

BUMPSTER: A Mobile Cloud Computing System for Speed Breakers and Ditches

Arhum Savera, Ahmed Zia, Muhammad Salman Edhi, Muhammad Tauseen and Jawwad Ahmed Shamsi
Systems Research Laboratory Computer Science
Department of Computer Sciences, National University of Computer and Emerging Sciences-FAST,
Karachi, Pakistan

Abstract—Road accidents are one of the universal problems faced by people all around the world. A major reason is reckless driving on speed breakers and ditches causing high casualties due to no warning signs, lack of street lights, substandard construction etc. There has been some research related to finding a Computer Science oriented solution but no substance in terms of a product. We present an Android application to solve this and have collected data from different parts of Karachi to detect upcoming speed breakers and ditches within a 10-12 meter radius. The application will be able to accurately warn users ahead of time and prevent hazards using a Support Vector Machine which has been trained with data from multiple devices. Speed breaker and ditch locations are loaded on the Cloud and filtered using a defined threshold. The model predicted speed breakers and ditches up to 85% accuracy with minimal power consumption.

Keywords: *Cloud Application, Mobile Application, Sensor-based Systems, GPS, Road Condition, Machine Learning, Mobile Cloud Computing*

I. INTRODUCTION

As society becomes more fast-paced, so do our speeds. Unfortunately the road conditions of cities around the world aren't always properly maintained enough to be relied on. Speed breakers were introduced as a means to control speeds of vehicles and maintain order, but as time has progressed they are now constructed with poor standards, great variation of height and width and have therefore proven to do more harm than good, especially in third world countries. Due to several reasons such as rain, garbage, and rough driving, roads do not last in terms of quality and eventually give away to ditches which are one of the hazardous causes that contribute to accidents. A news report reported that almost 50% of the speed breakers in Karachi were hazardous and constructed poorly. This, along with a lack of warning signs and proper maintenance bring the need for a system for the everyday driver to be aware of upcoming road anomalies to avoid and prevent a potential accident.

We present Bumpster, an Android application to alert drivers of upcoming speed breakers and ditches. The application is suitable as it uses built in sensors including the GPS sensor and the accelerometer which are available in every android smart phone.

Our work in this paper is inspired by the work put forward by M. Jain, A. P. Singh, S. Bali and S. Kaul [4], who presented a detection methodology for speed breakers. However, unlike the work presented by Jain which provides

alerts about speed breakers through a pre-downloaded data in the mobile, our work is more comprehensive and extensive. In that, we provide a comprehensive mobile application, available at [1], which is based on self-learning model for speed breakers and ditches. We utilize machine learning techniques to adopt to changes in road conditions and incorporating crowd sourcing to collect data from across a city. We utilize cloud services to dissipate information to users. Our model is generic and can be applied on any city. As a test case, we applied Bumpster on Karachi, the largest city of Pakistan and seventh largest in the world.

Our contribution in this paper can be highlighted as follows:

- We define a cloud based architecture for a detection system which can be used in any location.
- We describe a data collection method for vehicle movement, independent of smartphone position.
- We describe a method to alert users ahead of time based on a distance threshold
- We describe a cloud infrastructure to store and maintain all incoming GPS data, including an update and removal mechanism for speed breakers or ditches which can be added or removed in a city.
- We present different attempts to distinguish between speed breakers and ditches using built in Android sensors.

The paper is organized as follows: Section 2 presents the related work on the idea with the adopted methodology of different authors. Section 3 presents the overview of our system architecture along with our opted design and methodology of data collection and analysis. Section 4 presents all the challenges faced along with different tests. Section 5 discusses the results of speed breakers and ditches detection. Section 6 concludes with future enhancements.

II. RELATED WORK

Many methods and algorithms have been proposed by different researchers to prevent these accidents. One method was proposed by Ravi et al [2] called "Wolverine method"

in which the basic requirement for speed breaker detection is the magnetometer android sensor and a constant reorientation of axes. Also it uses K-fold Machine learning algorithm due to which it was able to reduce false positives but also added about 10% of false negatives. There was no discussion on update or removal of speed breakers and one of the drawbacks was if the mobile position had been disturbed then the whole collected data would be inaccurate.

Following the same algorithm and method another refined algorithm was proposed by Prashant et al [3] which uses android accelerometer sensor, GPS, cellular radio and microphones for detection of speed breakers as well as reorientation. The experiments performed were more accurate but the application forcefully killed background applications to increase processing power and save energy costs. The researchers did not discuss about the false positives of their method and no update or removal methods of new speed breakers were discussed.

Mohit et al [4] proposed an algorithm using accelerometer sensors [5] for reorientation and but their data did not come from detection, rather was fed into the application by pressing a marker on a separate device indicating whenever a speed breaker had been passed over. The experiment was carried out in bus, rickshaws and cars using different smart phones. The crowd sourcing data collection method [6] [7] was adopted to keep plenty of data of different locations within Bangalore and the results were not discussed with no threshold point to define what classifies as a speed breaker in their proposal and also there was no update or removal mechanism but this method can be improved upon and refined to make it more generic so that it can be applied anywhere more efficiently.

Gunjan Chugh et al discussed and summarized the techniques of various other works while comparing the sensors needed. The work presented in this paper also uses the Accelerometer and GPS but does not require sensors mentioned such as Microphone, Magnetometer, OBD device and Wi-Fi for detection. They suggested that Machine learning techniques may be used for detection. We have implemented such a method [8].

Roma Goregaonkar et al Developed a fall monitor using similar sensors used in this paper and designed a system for alerts in the event of an accident. Their system was designed that phone orientation and position inside the vehicle had to be configured in a fixed location to receive accurate results as they were monitoring individual axes. We are using the overall magnitude of the three axes combined which frees the phone of being in a fixed position during a drive. On any incident detection, they used an SMS and GSM module to send alerts. We introduce alerts ahead of reaching the road anomaly as well as a mechanism for uploading and receiving new locations from a cloud server [9].

Roma Goregaonkar et al used in-built sensors along with a Bluetooth module and microcontrollers to develop an Accident monitoring system. The difference from our work is that their system required a connection to web server

whereas no internet connection is needed in our application during a drive to detect upcoming speed breakers. The Accident was detected only after the vehicle had collided. Our application aims to avoid any collision or accident ahead of time with its early detection mechanism. They proposed a way to detect speed breakers and potholes but the experimental set up was done only using one smart phone fixing the phone in a specific location and observing one axis whereas our data collection and algorithm training has used several phones from different vehicles [10].

Ramakrishnam Raju et al Uses accelerometer, gyroscope, magnetometer, GPS and clustering to detect driving patterns classified as normal, abnormal and aggressive driving. Their results were inconclusive when it came to detecting lane changes and bumps on the road. Our presented application only uses two sensors to detect speed breakers and ditches and uses Support Vector Machine classification in contrast to their PLMST algorithm along with a cloud component for prediction of upcoming anomalies [11].

Nidhi Kalra et al used android sensors to observe different road events including breaking, pot holes, bumps and lane turns. Their experimental set up was with a single phone in a fixed position and discussed the various trends in the acceleration axes during each road event. They did not factor in the position of the phone in the vehicle and can only detect bumps or a pothole given the phone is configured to a specific position. This application is independent of position when detecting speed breakers and ditches which distinguishes it from the aforementioned work [12].

Mohamed Fazeen et al used one Google Nexus phone to detect road anomalies and discussed classification results. Our work differs from this in the way of data collection and feature generation. Also, we have incorporated the use of GPS to record these locations and filtering as a two-step process being carried out at the phone and the cloud server as well to reduce the number of false positives [13].

III. DESIGN AND METHODOLOGY

A. System architecture

The system architecture can be depicted in figure 1. The user will be using the application on his/her android device. The simple interface requires minimal user interaction. A simple start and stop button mechanism begins the service where the application will begin to collect the required data and check for upcoming speed breakers. When an upcoming speed breaker or ditch is detected the user will be alerted by a voice alert or vibration. When the user has completed the drive, the collected data is processed and that data which is useful for adding new locations is sent to a cloud server where all locations are stored of all users. These locations are then updated and new locations are sent as an update to users so the next time they approach a stored location they can be alerted ahead of time.

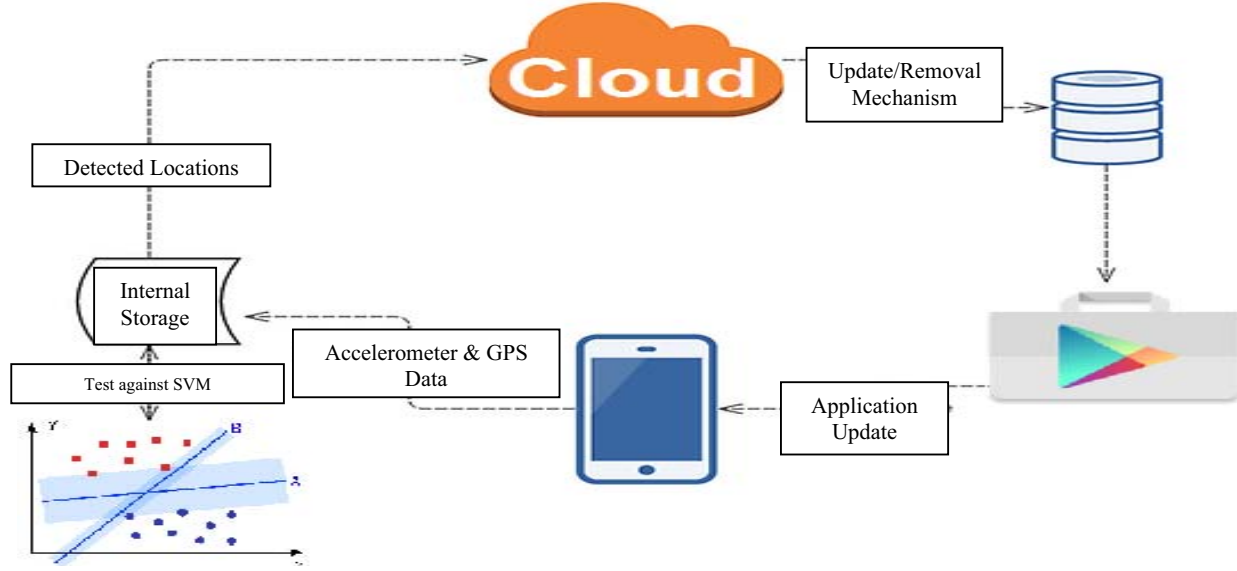


Figure 1. Cloud based system architecture.

Shapes of speed breakers vary from one another in terms of height and width. Our design does not distinguish between different shapes of speed breakers as the data detection methodology is the same for all types.

We decided to adopt the detection methodology as described by [4] for our product which utilized two sensors in Android smart phones, the Accelerometer and GPS sensor. The accelerometer is capable of detecting movement across three axes as shown in figure 2. Using the overall amplitude is preferred when wanting to keep the mobility of the smart phone. This is calculated by $a = \sqrt{x^2 + y^2 + z^2}$ where x, y and z are the respective axes of the device. Other methods would require the phone to be in a fixed position in the vehicle and be in a certain orientation. The amplitude calculated will represent the overall acceleration of the vehicle.

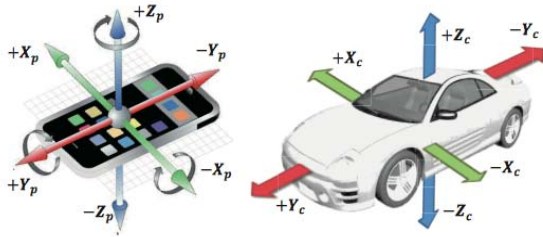


Figure 2. Axes of Smartphone and Vehicle.

B. Data collection

Accelerometer data was gathered from different areas of Karachi. Multiple phones were used along with different vehicles. On some drives, more than one phone was used to record drive data, but the data gathered was still unique due to the different positions of the phones. The total distance spanned was approximately 700km, covering some of the busiest and most visited parts of the city.

An android application was programmed to collect data which would be used for supervised learning and training a model which would be able to detect speed breakers and ditches. While the application is running, a log is maintained

which keeps track of all sensor inputs from the accelerometer from any change detected in any of the three axes and was broken down into two second intervals. A change is detected by using the android sensor managers and corresponding GPS locations were obtained by keeping track of the received GPS coordinates from the android location manager. As shown in the table 1, each row consists of the timestamp of when the change was detected, the overall acceleration of the phone which is derived from the individual axes as described by [4], and the corresponding longitude and latitude coordinates at that instance. Each line represents a change detected by the accelerometer.

Table 1: Acceleration data with GPS log and timestamp

Timestamp(sec)	Acceleration(m/s ²)	Longitude Coordinate	Latitude Coordinate
38.058..	9.327..	67.138..	24.898..
38.070..	9.612..	67.138..	24.898..
38.090..	9.485..	67.138..	24.898..
38.098..	9.889..	67.138..	24.898..
38.111..	8.552..	67.138..	24.898..
38.118..	7.745..	67.139..	24.898..
38.131..	8.500..	67.139..	24.898..
38.140..	10.112..	67.139..	24.898..
38.152..	10.909..	67.139..	24.898..
38.159..	10.768..	67.139..	24.898..

The data in Table 1 has been rounded to three decimal places for readability. All data was used to its highest available precision to maximize accuracy.

The raw accelerometer data gathered can be visualized as given in figure 3. A speed breaker was passed over during the intervals of 137 and 139 seconds. It can be seen that passing over a speed breaker or a ditch brings out a larger variance in a short amount of time. While travelling on a straight road, the value of acceleration is approximately constant with a value of 10, the value of gravity. When going over a speed breaker or ditch this value goes between ranges of 17 and 1. This is credited to the upward and downward motion the vehicle undergoes when going over a speed breaker.

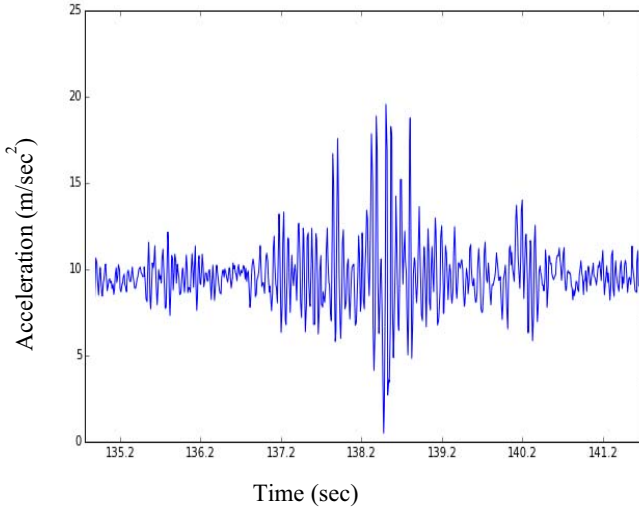


Figure 3. Acceleration(y) against time(x).

B. Feature Vectors

To collect data with labels, buttons to indicate when the car passed over a speed breaker were added to the interface. A marker was added to the log file to indicate that a speed breaker was encounter in that time window and the appropriate label was added when generating the features, as shown in Table 2 the five features are calculated from each 2 second window and each vector is assigned a label. The label 0 indicates straight road, 1 indicates a speed breaker and 2 represents a ditch. The features calculated are Standard deviation, Number of Mean Crossings, Maximum Mean Crossing Interval, Ratio of the Std. Dev of current to previous window and Ration of the Std. Dev of current to next window. These five features proved to be vital for differentiating a straight road from a speed breaker or ditch as the values of features 1,3,4,5 are distinctly higher and feature 2 should be lower in the case of a speed breaker or ditch than that of a straight road.

Table 2: Feature Vectors with labels.

Standard Deviation	Number of Mean Crossings	Maximum Mean Crossing Interval	Ratio of Current: Previous	Ratio of Current: Next	Label
0.274..	115	0.048..	1.064..	0.362..	0
0.357..	100	0.116..	1.762..	1.114..	0
1.679..	91	0.204..	1.897..	1.533..	2
0.443..	106	0.079..	0.652..	0.394..	0
1.123..	61	0.197..	2.535..	1.489..	1
0.293..	90	0.121..	2.040..	0.581..	0

C. Classification and testing

Data was gathered using multiple android phones and several different vehicles. It was then combined to create a training data set and then after simple preprocessing, a model was trained using a Support Vector Machine with kernel. We decided to use the SVM as it is an efficient machine learning method for classification and showed better efficiency on mobile in comparison to other methods such as KNN, which required that the entire training data set be stored and present at the time of classification of new instances. The SVM simply generates a model and stores parameters to define it, making the need to store training data unnecessary and speeding up classification. This model was then added to final product that the user would be using. The data gathered came from areas all over Karachi, this covered multiple types of speed breaker shapes, varying from thin and tall to short and wide ones, which would allow the model to recognize all kinds of bumps. The population ratio of the classes of data was kept near equal to avoid bias and ensure accuracy. Table 3 shows the results of the model on separate drives. The testing locations were different than the training areas to ensure that the model would be able to detect different kind of speed breakers of varying shape. The column ‘Total’ represents the actual number of speed breakers observed. In the worst case, our model shows a 70% accuracy at minimum. This can be credited to the phone being moved during detection as well as other variables including shape of the speed breaker being too low or the position of the phone in the vehicle.

The smartphones used in the tests shown in Table 3 were Samsung Galaxy Core Prime, Samsung Galaxy Note 3, Samsung Galaxy S5, Samsung Galaxy S4 and LG G2.

Table 3: Speed breaker classification Results on Mobile.

Test Drive	Total	Correctly Detected	False Alarms	Undetected
1	45	38	3	4
2	46	43	1	2
3	32	28	3	1
4	32	29	3	0
5	28	20	6	2

D. Early detection mechanism

When the application starts, all stored locations of speed breakers and ditches are loaded into memory. During the drive, after a set time interval, the application receives the users current GPS coordinates via location manager and it checks if it is approaching a speed breaker or ditch whose location was stored initially. These stored locations are confirmed to be true from the cloud server. For all these stored locations, the application calculates the distance to them using the Haversine formula which calculates the distance between two GPS coordinates. The Formula used to calculate distance d is described as follows:

$$a = \sin^2(\Delta\phi/2) + \cos \phi_1 \cdot \cos \phi_2 \cdot \sin^2(\Delta\lambda/2) \quad (1)$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a}) \quad (2)$$

$$d = R \cdot c \quad (3)$$

Where ϕ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km);

If any of the calculated distances is below a set value (10 meters) then this indicates that the car is approaching a speed breaker or ditch so an auditory alert is given to the user. This process is then repeated after a set time interval (2-5 seconds) to prevent redundant alerts of the same detected speed breaker or ditch. The application has been designed to not require an active internet connection during the drive which saves power and is beneficial to the user as a strong internet connection is not always available.

E. Cloud Server

Once the user has concluded the drive and stops the application, the log file is processed and feature vectors are generated and passed to the model. If the model identifies an instance as a speed breaker or ditch then it sends the corresponding GPS coordinates to the cloud where all incoming files from all application users are received. A database is maintained which consists of all GPS coordinates of speed breakers and ditches along with their frequency. When an incoming file is received it is parsed and each received GPS coordinate is compared with the already stored locations. Due to the inaccuracy of GPS Coordinates, exact matching is not suitable. Thus, if an incoming instance is within three meters or an already stored location, they are assumed to be the same. The frequency of that location is then increased. If a locations occurrence frequency exceeds a preset minimum value such as 5 then its entry is added to the master file which contains all speed breaker and ditch locations which

will be deployed to the users as an application update. This step serves as a further filtering step to make sure false positives are not included in the master file. Thus, this will refine the quality of stored locations by filtering at the user and server side. To establish an update mechanism in the case of the repair of roads to remove ditches or the removal of illegal speed breakers is necessary. Along with keeping frequency of locations, the last occurrence date and number of occurrences in a time period is maintained. If these fall below a certain value then that location is removed from the master file.

IV. CHALLENGES

A. Battery Consumption

The application is intended to be able to run in the background and will be collecting data from sensors and writing to files. The application code was revised and structured to be efficient and consume less power on mobile phones. Design changes such as reducing the number of file writes and the frequency of checking the users location showed to improve results. When checking for upcoming speed breakers or ditches, all the stored locations are read to memory instead of repeatedly accessing storage to improve performance. When the user is warned of an upcoming speed breaker or ditch then the application will not check for a few seconds because no other anomalies can come in between the currently detected upcoming anomaly and the user's current location. The results are depicted in Table 4 covering different smart phones and different lengths of use of the application.

The application was turned on and users were asked to use their phones as they normally would.

Table 4: Mobile battery consumption results. Test cases T1 and T2.

Mobile set	Android Version	30 minutes		45 minutes		60 minutes	
		T1	T2	T1	T2	T1	T2
Samsung Galaxy Core Prime	Android KitKat 4.4.4	4%	5%	6%	9%	9%	12%
Samsung Galaxy Note 3	Android KitKat 4.4.2	6%	3%	10%	5%	15%	6%
Samsung Galaxy S5	Android KitKat 4.4.2	2%	3%	3%	7%	6%	10%
Samsung Galaxy S4	Lollipop 5.0.1	8%	7%	12%	12%	17%	16%
LG G2	Android KitKat 4.4.2	7%	6%	11%	9%	14%	13%

B. GPS Inaccuracy

During a drive, the application logs both accelerometer and GPS sensor data. The inaccuracy of GPS coordinates of smart phones varies between 5-8 meters depending on factors such as number of available satellites and the surrounding environment. This introduced a challenge as speed breakers and ditches may not be detected with their true position. To tackle this, we used the Haversine formula with a threshold of 5 meters to assume two reported instances of a speed breaker or ditch to be the same.

C. Phone to be switched on

Another issue regarding collecting locations of speed breakers and ditches is that if the user's smartphone should power off during a session, then all potential detected locations will be lost and not sent to the server. To overcome this, the application was designed with simple start and stop buttons so the user can stop the application before her phone shuts down. The application was also designed to have minimal impact on battery performance to reduce the chance of such an event from occurring.

D. Speedbreakers vs. Ditches

The main challenge encountered was being able to differentiate between a speed breaker and a ditch as both had similar vibrations and patterns when data was collected. This problem proved difficult for a number of reasons. Though a speed breaker follows a general shape, a ditch does not. Even going over the same ditch multiple times gives different results as the pattern obtained depends heavily on the angle at which the car approaches it, the speed of the vehicle, the suspension system and height of the vehicle. There are many external variables which must be considered when tackling this problem. Though speed breakers and ditches can be distinguished in a controlled experimental setup, those approaches are not suitable for an application which is intended to be used in any position. The following section attempts to solve this.

V. RESULTS

A. Acceleration of Z-axis

Our initial attempt was to monitor the individual axes and find a distinguishing indicator between speed breakers and ditches. When the smart phone is laying in a straight horizontal position facing upwards, there should be a rise in the Z-axis when going over a speed breaker and a decline when going over a ditch [12]. The problem with this approach in regards to our product is that when the user is using the application, the phone can be in any part of the car and in any orientation. The method of monitoring the axes will only work if the phone is constrained to a fixed position. The graph in figure 4 shows the Z axis of a certain drive. Even though the spike at approximately 39 seconds is a true indicator of a speed breaker, the initial drop before the rise leads to the false prediction of a ditch, rendering this unsuitable to use as a strategy at a product level.

The fact that the phone can be used in any position and rotated at any moment also supports this decision.

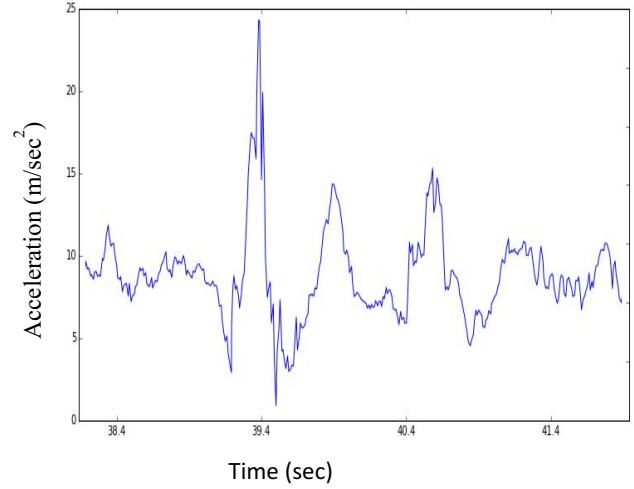


Figure 4 Acceleration(y) against time(x) of z-axis.

B. Calculation of Delta Values

While monitoring individual axes, another approach was to calculate the delta values and observe the trend of data approaching a speed breaker or ditch and understand how it changes. This is represented in figure 5. The variations in patterns were too heavily dependent on external conditions to be able to rely on a pattern as the depth and the width of ditches and suspension system of vehicles greatly affected the data obtained.

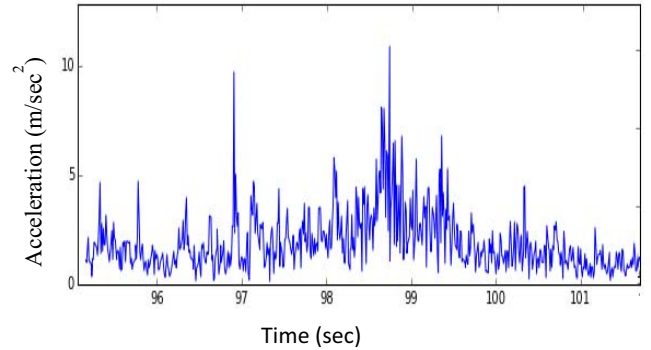


Figure 5 Change in Acceleration(y) against time(x).

C. Minimizing Sensor Delay

By adjusting the delay of the sensors in the application to its minimum value, the data gathered in a two second interval increased by 50%. It was expected that the increased amount of data would help identify distinguishing patterns between speed breakers and ditches. The data collected can be visualized in Figure 6. The setback of this approach was the addition of random noise which affected the overall detection mechanism and the values of the features which the model was using to classify a location as a straight road, speed breaker or ditch.

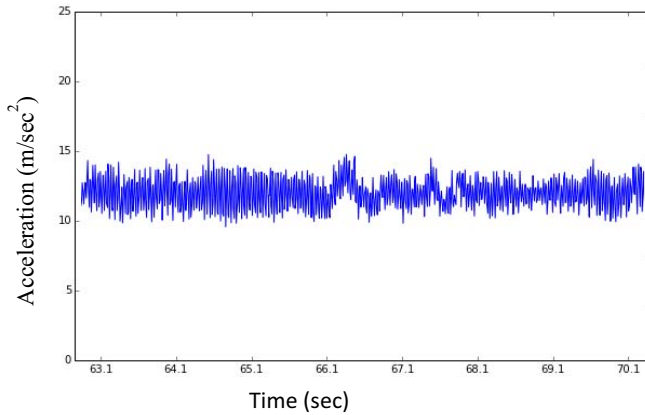


Figure 6 Acceleration(y) against time(x) with minimum sensor delay.

D. Moving Average

To eliminate random noise and to improve efficiency by reducing false positive predictions we applied a moving average technique on the data collected. This was intended to smooth the data and make it easier in detecting the main drop or rise which would indicate a speed breaker or ditch. The original and transformed data is visualized in figure 7. This technique proved to reduce the number of false positives but was ineffective in distinguishing speed breakers and ditches as the data gathered was heavily dependent on external conditions. The results were positive in controlled experiments but proved to not perform well in real life conditions.

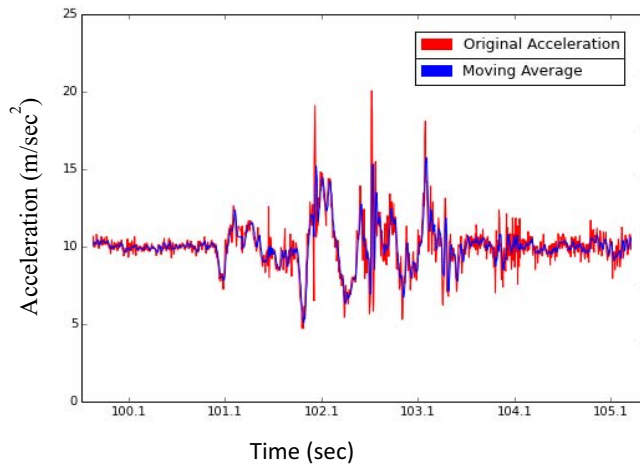


Figure 7 Acceleration(y) against time(x) before and after moving average technique.

VI. CONCLUSION AND FUTURE WORK

Using built in mobile sensors, we have presented a system which accurately detects and predicts upcoming speed breakers and ditches. This application is not limited to a fixed position and orientation inside the vehicle and is scalable to any city in the world. Running as a service, the application has been designed to have minimal power consumption. Challenges in distinguishing between Speed breakers and ditches are discussed.

The application can accurately warn users of speed breakers and ditches 10-12 meters ahead of the detected location.

Though the application is accurate in predicting speed breakers and ditches, it is still limited as not being able to distinguish between the two to a high degree of accuracy when mobile position and orientation is left to be in any position in the vehicle. These issues can be worked on in the future and resolved, possibly with the advancement of built in sensors in mobile phones. Another problem which can be solved is the GPS inaccuracy, this can be tackled as GPS coordinates can still be inaccurate to a few meters and thus the exact location of speed breakers and ditches cannot always be detected.

ACKNOWLEDGMENT

The authors would like to acknowledge Amazon Web Services for facilitating cloud services.

REFERENCES

- [1] "Systems Research Laboratory," [Online]. Available: syslab.khi.nu.edu.pk/
- [2] Ravi Bhoraskar, Nagamanoj Vankadhara, Bhaskaran Raman and Purushottam Kulkarni, *Wolverine : Traffic and Road Condition Estimation using Smartphone Sensors*, Bangalore: IEEE, 2012.
- [3] Prashanth Mohan, Ramachandran Ramjee and Venkata N. Padmanabhan, Nericell: Rich Monitoring of Road and Traffic Conditions, New York: Association for Computing Machinery, Inc., 2008, p. 14.
- [4] M. Jain, A. P. Singh, S. Bali and S. Kaul, *Speed-Breaker Early Warning System*, Boston, 2012, p. 6.
- [5] P. Singh, N. Juneja and Shruti Kapoor, *Using Mobile Phone Sensors to Detect Driving Behavior*, Bangalore, 2013.
- [6] Dejun Yang, Guoliang Xue, Xi Fang, Jian Tang, *Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing*, New York: ACM, 2012, pp. 173- 84.
- [7] G. K. A. Chatzimilioudis, C. Laoudias and D. Zeinalipour-Yazti, *Crowdsourcing with Smartphones*, vol. 16, no. 5, pp. 36-44, 12 June 2012.
- [8] Gunjan Chugh , Divya Bansal and Sanjeev Sofat, *Road Condition Detection Using Smartphone Sensors: A Survey*, vol. 7, Chandigarh: International Research Publication House, 2014.
- [9] Roma Goregaonkar, Prof. Snehal Bhosale, *Safe Driving and Accidental Monitoring Using GPS System and Three Axis Accelerometer*, vol. 3, International Journal of Emerging Technology and Advanced Engineering, 2008.
- [10] Roma Goregaonkar, Snehal Bhosale, *Driving Assistance and Accident Monitoring Using Three Axis Accelerometer and GPS System*, vol. 3, Pune: International Journal of Science and Research (IJSR), 2014.
- [11] Ramakrishnam Raju, Valli Kumari, Azad Naik, *Driving Pattern Recognition Using Parameter-Lite Clustering Approach*, vol. 2, International Journal of Research in Advent Technology, 2014.
- [12] Nidhi Kalra, Gunjan Chugh and Divya Bansal, *Analyzing Driving and Road Events via Smartphone*, vol. 98, International Journal of Computer Applications, 2014.
- [13] Mohamed Fazeen, Brandon Gozick, Ram Dantu, Moiz Bhukhiya, and Marta C. González, *Safe Driving Using Mobile Phones*, vol. 13, IEEE, 2012.