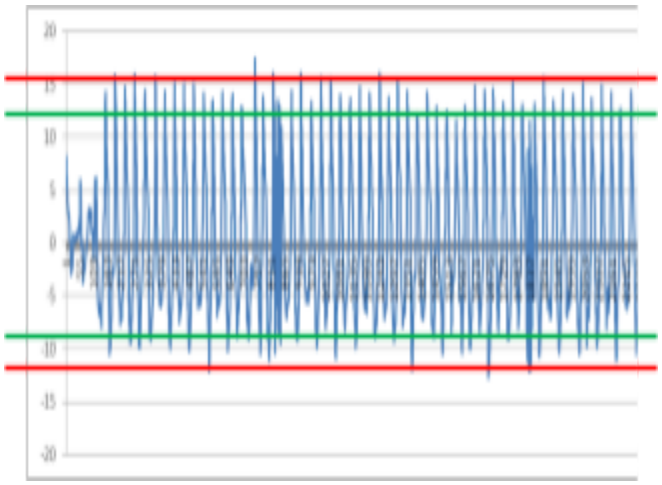

Figure 8. Typical filtered data output with specific threshold lines placed approximately.

Every 10 seconds the app looks at the bump count and uses it to rate the road's condition. If the count is 0 or 1, the road's quality is rated "High." One bump is allowed to account for false positives that might be caused by the user. If the count is exactly 2, the road is rated "Average." A count of 3 or higher is rated as "Low." The app also stores the GPS coordinates associated with the rating, and after the user stops the data collection the road condition ratings are added to a Google map. The colors chosen to mark the route represent the rating given: green is for "High," yellow is for "Average," and red is for "Low." This can be seen in Figure 4.

This app was written in Java using the Eclipse IDE with the Android SDK plug-in. The OpenCSV library was used to create and add data to CSV files. The Google Play Services library was used to add a Google map so that certain data could be displayed graphically and in context.

6. TESTING AND VERIFICATION

The app was tested on two sections of road that were each approximately ¼ of a mile long. One section of road was high quality with a smooth surface and a constant elevation. The other section of road was low quality, having a rough surface with many bumps and cracks. After significant adjustments were made to areas that might change the output, for example, sensor data collection rate or a change in threshold level, the app would be tested several times on each section of road. The data gathered from this testing is saved in two CSV files. Some examples of these files can be seen in Tables 1 and 2.

Table 1. Sample run results output.

Table 2. Sample filtered accelerometer data output.

Time (s)	X (m/s^2)	Y (m/s^2)	Z (m/s^2)
0.81	-6.4132	17.98726	7.37187
0.84	-5.63393	7.221577	2.08223
0.86	-5.07108	6.49888	1.873468
0.89	-4.56397	5.848992	1.686121
0.97	-2.34813	3.264312	1.345607

0.107	-2.11386	2.937342	1.210507
0.117	0.512248	-0.42531	-0.35851
0.127	1.737089	-3.00389	-1.6677
0.148	1.528892	-4.70382	-2.67354
0.149	1.375464	-4.23398	-2.40672
0.157	1.237917	-3.81058	-2.16605
0.167	0.114504	-4.87749	-3.29395
0.177	-0.10388	-5.45888	-3.89574
0.187	-0.09403	-4.91353	-3.5067
0.199	0.916075	-4.8355	-2.87959

This data is then examined to determine if the changes made have helped or hindered the detection of road conditions. Adjustments are made accordingly and the app is tested again in the same manner. Once the app seemed to be working well with one specific user, it was tested with different users to see if there would be a change in performance. Three different users have been tested and so far there the app appears to work equally well between each user.

The app has a high detection rate for large bumps; however, it has some difficulty distinguishing between small bumps and a deliberate variation in the user's motions, like a change from a longer to a shorter stride. This will likely always be the case when using a threshold test. Perhaps greater accuracy can be achieved if more tests are introduced for the determination of what is considered a bump.

7. REFERENCES

[1] Vittorio Astarita, Maria Vittoria Caruso, Guido Danieli, Demetrio Carmine Festa, Vincenzo Pasquale Giofrè, Teresa Iuele, Rosolino Vaiana, A Mobile Application for Road Surface Quality Control: UNIquALroad, Procedia - Social and Behavioral Sciences, Volume 54, 4 October 2012, Pages 1135-1144.

[2] Mednis, A.; Strazdins, G.; Zviedris, R.; Kanonirs, G.; Selavo, L., "Real time pothole detection using Android smartphones with accelerometers," *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on* , vol., no., pp.1,6, 27-29 June 2011.