iRoads - Smartphone-Based Road Condition Monitoring

FINAL YEAR PROJECT REPORT

Group 26 - codemo

H.M.A. Abeywardana (140011X)

U.M.J. Abeywikrama (140014J)

P. T. Amarasinghe (140024N)

R.P.D. Kumarasinghe (140323F)

Internal Supervisor

Dr. H.M.N Dilum Bandara

External Supervisor

Dr. H.R Pasindu

Degree of Bachelor of Science of Engineering

Department of Computer Science & Engineering

University of Moratuwa

Sri Lanka

November 2018

Declaration page of the candidate & supervisor

We declare that this is our own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, we hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute our thesis, in whole or in part in print, electronic or other medium. We retain the right to use this content in whole or part in future works (such as articles or books).

Name	Signature	Date
H.M.A. Abeywardana (140011X)		
U.M.J. Abeywikrama (140014J)		
P. T. Amarasinghe (140024N)		
R.P.D. Kumarasinghe (140323F)		

The above candidates have carried out research for the BSc Degree final year project thesis under my supervision.

Name of the supervisor: Dr. H.M.N Dilum Bandara

Signature of the supervisor: Date:

Abstract

Measuring and monitoring road conditions is essential to ensure public and vehicle safety, promptly maintenance, as well as fuel and time savings. While developed countries use sophisticated devices installed on specialized vehicles to measure and monitor road conditions, it is cost prohibitive for countries like Sri Lanka. Moreover, diversity of road types and non-standard physical properties make it impractical for specialized vehicles to travel on roads in Sri Lanka and many other countries. Therefore, we developed a crowdsourced, smartphone-based, and low-cost road condition monitoring solution that can be used anywhere and anytime on any road and vehicle.

We found crowdsourcing is a viable option because the proposed mobile app can be used to detect the road conditions such as potholes and bumps, as well as estimate International Roughness Index (IRI) at a high spatial and temporal granularity. Sensors such as 3-axis accelerometer, GPS, and magnetometer and GPRS connections included in most smartphones used to collect data. We use a combination of signal processing and machine-learning techniques to detect anomalies, estimate IRI, and classify road segments based on IRI. Moreover, we developed a map-based dashboard to visualize the data, as well as estimate IRI values and location of anomalies. The proposed solution can identify road anomalies with a precision of 92% and estimate IRI similar to a class three road profiling instrument with a MAE of 0.3913.

Keywords. 3-axis accelerometer, Crowdsourced, IRI, Magnetometer, Smartphone-based.

Acknowledgments

We wish to sincerely thank our supervisor Dr. H.M.N. Dilum Bandara for providing us with research ideas and supervising our work continuously. He provided us with necessary guidance and encouragement to fulfill our objectives. Also, the support and guidance received from Dr. H. R. Pasindu along with other staff members in the Civil Engineering Department was very resourceful for completing our research more accurately. Special thanks to Mr. Sasika Ranawaka and Mr. Kalum Sadamal for the support given to operate the road profiling equipments. We would also like to thank the other members of the evaluation panel Dr. Chathura De Silva and Dr. Kutila Gunasekera for pointing out new approaches and other valuable feedbacks provided. And also we would like to extend our gratitude to all the academic staff of the University of Moratuwa for the great work they did for us during the course of study. Also, we are grateful to our family members and all the friends who helped us throughout the project.

Table of Content

List of Figures

List of Tables

List of Abbreviations

API	**Application Programming Interface**
CSV	**Comma-Separated Values**
eIRI	**estimated International Roughness Index**
GPS	**Global Positioning System**
GSM	**Global System for Mobile communication**
IRI	**International Roughness Index**
JSON	**JavaScript Object Notation**
MAE	**Mean Absolute Error**
MEMS	**Micro Electro Mechanical System**
OBD	**On-board diagnostics**
OBD2	**On-board diagnostics version 2.0**
PC	**Personal computer**
RDA	**Road Development Authority**
RMS	**Root Mean Square**
ROMDAS	**Road Measurement Data Acquisition System**
SDK	**Software Developer's Kit**

1. Introduction

1.1. Background

Due to lack of road condition monitoring and measuring in Sri Lanka pedestrians, passengers, drivers and vehicles face to various safety issues. Developed countries use sophisticated devices installed on specialized vehicles [1] to measure and monitor road conditions. However unfortunately, Countries like Sri Lanka cannot afford those kinds of high tech devices mainly due to their cost. Also, due to the diversity of road types and non-standard physical properties make it impractical for specialized vehicles to travel on roads in Sri Lanka and many other countries. Hence, it is essential to look into alternative and cost-effective means of measuring and monitoring the road conditions. Recent technological advancements in the domains of smartphones, sensors, wireless broadband connectivity, and crowdsourcing could be combined to derive low-cost road condition monitoring solution that can be used anywhere, anytime, any road, and on any vehicle.

1.2. Problem Statement

Low cost and pervasive solutions are desirable to measure and monitor road conditions in developing countries while addressing the problems related to diverse road structures. Such a solution could be developed using smartphones where sensors such as 3-axis accelerometers, gyroscope, and GPS available in the smartphone could be used to measure a vehicle's response to varying road conditions. Aggregation of such sensor readings from multiple users that use the same roads again and again through crowdsourcing could enhance the accuracy of detection of road conditions as well as road quality metrics such as IRI [2]. However, it is not straightforward to use sensor readings from 3-axis accelerometers, gyroscope, and GPS to detect and measure the road conditions. For example, a time series of acceleration readings needs to be converted to IRI to determine road quality. Moreover, different vehicles and smartphones will detect somewhat different physical responses when they go over a pothole or a bump depending on their quality of suspensions, speed, whether they fully or partially went over the pothole, orientation of the phone, and sensor quality.

Therefore, it is essential to be able to correct for biases, errors, reorient the smartphone, and calibrate the sensors readings with IRI readings from high-precision monitoring equipment. Thus, the problem to be addressed by this project can be stated as follows:

How to develop a smartphone-based, crowdsourced road quality monitoring solution that could estimate IRI and location of potholes?

1.3. Objectives

Following objectives are to be achieved to address the above problem:

1. Develop an app that can collect sensor data and reorient the acceleration vectors of the phone.
2. Integration with OBD2 for speed & fuel consumption data. Calculate fuel consumption of vehicles using engine rpm collected from OBD2 adapters.
3. Statistics, Signal Processing Machine Learning solutions to find IRI & potholes and cluster them according to GPS coordinates.
4. Calibration and performance monitoring. Use ROMDAS bump Integrator [3] device to compare and improve the results of IRI calculating algorithm.

2. Literature Review

Several research work have been carried out to find a method to measure the conditions of roads using different data collection approaches. When considering the sources for creating road profiles, researchers have used accelerometers in separate circuits and accelerometers inbuilt within smartphones. So using a smartphone can be considered a more viable option than designing separate circuits to collect acceleration data.

2.1. Smartphone Sensors

2.1.1. Accelerometer

The accelerometer in a smartphone is a circuit based on MEMS. Forces of acceleration caused by the gravity of movement or tilting actions can be sensed by this mechanism. MEMS measures these moving or gravitation accelerations of the attached device. Acceleration is given according to x, y, z-axes relative to the phone as shown in Figure 5.1.

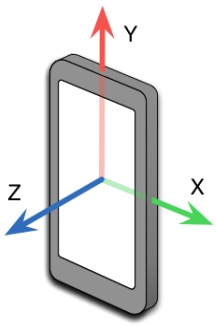


Figure 2.1. Three-axis acceleration vectors.

2.1.2. Magnetometer

The magnetometer is crucial for detecting the relative orientation of a mobile device relative to the Earth's magnetic north. It detects Earth's magnetic field along three perpendicular axes X, Y, and Z. The hall-effect sensor produces a voltage which is proportional to the strength and polarity of the magnetic field along the axis each sensor is directed.

2.2. Reorientation

Smartphone also delivers the advantage of easy implementation of a crowd-sourced application for data collection. Since a smartphone could be in any arbitrary position, issues like disorientation need to be resolved when collecting acceleration data. Road quality and Ghats complexity analysis using Android sensors [4] is proposing a solution to road quality monitoring using accelerometer and GPS sensors of the mobile to implement the system. To keep the mobile phone's axes and the vehicle axes in the same direction they have mounted the mobile into vehicle accordingly. Researchers have come up with a solution including an algorithm based on acceleration deviation of the x, y, and z-axes. Acceleration deviation on x, y and z-axes is a prominent way to identify road conditions because the acceleration deviation is dependent on the road condition. The algorithm must be improved to get correct acceleration data when the orientation of the mobile is changing which need to be addressed using a reorientation mechanism introduced by Nericell [5] and Wolverine [6].

Nericell [5] uses an accelerometer, microphone, GSM Radio and GPS sensors available in smartphones. It uses accelerometer readings to achieve virtual reorientation on each individual phone. Here Nericell uses Euler angles to find orientation and convert accelerations into the proper orientation.

Wolverine [6] has focused on virtually reorienting acceleration vectors of the mobile and processing them to identify road conditions. The application first reorients the acceleration vectors into geometric directions (North, East, Vertical down) using a magnetometer. Then it reorients the geometrically oriented vectors into vehicle X, Y and Z axes by identifying moving direction of the vehicle using GPS and magnetometer (By calculating the angle of the line made by two latitude-longitude points with geometric north). This reorientation uses a predefined matrix of trigonometric equations to simply calculate the vehicle acceleration from smartphone accelerometer vectors. Finally, it identifies the road surface anomalies and braking events of the vehicle by processing the acceleration vectors which were reoriented into vehicle axes with the use of machine learning techniques.

If the system is using an unfixed device, reorientation of the acceleration vectors is the first concern. Nericell [5] and Wolverine [6] mechanisms have addressed that concern.

2.3. Anomaly Detection

By analyzing these gathered acceleration data road anomalies need to be identified with a considerable accuracy. So using thresholds to identify anomalies has been used by some researchers while more recent research works are more focused on machine learning approaches. So filtering out other disturbances and noise in acceleration datasets due to vehicle vibration and sudden vehicle movements has a considerable influence on creating optimum prediction model.

To detect anomalies on roads using the sensor readings, Pothole Patrol [7] application uses z-peak values (i.e., vertical accelerations) within a selected time interval. If the z-peak value does not exceed the threshold value, then application rejects that period. If some period exceeds the threshold value, then sensor data is sent to the server to identify the anomaly as whether a pothole or not, using machine learning. Also, it uses GPS location-based clustering to get more accurate results. If data from k vehicles suggest a specific GPS location as a pothole, then it suggests that location as a pothole. Due to the three to four-meter error range in GPS [8], it suggests a pothole after considering a GPS range. Also, it blacklists some GPS locations if they are identified as other road anomalies.

Pothole Patrol [7] application sends accepted sensor data with time, speed, location and heading to the server for post data processing. Filtering algorithms used by the application is capable of filtering out abnormal sensor data using the vehicle's speed. If a vehicle is traveling at a low speed, the mobile application will filter out unrelated events like door slams because events like door slams will give a spike in accelerometer and led to wrong decisions. Researchers have used high speeds to detect braking and turning events detected in accelerometer to ignore them. Also, Pothole Patrol can reject veering properties of the vehicle at high speeds. Since it calculates speed using GPS, the speed of the vehicle won't be much accurate. Hence, OBD2 telematics would be used in iRoads application to get vehicle speed directly from the vehicle. As speed is a primary factor to reject abnormal sensor data, Pothole Patrol [7] is not very much accurate with speed-related decision making. Also, it hasn't considered a crowdsourced solution whereas Nericell [5] uses smartphone sensors which can be crowdsourced. Nericell [5] uses spikes along the vertical direction to identify potholes and bumps. To differentiate bumps from potholes Nericell [5] uses an external database of bumps. But Pothole Patrol [7] provides a more practical solution to distinguish potholes from road bumps. Since only one wheel of the vehicle meets the pothole, Pothole Patrol [7] considers spikes along other axes of the accelerometer to identify potholes. In a bump, wheels on both sides of the vehicle will meet the bump. Hence, a bump will not lead to spikes in the axis which is perpendicular to both vertical direction and vehicle's moving direction.

According to researchers of Road condition monitoring and alert application [8], raw data gathered from different sensors at a low sampling rate helps to reduce the power usage of the application. Also, an efficient classifier algorithm was used by them to provide reasonably accurate results and a confidence score was generated for each of the identifiers like pothole, bump etc. Then those results including location, score and identifier were sent to the web server to store in a remote database.

The backend server of this system [8] consists of a geospatial database that is able to store the location as points and the identifier, score as attributes to those points. Since there can be miscalculations and false readings in the data received by the back-end

server, multi-user fusion algorithm is used to mark the locations as anomalies after aggregating data from multiple users. Based on a threshold value, the application decides whether the condition indicated by the score exists on that location and if the answer is yes then that location is plotted as an anomaly on the map. This approach of the researchers helps to increase the reliability of the system which is critical to make decisions on the maintenance work. Other than the designs used in Nericell [5] and Road condition monitoring and alert application [8], iRoads focused on precise anomaly identification by using machine learning techniques on collecting data.

2.4. IRI Calculation

There was no significant amount of research carried out on calculating or predicting IRI for given road segments. So this research has looked into providing a road profiling solution similar to ROMDAS Bump Integrator which is a standard class 3 road profiler. Previous research work [9] has proven that the IRI value has a considerable correlation with the deviation of acceleration data. Based on that founding discovery, this research has used that correlation to predict the IRI values for a given road segment. Also, with the intention of improving the accuracy of the IRI prediction values and showing the correlation between fuel consumption of a vehicle and the IRI, vehicular data was collected using OBD2 adapters.

Research work on Pavement roughness evaluation method for low volume roads [9] suggests a technique to estimate road roughness using smartphone sensors. Accelerometer and GPS location data was obtained from both Roadroid [10] and Androsensor [11] applications. Authors performed regression analysis to identify a relationship between eIRI taken from Roadroid [10] application for every 20-meter road section and resultant acceleration has taken from Androsensor [11] application. Roadroid application can only calculate eIRI when driving speed of the experiment vehicle is 20 km/h or faster. Resultant acceleration is the resultant of the x, y and z accelerometer readings are taken from Androsensor application for 20-meter road sections. From the analysis, it has been found that acceleration data from smartphones has a linear relationship with road roughness condition. Hence, it opens the way to develop a system to measure road roughness using smartphone sensors.

Since there is no robust implementation of these researches, there is a need for designing a more reusable and accurate way of monitoring the condition of road networks.

3. Methodology

We believe crowdsourcing is a viable option as a well-crafted mobile app could be used to measure the road conditions such as potholes, bumps, speed breakers, and estimate International Roughness Index (IRI) at a high spatial and temporal granularity. Sensors such as 3-axis accelerometer, GPS and magnetometer included in most smartphones could be used to detect potholes and bumps, as well as estimate IRI and classify road segments based on IRI values using a combination of signal processing and machine-learning techniques. Moreover, we plan to and visualize this information using a map-based dashboard.

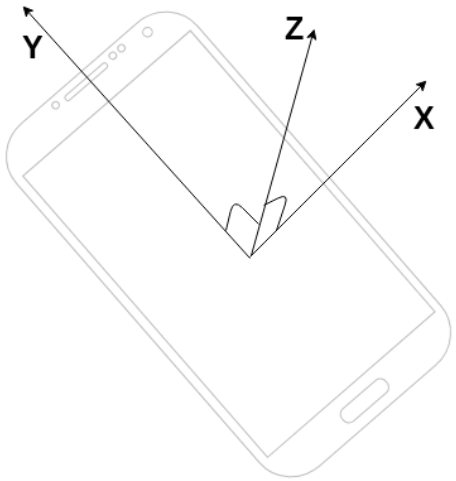


Figure 3.1. Smartphone in an arbitrary position.

We use 3-axis accelerometer as the main source for road profile evaluations. Because our solution is based on crowdsourcing, a reorientation mechanism is essential to convert accelerometer data from any arbitrary smartphone position as shown in Figure 6.1 respect to the vehicle's axis. We use a reorientation mechanism proposed by Nericell [5] and Wolverine [6] to overcome this dis-orientation issue. To remove noise in sensor inputs we use signal processing mechanisms to filter out the sensor noise.

Also, raw data and reoriented data collected from the mobile app are sent to backend servers for further processing. Finally, machine-learning and signal processing happens in the backend servers for detection of potholes and bumps as well as to estimate IRI. Finally, road segments will be classified based on IRI and a map will be annotated based on the IRI values. Moreover, we will improve the IRI calculation process to get more accurate results by comparing IRI values from our solution with ROMDAS bump integrator [3]. We further plan to identify the correlation between IRI and fuel efficiency of vehicles.

3.1. High-Level Architecture

Figure 3.2 shows the high-level architecture design of the monitoring system which consists of a mobile application and backend servers. The mobile application which is capable of collecting sensor data and database server is syncing through a sync gateway. Backend server is providing an API service to access collected data and processed data by machine learning models and signal processing approaches. Frontend is visualizing the data providing by the backend API.

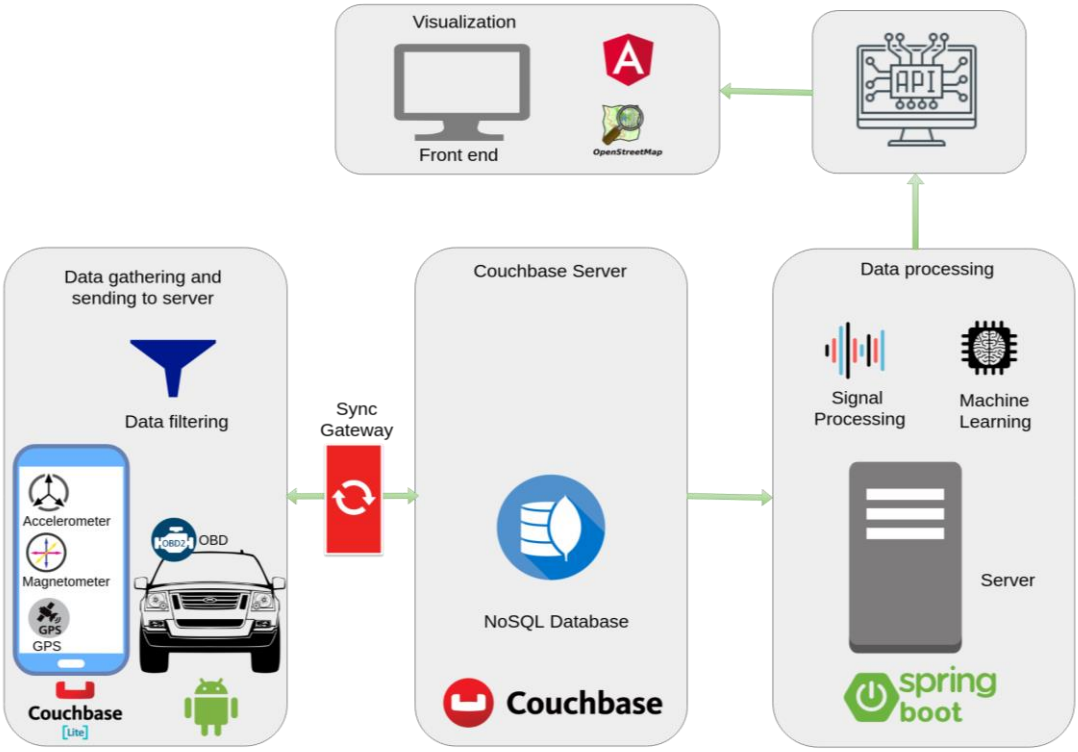


Figure 3.2. High-level architecture.

3.2. Mobile Application Architecture

Since the mobile application needs to utilize a lot of sensors and other native functionalities in the mobile device, the app is developed as an Android application. Also, Android is more supportive with external libraries [13, 14] that are required for the app development. As shown in Figure 3.3 the mobile application is designed with the objective of fulfilling both data gathering and in-app data processing capabilities. Average filtering and noise removal operations are carried out on raw sensor data like accelerometer readings. GPS coordinates are used to calculate the speed of the vehicle or OBD adapters can be used to get more accurate vehicular data. All these raw, filtered and processed data received from sensors are send to device storage and store there until a network connection is available to sync with the remote database server.

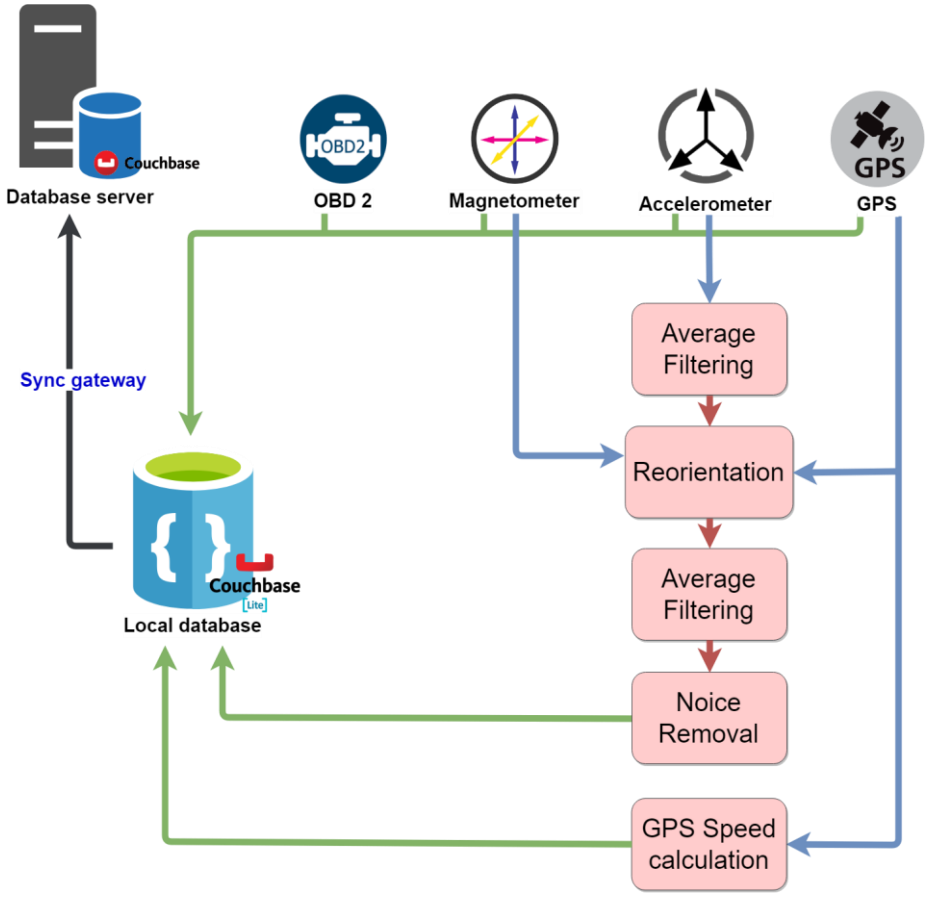


Figure 3.3. Architecture diagram of iRoads mobile.

3.2.1. Signal Processing

Using signal processing inside the mobile, the system eliminates the noise generated by the smartphone and vibration due to the vehicle engine. Here system uses a simple moving average filter on raw accelerations. Also, the system calculates average constant noise caused by the vehicle engine using a simple moving average filter. As the final step system reduces this constant noise from the filtered accelerations to obtain more refined values.

3.2.2. Reorientation Mechanisms

The system is capable of reorienting acceleration along any axis into a stable position. During the reorientation process system first applies a signal filtering process in order to remove the noise. As the next step system performs the reorientation using one of the following methods. Then again system performs a signal filtering process to remove constant noise caused by the vehicle engine.

3.2.2.1. Nericell

Through this reorientation mechanism, the system uses only acceleration vectors of the mobile. Using the acceleration vectors, the system calculates Euler angles for the mobile's orientation. Then with these Euler angles, the system converts the acceleration vectors into the stable position of the mobile.

3.2.2.2. Wolverine

Through this method of reorientation, the system uses the rotation vector generated by the mobile. The mobile device generates this rotation vector using magnetometer readings. Using the rotation vector, the system converts any acceleration vector from any arbitrary position to geometrical coordinates. Then the system uses the GPS bearing of the vehicle to convert these accelerations to the coordinate system of the vehicle.

3.2.3. GPS speed calculation

The system calculates vehicle speed using GPS to use in scenarios where the vehicle doesn't support OBD2. In order to calculate speed, the system calculates the distance between two GPS coordinates using the haversine formula[15]. By using this distance and time difference between two GPS coordinates system calculates vehicular speed.

3.2.4. Automatic Activity Recognition

With the intention of making the application more user-friendly, Pathsense [14] Activity SDK was integrated into the mobile application. This Activity Recognition tool currently supports activities such as Walking, Driving, Holding, Still, Shaking, and In-Vehicle Holding. So the sensor data will only be saved in the local database if the current activity is Driving. Therefore, users do not require to tell the application to start collecting data and when to stop manually. This also helps to reduce unnecessary data collection and transmission.

3.2.5. Database

Every iRoads mobile application has its own Couchbase Lite NoSQL database instance. All sensor data save in that local database and automatically syncing to online Couchbase database when the network is available. Couchbase Lite local databases syncing with online Couchbase database using Couchbase sync gateway. Couchbase supports both cloud database to local databases and local databases to cloud database data syncing. But in iRoads application, it uses only local databases to cloud database syncing functionality.

3.3. Anomaly Detection

Anomaly detection is done inside the backend servers. Before identifying anomalies system performs signal processing step. In order to detect anomalies, the system uses two approaches. The first approach is based on machine learning. The second approach is based on a threshold. After detecting anomalies system performs a clustering process to more accurately identify anomalies.

3.3.1. Signal Processing

Using signal processing techniques System more precisely differentiate road anomalies from normal road segments. As signal processing techniques following techniques are used by the system:

- Average Filtering
- First Derivative
- Second Derivative
- Fourier Transform
- Wavelet Transform

We applied Fast Fourier Transform for vertical acceleration and horizontal acceleration which is perpendicular to the vehicle's moving direction. Then we analyzed the real and imaginary components obtained for each acceleration using graphs.

In Wavelet Transform we went up to four levels of signal differentiation into frequent and infrequent components. Then we analyzed those components using graphs.

3.3.2. Threshold-based Approach

By using a threshold value system can identify anomalies on roads. In order to identify anomalies system searches for spikes larger than a threshold value. After finding locations those have spikes greater than the threshold value, the system removes duplicate locations identified as anomaly locations. Then these GPS locations are transferred to the clustering process.

3.3.3. Machine Learning Approach

Using a classification process system identifies anomalies on roads. After detecting road anomalies, the system identifies GPS coordinates of those anomalies. For the classification process system requires a training dataset classifying each data point collected from the mobile app into an anomaly or a normal road. Classification model Random Forest Classifier [16] is used for the classification process.

3.3.4. Clustering Process

For the clustering process, the system uses GPS coordinates of anomalies identified through the classification process and threshold-based approach. Here system clusters GPS coordinates within a radius of five meters to a single cluster. In the end, the system identifies these cluster centers as the GPS coordinates with road anomalies. The system uses clustering model DBSCAN [17] for clustering. The system considers the number of times that data has collected for a particular road segment as the sample size for the DBSCAN clustering algorithm.

3.4. IRI Estimation

Currently, the Civil Engineering Department is using specified equipments installed into a vehicle to collect IRI values. Through our application, IRI values are predicted from the acceleration data and GPS data collected only using a smartphone application.

3.4.1. International Roughness Index (IRI)

IRI indicates how much vertical movement would be experienced by a standard passenger vehicle body if driven over a 1 mile of a road segment at a speed of 50 mph [18]. Hence, higher IRI values indicate rougher road surfaces.

3.4.2. Pulse counting

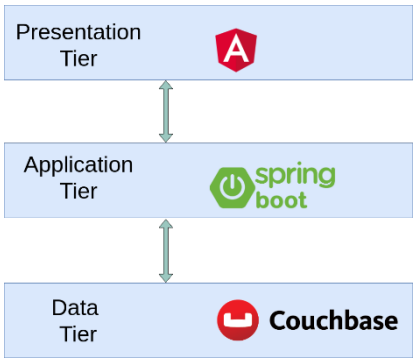
When considering the IRI calculation in miniRomdas it mainly depends on bump integrator raw roughness value. This raw roughness value is a pulse count which indicates movement of the vehicle body with respect to the vehicle axle. System's pulse counting algorithm is a very simple one. If vertical acceleration value is greater than a threshold value that data point considered as a pulse(spike).

3.4.3. Machine Learning Approach

Data collected from both iRoads mobile application and ROMDAS equipments are used to create a dataset that is used to train a prediction model. Random Forest Regressor [19] from the sklearn library is used as the model used to predict IRI values.

3.5. Web Application

The web application is designed with 3 tiered architecture to independently provide an API service in a way that even a 3rd party can access iRoads service.



API service is responsible for providing processed data to frontend API and providing service to other parties. This includes the following functions.

Figure 3.4. Three-tiered Architecture of the web application.

- Getting journey names and ids
- Getting journey data according to journey id (json or csv)
- Getting acceleration graph of a journey as segments divided by a time period
- Getting GPS path of a journey by id
- Getting anomaly tags
- Uploading predicted anomalies json
- Getting predicted anomalies
- Getting journey as segments including
    - Segment average speed
    - Segment average RMS acceleration
    - Segments with
    - Standard deviation of vertical acceleration in the segment

4. Implementation

4.1. Mobile application

The main source for data gathering is the mobile application in iRoads. Here we have developed an android application for this purpose. Following subsections will illustrate how the mobile application is implemented.

4.1.1. Signal Processing

Using signal processing inside the mobile, the system eliminates the noise generated by the smartphone and vibration due to the vehicle engine. Here system uses a simple moving average filter on raw accelerations. Also, the system calculates average constant noise caused by the vehicle engine using a simple moving average filter. As the final step system reduces this constant noise from the filtered accelerations to obtain more refined values.

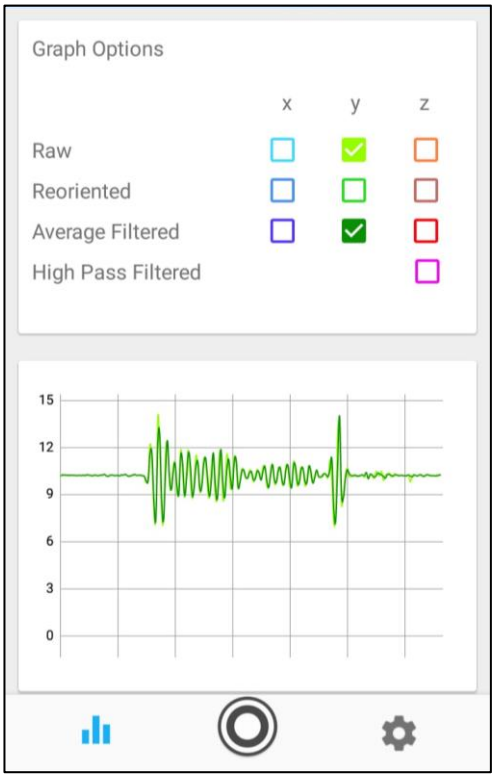


Figure 4.1. Average filtered y-axis accelerometer.

We maintain a queue inside the mobile application to implement the average filter. This is a fixed size queue. Hence, when each new accelerometer reading joins the queue, the oldest accelerometer reading is automatically removed from the queue. The value given from the average filter is the mean of the sensor readings in the queue. When considering the queue size it can be adjusted. But as the default size, we have given five which is a good value to remove sensor noise as well as provide sufficient sensitivity to identify that phone has changed its position. For each sensor, the system maintains separate average filters.

To remove constant noise system maintains a separate queue for each average filter. This queue takes sensor data only when the vehicle is stable. For this queue system takes only reoriented accelerometer readings. The mean value of this queue is reduced from the mean value obtained from the normal queue to obtain average filtered constant noise removed accelerometer readings.

4.1.2. Reorientation

The system is capable of reorienting acceleration along any axis into a stable position. During the reorientation process system first applies a signal filtering process in order to remove the noise. As the next step system performs the reorientation using one of the methods Wolverine or Nericell. Then again system performs a signal filtering process to remove constant noise caused by the vehicle engine.

4.1.2.1. Wolverine

In the reorientation technique based on Wolverine the system uses the rotation vector generated by the accelerometer. This rotation vector is generated using magnetometer readings. Using the rotation vector, the system converts any acceleration vector from any arbitrary position to geometrical coordinates. Then the system uses the GPS bearing of the vehicle to convert these accelerations to vehicular coordinates.

In Android we can obtain rotation matrix from following code:

```
float[] rotation = new float[9];
float[] inclination = new float[9];
float[] gravity = {xValueA,  yValueA,  zValueA};
//here  xValueA means accelerometer x axis reading
float[] geomagnetic = {xValueM, yValueM, zValueM};
// here xValueM means magnetometer x axis reading
SensorManager.getRotationMatrix(rotation, inclination,
gravity,  geomagnetic);
```

With the above code rotation matrix will be generated accordingly to the orientation of the smartphone.

Using following code we can convert accelerometer readings into geometrical coordinates:

```
float geometryAx = rotation[0]*gravity[0] +
rotation[1]*gravity[1] + rotation[2]*gravity[2];
float geometryAy = rotation[3]*gravity[0] +
rotation[4]*gravity[1] + rotation[5]*gravity[2];
float geometryAz = rotation[6]*gravity[0] +
rotation[7]*gravity[1] + rotation[8]*gravity[2];
```

Generated rotation matrix will be used to transform acceleration vectors into geometric axes with the above code. These vectors will be used to generate correct predefined positioned vectors showing in the Figure 2.1 which will be the vehicular coordinates with the below codes.

Using following code we can find magnetic inclination and GPS bearing of the vehicle:

```
GeomagneticField geomagneticField = new
GeomagneticField(latitude, longitude, altitude,timeMilis);
float magneticDeclination = geomagneticField.getDeclination();
float bearing  = previousLocation.bearingTo(location);
float teta = bearing - magneticDeclination;
```

Using following code we can convert accelerations from geometrical coordinates to vehicular coordinates:

```
double ay = geometryAy * Math.cos(teta) - geometryAx *
Math.sin(teta);
double ax = geometryAy * Math.sin(teta) + geometryAx *
Math.cos(teta);
double az = geometryAz;
```

These ay, ax, az are the reoriented vectors that is based on vehicles coordinates.

Figure 4.2 illustrates how Wolverine mechanism reorients accelerations into a stable position.

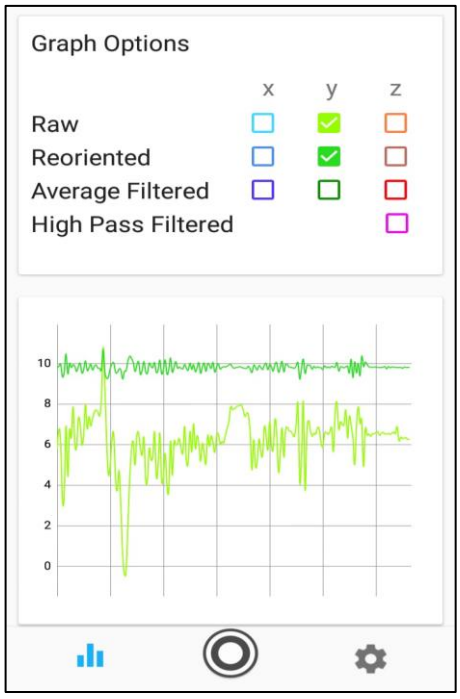


Figure 4.2. Reoriented y-axis acceleration with Wolverine.

The actual implementation of Wolverine mechanism can be found at [20].

4.1.2.2. Nericell

Through this reorientation mechanism, the system uses only acceleration vectors of the mobile. Using the acceleration vectors, the system calculates Euler angles for the mobile's orientation. Then with these Euler angles, the system converts the acceleration vectors into the stable position of the mobile.

In this implementation first, we need to calculate 'teta' and 'pie' angles when a vehicle is not moving. Using the following code we can calculate 'teta' and 'pie' angles:

```
double teta = Math.acos(y / 9.800);
double pie = Math.atan(z/x);
```

Using 'teta' and 'pie' angles we can reorient accelerations along x, y, z axes. Using following code we can obtain reoriented acceleration along x axis:

```
double xPie = x*Math.cos(pie) - z*Math.sin(pie);
double yPie = y;
double zPie = x*Math.sin(pie) + z*Math.cos(pie);
double xTeta = xPie*Math.cos(teta) + yPie*Math.sin(teta);
double zTeta = zPie;
double alpha = Math.atan(xPie/zPie);
double xReoriented = xTeta*Math.cos(alpha) -
zTeta*Math.sin(alpha);
```

Using following code we can obtain reoriented acceleration along y axis:

```
double xPie = x*Math.cos(pie) - z*Math.sin(pie);
double yPie = y;
double yReoriented = -xPie*Math.sin(teta) +
yPie*Math.cos(teta);
```

Using following code we can obtain reoriented acceleration along z axis:

```
double xPie = x*Math.cos(pie) - z*Math.sin(pie);
double yPie = y;
double zPie = x*Math.sin(pie) + z*Math.cos(pie);
double xTeta = xPie*Math.cos(teta) + yPie*Math.sin(teta);
double zTeta = zPie;
double alpha = Math.atan(xPie/zPie);
double zReoriented = xTeta*Math.sin(alpha) +
zTeta*Math.cos(alpha);
```

Figure 4.3 illustrates how Nericell mechanism reorients accelerations into a stable position. Implementation of Nericell mechanism can be found at [20].

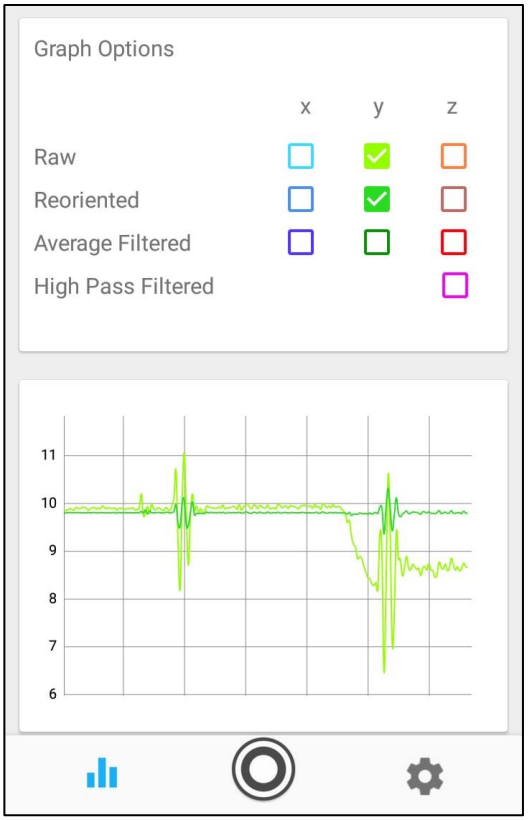


Figure 4.3. reoriented y-axis acceleration (Nericell).

4.1.3. GPS Speed Calculation

The system calculates vehicle speed using GPS to use in scenarios where the vehicle doesn't support OBD2. The most popular way to calculate GPS-based speed is to use the Haversine formula [15]. Haversine formula provides distance between two GPS coordinates along the earth's surface. By keep tracking the time between two position changes it is possible to calculate the speed of an object. Hence, we can directly apply that mechanism to our solution to calculate vehicle speed. Implementation of GPS based speed calculation using Haversine formula can be found at [20].

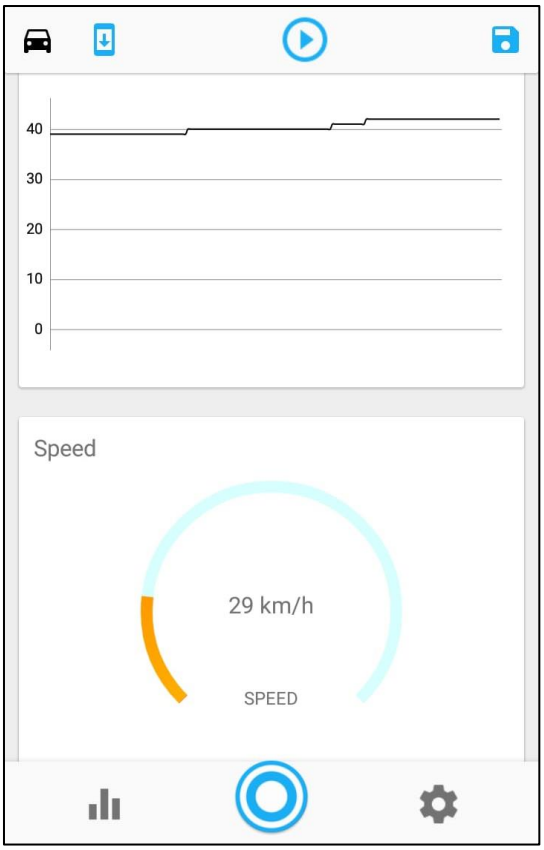


Figure 4.4. GPS Speed.

4.1.4. Mobile app UI

Following user interfaces are designed to achieve more user-friendliness when using the mobile application. Almost all the functionalities are automated to provide the best user experience.

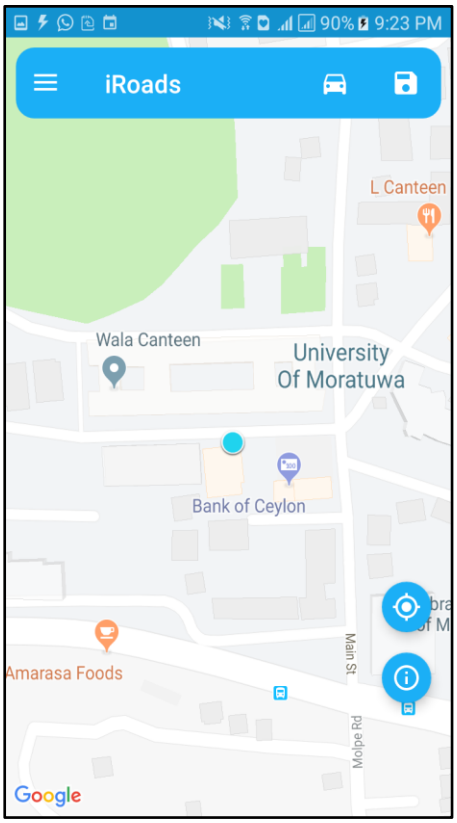


Figure 4.5. Home Screen.

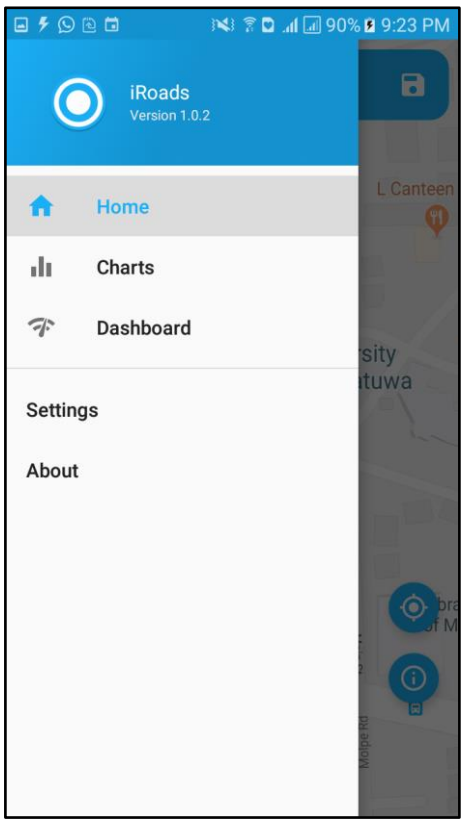


Figure 4.6. Navigation drawer.

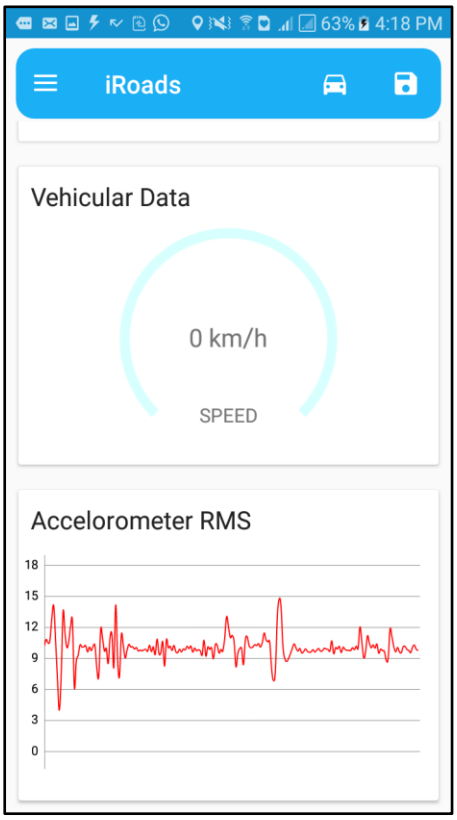


Figure 4.7. Dashboard.

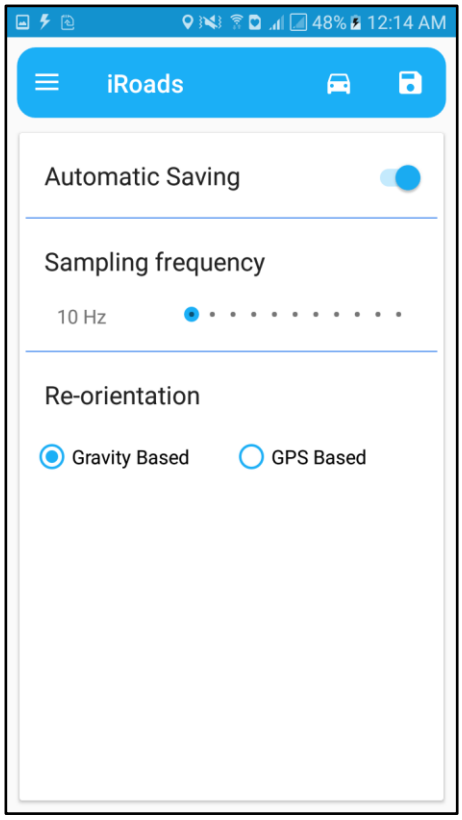


Figure 4.8. Settings.

4.2. Database

4.2.1. Local Database

In applications such as iRoads data is doing a main role. So that data collecting and saving those data into a database is a vital task. When considering the local data saving in Android applications there are several approaches we can take.

Android SharedPreferences [20] can save data as key-value pairs. But this method suitable only for a small amount of data. Without using third-party libraries, the application can save data as .txt, .json files in internal or external storage. But such a method wants more developments in data syncing with an online database. Because of this row data saving method has not any database schema there are difficulties in accessing data again inside the mobile application. With built-in support for SQLite [21] databases in Android, SQLite gives service of data saving and accessing for structured data. Then there are several NoSQL database providers for android. iRoads application collects unstructured sensor data so NoSQL databases are the most suitable databases to save such data. All these NoSQL database schemes provide the functionality of document-based data saving. Noodle [22] and Paper [23] database projects give such NoSQL database schemas. Firebase Database [24] provide the functionality of automatic data syncing across other instances. Also, gives offline data saving functionality with saving data in phone storage. iRoads application doesn't need the functionality of data syncing across multiple devices.

CouchBase Lite [25] databases give document-based data storage, online-offline data syncing with cloud database. This data syncing can be done both one way or two way. But Iroads application only wants one-way data syncing functionality. So when considering the NoSQL database scheme for Android, Couchbase Lite gives efficient and easy to implement service than other solutions. This system uses CouchBase databases for data storage. Offline first behavior and automatic syncing with the online database are the main reasons to choose Couchbase as the database in this project.

4.2.2. Backend Database

Couchbase project gives online Couchbase database server. It communicates with local Couchbase database instances using Couchbase sync gateway. Couchbase sync gateway is also deployed on the same server. The following figure shows a data sample stored in the cloud database. All the data objects created in the local database and synced to the online database.


```
{
"acceX": 1.77635683940025e-16,
"acceX_raw": "8.607155",
"acceY": 9.558568693324924,
"acceY_raw": "0.009576807",
"acceZ": -0.3475502018437826,
"acceZ_raw": "-4.6399627",
"dataType": "data_item",
"gpsSpeed": 4998.357599877832,
"gyroX": "0.10567969",
"gyroY": "0.103847094",
"gyroZ": "0.029321533",
"imei": "",
"journeyID": "1539258159952",
"lat": "6.0834108",
"lon": "80.2551452",
"magnetX": "-7.8599997",
"magnetY": "6.3599997",
"magnetZ": "35.399998",
"obdRpm": "0.0",
"obdSpeed": 0,
"time": 1539259988623
}
```


Figure 4.9. Data item format.

Other than sensor data from mobile application also we store configuration data in the cloud database.

4.3. Anomaly detection

In anomaly detection process we used python as the programming language for implementation. We used Anaconda [26] platform to perform machine learning tasks. Also, we used several python libraries in our anomaly detection process. Implementation of the anomaly detection process can be found at [25].

During the anomaly detection process, we mainly used two python libraries, namely Pandas and Numpy. Pandas [27] library gives the capability of handling data as data

frames. Our backend database consists of data in the form of JSON objects. Hence, using the following code we can convert JSON objects into a pandas dataframe for further processing.

4.3.1. Signal Processing

We used python libraries installed on Anaconda [26] platform for signal processing. Other than two main python libraries mentioned above, we used following two libraries for data visualization, namely Pyplot and Seaborn.

Pandas provides the capability to find moving average for its data frames. Hence, we can use that feature to our average filtering process. Pandas library provides the first derivative functionality for a data frame. Hence, we can directly apply that to our accelerometer signal which is in the form of a data frame. By applying derivative functionality of Pandas on results obtained from the previous step, we can obtain the second derivative. To analyze wavelet transform of acceleration signal we used PyWavelets [31], open source wavelet transform python library. In order to perform Fourier transform we had to use another library. SciPy [32] provides a fast Fourier transform to Pandas data frames. Hence, we used it to find fast Fourier transform of accelerometer signals.

SciPy provides a Fourier transform in the form of complex numbers. We analyzed the real part and imaginary part of the Fourier transformed signal separately. To obtain real and imaginary parts, we used functionalities of Numpy [28] library called Real part and Imaginary part.

4.3.2. Threshold-based Approach

In threshold based approach we checked data points where vertical acceleration greater than ten. Then those GPS coordinates were identified as anomalies. After detecting anomalies we had to remove duplicated GPS coordinates. Then we removed duplicated GPS coordinates.

4.3.3. Machine Learning Approach

During the machine learning process, we used the scikit-learn [33] library. To do oversampling we used SMOTE [34]. We used Random Forest Classifier [16] as our classification model. Reoriented accelerometer readings, GPS speed and anomaly type was used as features to train the model. We used Grid Search [35] to fine tune the parameters of the Random Forest Classifier [16] model. Using the tuned parameters we created a model to predict anomalies as follows:

4.3.3.1. Parameter Tuning

```
forestModel = RandomForestClassifier(bootstrap=True,
class_weight=None, criterion='gini' max_depth=1,
max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0,min_impurity_split=None,min_samples_
leaf=0.01, min_samples_split=2, min_weight_fraction_leaf=0.0,
n_estimators=1,
n_jobs=1,oob_score=False,random_state=None,verbose=0,warm_star
t=False)
```

As the next step we trained the model. As final step we predicted anomalies for new datasets. In this approach also, we need to remove duplicate GPS coordinates detected as anomalies similarly using an approach like in threshold based approach.

4.3.4. Clustering

In order to cluster anomalies we used DBSCAN [17] clustering. Clustering model needs a sample size and a radius. Here we used radius as five meters. And sample size as number of times that data has collected for a particular road segment. For an example if we collected seven times for Moratuwa to Katubedda segment in Galle road. We performed predictions for all seven datasets. And clustered them to get to a single cluster, if at least seven data points in the predicted datasets are in five meter radius.

```
kms_per_radian = 6371.0088 // Earth radius
epsilon = 0.005 / kms_per_radian // five meter in GPS terms
coords = raw_data.as_matrix(columns=['lat', 'lon'])
db=DBSCAN(eps=epsilon,min_samples=7,algorithm='ball_tree',metr
ic='haversine').fit(np.radians(coords))
```

As final step cluster centers were identified as anomaly locations. The implementation to find cluster centers can be found at [xx].

4.4. IRI estimation

Data collected from both ROMDAS equipment and iRoads mobile application is analyzed and used for training prediction models.

4.4.1. IRI Data Collection from ROMDAS

For the IRI data collection, we used miniRomdas [36] system. MiniRomdas is a streamlined version of the full ROMDAS system. With compared to other IRI calculating options this system is easy to install and cost-effective solution. According to the manufactures, the system is unaffected by wet, unpaved and rough conditions. Even with a minimum operating speed (10km/h). This system consists of a central data logger, odometer and Bump Integrator module as shown in Figure 4.11.

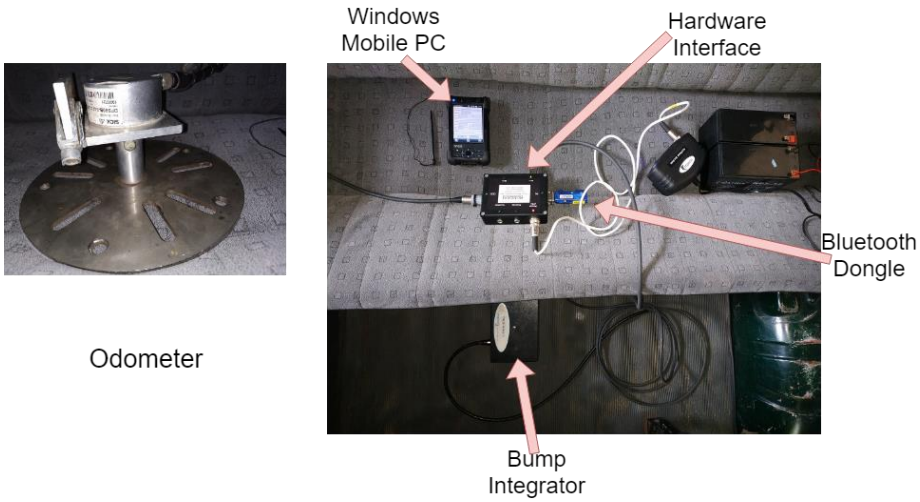


Fig 4.10. miniRomdas components.

The central data logger is a Microsoft Windows mobile pc and it connects to hardware interface using a Bluetooth dongle and take odometer and bump integrator data. At the end of the data gathering, we can collect those data by connecting the mobile PC to a Laptop or a desktop computer. The odometer is fixed to a wheel of the vehicle and it connects to hardware interface using a cable.

Bump integrator is fixed to the vehicle and it connects to the middle of the axle as shown in Figure 4.12. Bump integrator sends data to hardware interface using a cable. After fixing the miniROMDAS system to the vehicle we have to calibrate this system for the vehicle. For the calibration process, we have to collect data for five different roads using this miniROMDAS system. Then for the calibration, we have to find IRI values of those roads and for that, we used Z-250 Reference Profiler [37]

Fig 4.11. Bump Integrator connected to a vehicle axle

.

Z-250 Reference Profiler is a class one IRI measuring instrument that has been developed to measure high accuracy reference profiles. We measured IRI values of 300m road segments using this device and collected miniRomdas data using the vehicle. Using those data we created a calibrated profile for Mitsubishi Delica vehicle in miniRomdas software. Then we collected IRI data for different road segments using

Mitsubishi Delica vehicle. In every ride, we also collected acceleration and GPS data using mounted smartphones. MiniRomdas system outputs collected data as Microsoft Access (.mdb) files. These files contain both raw data (chainage, roughness, time) and calculated data (IRI, vehicle speed).

Figure 4.12. Z-250 Reference Profiler.

Figure 4.13. Data collecting using mounted smartphones.

Table 4.1 shows a sample of miniRomdas output data. **SPEED** column indicated the average speed of the vehicle for the considered road segment. Here data is collected for 100m road segments. **ROUGH_1** is the pulse count received from the bump intregrator. **C_ROUGH_1** is the calculated IRI value for that road segment using the pulse count. Also this table shows the time taken for cover each road segment.

Table 4.1. Sample output data table of miniRomdas system.

CHAINAGE	SPEED	LRP_NUMBER	ROUGH_1	ROUGH_2	TIME	C_ROUGH_1
100.16	27.4244	0	1214	0	13.148	3.95
200.31	36.5919	0	1403	0	23.001	4.17
300.3	40.743	0	1268	0	31.836	4.02
400.19	43.0097	0	943	0	40.197	3.64
500.03	6.03081	0	1715	0	99.795	4.54
600.05	26.8751	0	1282	0	113.193	4.04
700.14	39.1231	0	1098	0	122.403	3.82

4.4.2. Linear Regression approach

Using collected iRoads and ROMDAS data, different features were extracted. As a result of we found a set of features give good correlation with romdas IRI values. Then we calculated best-fit lines using linear regression algorithms and calculated iRoads IRI using those equations.

4.4.3. Machine Learning approach

Considering the results from correlation analysis and threshold selection process, primary features from the dataset were selected to train the Machine Learning model.

According to the above correlation matrix in Figure 47.15, there are only few features with significant relation to the IRI values. They can be listed as follows:

- ‘calSpikesY’ - Calculated spikes from y-axis for a given road segment.
- ‘speed’ - The Actual speed of the vehicle recorded through Odometer.
- ‘gpsSpeed’ - The Calculated speed of the vehicle using GPS coordinates.

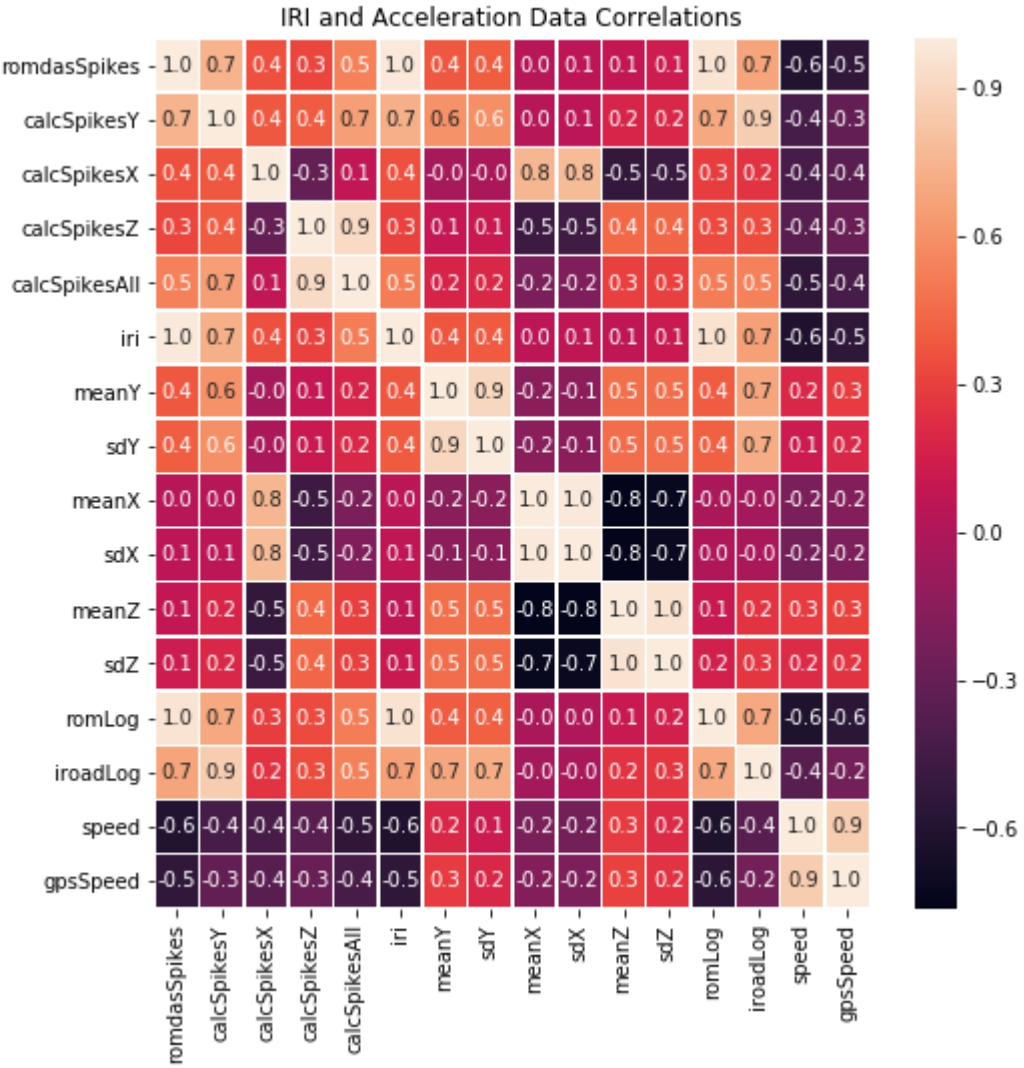


Figure 4.14. Correlation Analysis.

4.5. Data Visualization

Due to the scarcity of reliable and revised data on the condition of road networks, authorities are in lack of insight when implementing road maintenance programs or making road design guidelines. With proper visualization of road condition through a dashboard, these problems can be addressed. Visualization part consists of data accessing from the database through API and visualizing in the frontend. Architecture design of the project is shown in Figure 6.1.

Here the data are managed as journeys. Every time a journey starts when someone starts traveling in a vehicle with iRoads application. Those journeys are saved with journey ID and journey stops when the vehicle stops. This starting stopping behavior is controlled by pathsense [14] android library. Because of having a limited amount of data, visualizations are mostly done according to those journeys at the moment.

4.5.1. API

To access raw data and predictions from the database and processing required visualization data is done by iRoads API service. This spring boot application is running on a tomcat server that has been integrated with it.

Here we have used Spring boot for development of this API. Mainly due to default configuration management and better dependency management in spring boot were the reasons to choose it for backend developments. Database handling is managed with Couchbase data accessing dependencies. API service is providing these services:

- Getting journey names and ids
- Getting journey data according to journey id (json or csv)
- Getting acceleration graph values of a journey as segments divided by a time period(in seconds)
- Getting GPS path of a journey by id
- Getting anomaly tags
- Uploading predicted anomalies json
- Getting predicted anomalies

- Getting journey as segments including
    - Segment average speed
    - Segment average RMS acceleration
    - Segments with
    - Standard deviation of vertical acceleration in the segment

4.5.2. Front-end

The front-end of iRoads has developed using Angular. Angular was used for better division of components and reusing the existing code. This angular application is running on an apache server. In the front-end part of the web application, there are mainly two types of visualizations, namely graph view, and map view

4.5.2.1. Graph view

Graph view has implemented using D3 and ng2-nvd3 libraries. In graph view, it is showing acceleration x,y,z vectors of raw and reoriented data. These are the accelerations gathered while traveling with mobile application running. Accelerations are the main considering data for predictions. So, acceleration graph view is important to tagging data and check moves in anomaly situation. The graph in Figure 4.16 is shown as segments for smooth interaction.

Figure 4.15. Acceleration graphs.

4.5.2.2. Map view

Map view has been developed using OpenStreetMap [38]. In map view following details are shown. Routes are shown according to journey names. Names are automatically assigned by considering starting GPS location with the help of OpenStreetMap API. To visualize predictions there is an option to check tagged anomalies. The figure below shows the tagged anomalies in a given route.

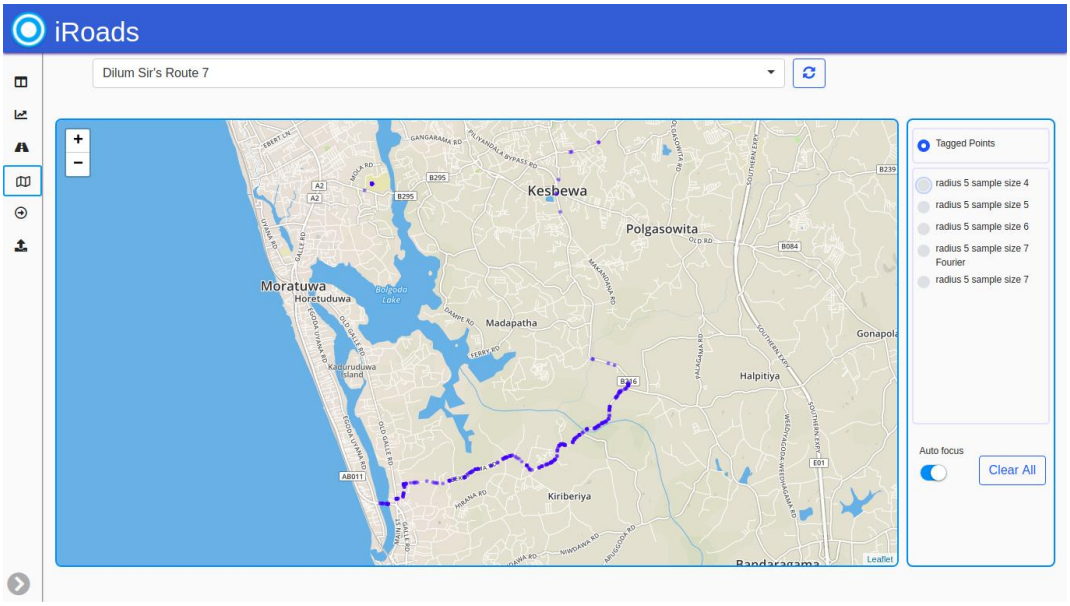


Figure 4.16. Tagged anomalies view.

Predicted anomalies are shown in Figure 4.18 by considering a given route. These GPS locations were clustered to find one location from several journeys that we collected data on the same route.

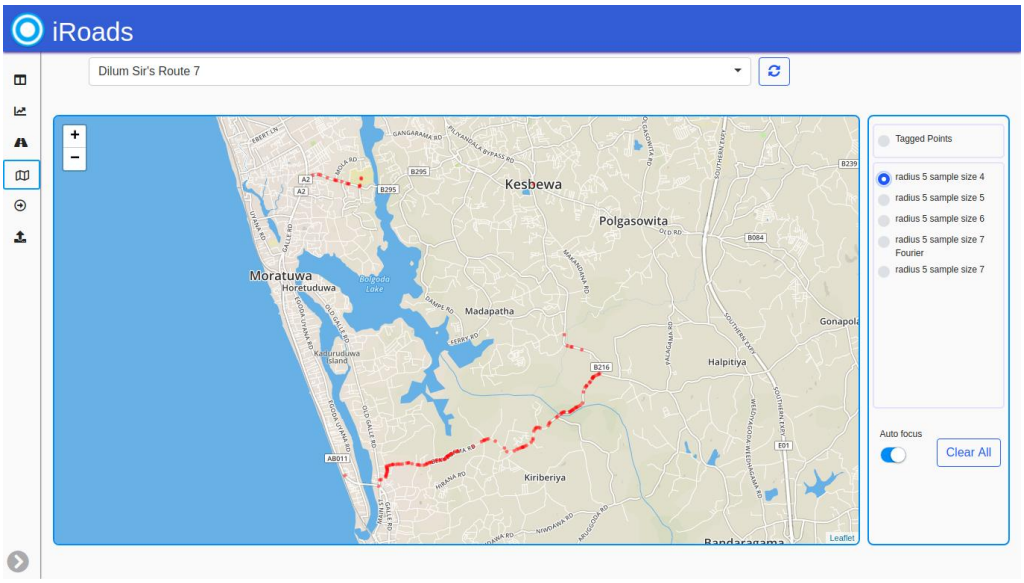


Figure 4.17. Predicted anomalies view.

Figure 4.19 shows the tagged anomalies and predicted anomalies on one map. This can be used to measure the accuracy of the prediction models.

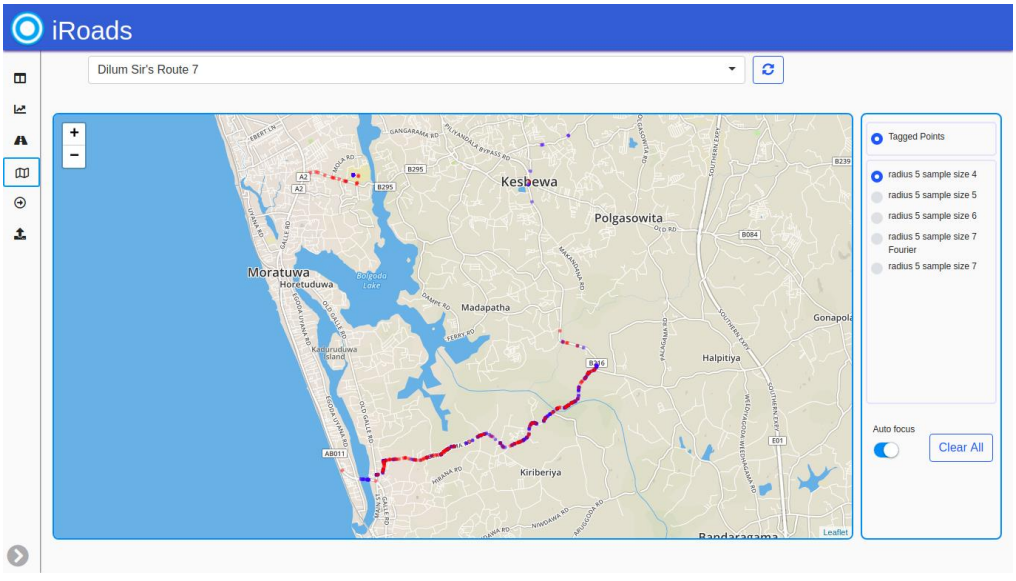


Figure 4.18. Tagged anomalies with Predicted anomalies.

Currently, IRI calculation is being done by calibrating with ROMDAS values. Average vertical movement (Acceleration Y) is shown by color codes in the below diagram.

Average acceleration RMS value is shown in the Figure 4.20. These are shown in segments that were divided by 100m.

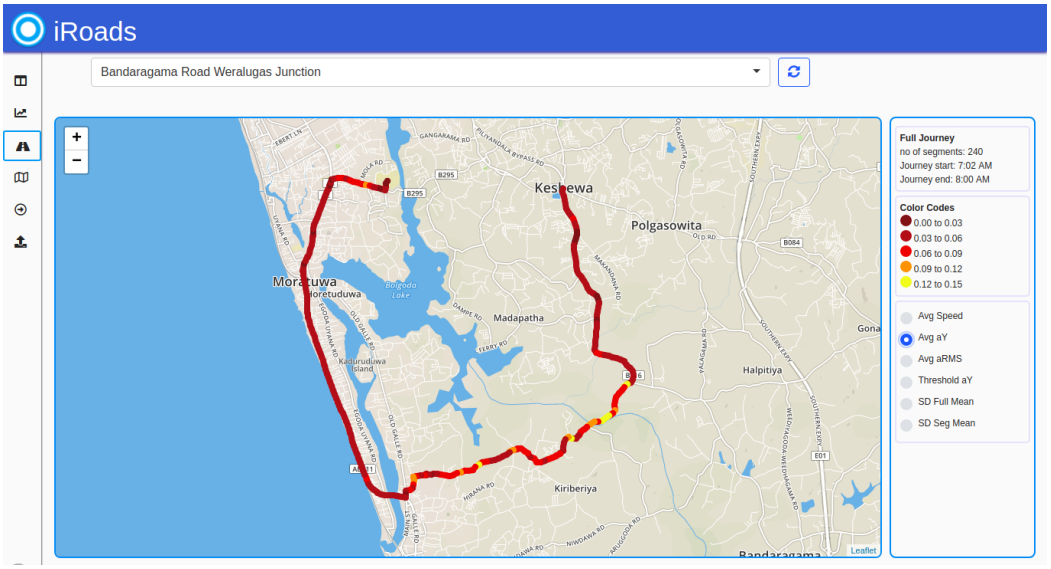


Figure 4.19. Average vertical acceleration of a journey.

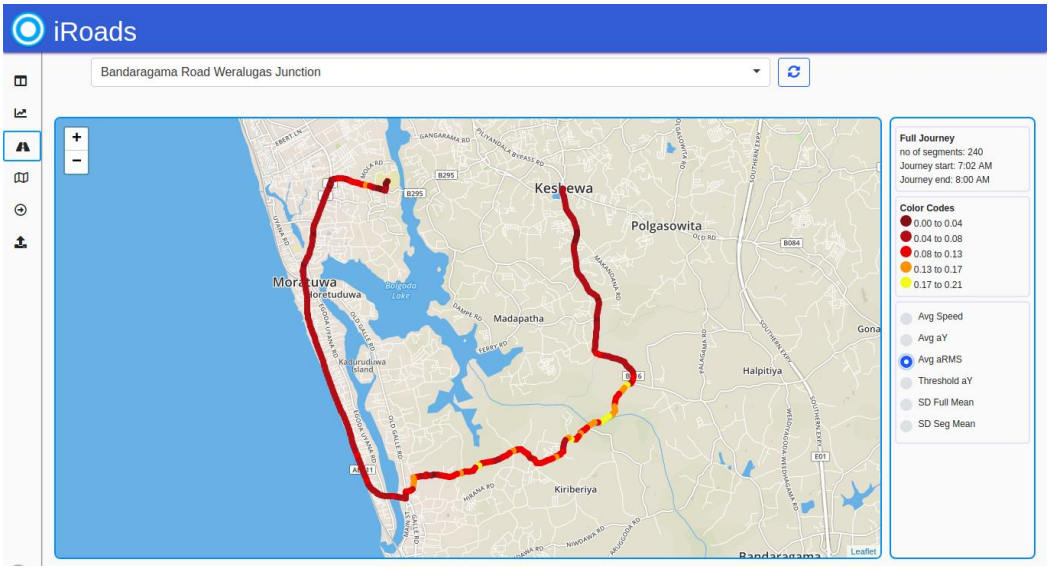


Figure 4.20. Average x,y,z acceleration RMS of a journey.

The standard deviation of the vertical acceleration is shown by getting the total mean value to calculate the standard deviation here. Figure 4.22 shows the standard deviation of the segments by getting mean of those specific segment.

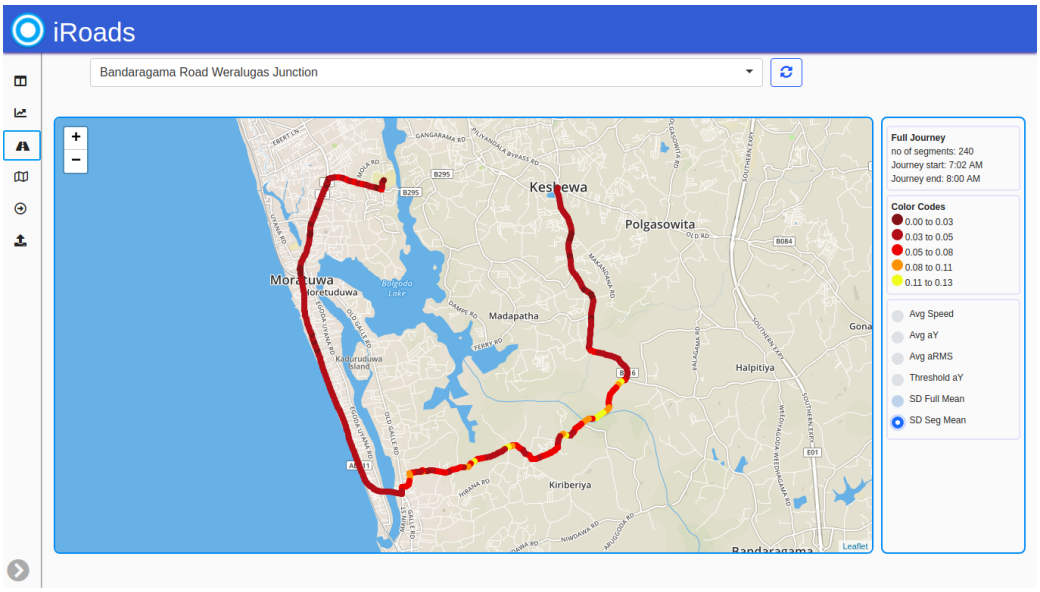


Figure 4.21: Standard deviation of vertical acceleration of a journey with segment mean.

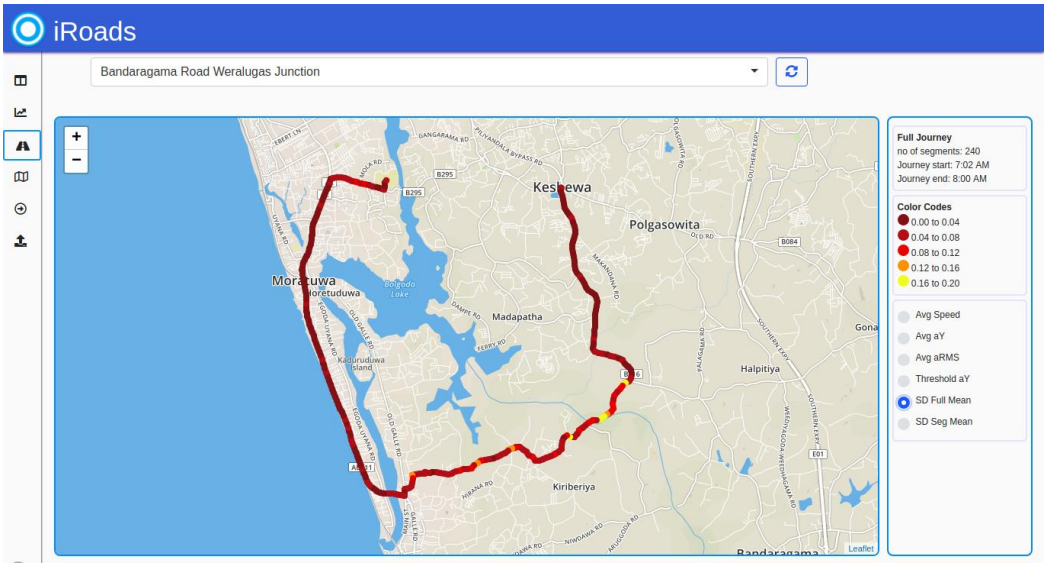


Figure 4.22. The standard deviation of a journey with the full mean.

The average speed of the vehicle in the journey shown in Figure 4.24 as iRoads showing in the web.

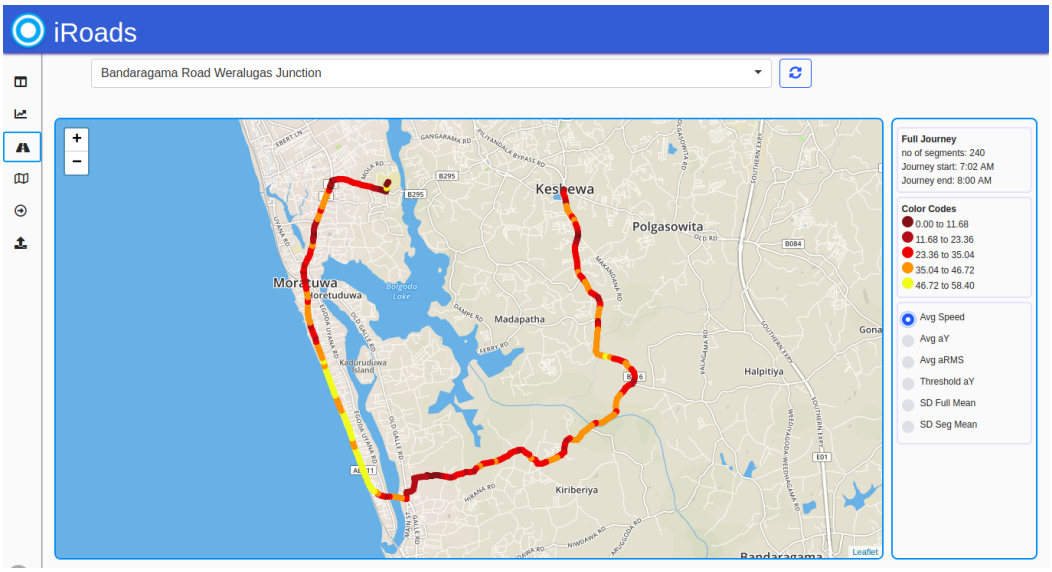


Figure 4.23. The average speed of a journey.

4.5.3. Summary

Visualizing the data processed to get useful information is an important task in this project. Mainly for RDA, it is a requirement to check the routes with their current situation and continuously monitor them. By providing a web-based solution, we can achieve this requirement without access limitations. From any smartphone, tablet or PC anyone can check this information with the responsive user-friendly interface developed by us.

4.5.4. Contribution in implementation

While everyone has contribute to solve problems in each part by discussing U.M.J Abeywikrama was mainly doing IRI correlation analysis and IRI linear regression analysis. P. T. Amarasinghe was mainly doing anomaly detection and signal processing. H.M.A Abeywardana was mainly maintaining the mobile application and developing IRI machine learning approach. Dushan was mainly doing frontend, backend developments, and road segment feature calculating. Contributions mentioned in the following table are not limited only to the mentioned parts but these were the rough division of tasks to members.

Table 4.2. Individual Contribution.

	H.M.A Abeywardana	**U.M.J Abeywikrama**	**P. T. Amarasinghe**	**R. P. D. Kumarasinghe**
Mobile application	**55%**	15%	15%	15%
IRI Data gathering with ROMDAS	20%	**40%**	20%	20%
Anomaly detection	10%	10%	**70%**	10%
Anomaly Data Collection	25%	25%	25%	25%
Feature analysis	10%	**70%**	10%	10%
IRI prediction	**40%**	**40%**	10%	10%
Road segment creation and feature calculating	10%	10%	10%	**70%**
Backend and frontend development	10%	10%	10%	**70%**
Signal processing	10%	20%	**60%**	10%
Report writing and literature finding	25%	25%	25%	25%

5. Performance analysis

5.1. Mobile Application

We use a mobile app as the data collection mechanism in our solution. We collect accelerometer readings, magnetometer readings and GPS readings from the mobile phone and send them to the backend servers for further processing. Our mobile app is capable of detecting whether the mobile is in a moving vehicle and it only collects data relevant to scenarios where the mobile is in a moving vehicle. Hence, the user only needs to turn on our mobile app and do his or her normal driving routing. Therefore, our mobile app is pretty much user-friendly. Currently, this mobile app is available in the Google play store.

Our mobile app has a local database and it automatically syncs with the remote database. Therefore, the user does not need to connect to the internet while our mobile app is turned on. Hence, this data syncing mechanism has increased the user-friendliness of our mobile app.

5.1.1. Signal Processing

In order to remove mobile phone sensor noise, we used several signal processing mechanisms inside the mobile app. We have implemented a simple average filter to smoothen the accelerometer reading and a constant noise removal mechanism after the reorientation process. With the use of these signal processing mechanisms, we have been able to send a more stable accelerometer reading for further processing. Therefore, we were able to completely remove smartphone accelerometer noise. Also, when we consider the noise caused by the vehicle, we were able to reduce it to some extent by using this signal processing technique. But we cannot remove the larger noise caused by the vehicle's engine when the vehicle is moving compared to noise caused by the engine at stable time from this technique.

5.1.2. Reorientation Mechanisms

Since our solution is a crowd-sourced solution, our system should be capable of collecting acceleration data from any arbitrary position. Therefore, we needed a reorientation mechanism to reorient disorient acceleration vectors. We have implemented two mechanisms for reorientation. Both mechanisms have strengths and weaknesses. But both mechanisms gave excellent results to identify spikes caused by an anomaly with adequate sensitivity.

When we compare and contrast two reorientation mechanisms. We observed that Wolverine mechanism showed more sensitivity compared to Nericell mechanism. Since Wolverine mechanism uses GPS to find vehicle moving direction there is an error when the vehicle travels in a road which has bends. This is the main drawback of Wolverine mechanism. But this issue is not a major concern because most of the time we consider vertical acceleration and it is not affected by this error. Also, Wolverine mechanism cannot be applied in every mobile phone because some mobile phones do not exist magnetometers. Nericell has overcome above drawbacks. But it's gravity-based Euler angle calculation cannot be done when the vehicle is moving. Therefore, we need to assume that the mobile phone position is unchanged when the vehicle is moving. But we have tested this issue and we observed that there is no huge variation in reoriented values. Therefore, considering these issues we recommend drivers to use a phone holder to collect data when they use Nericell as reorientation mechanism.

5.1.3. GPS Speed Calculation

Another important process we conduct inside the mobile app is that we calculate speed using GPS coordinates. In anomaly detection and IRI, calculation speed plays a vital role. Hence, we need to send current speed of the vehicle with other sensor data to backend servers.

Due to errors in GPS readings speed calculated using GPS is bit inaccurate. But we observed that between the range 30kmp/h to 60kmp/h GPS based speed is similar to

actual vehicle speed. Since we have OBD2 based speed as an option as well, we can use GPS based speed in vehicles that do not support OBD2.

5.1.4. Summary

The mobile app is fully developed and currently, it is in the Google play store. We have collected data using this mobile app several times and it is properly syncing with backend servers. Signal processing mechanisms and reorientation mechanisms provide processed sensor data with adequate sensitivity to identify road anomalies and calculate IRI values. Due to automatic syncing option, no data is lost and the user does not need to always connect to the internet. Hence, there is an efficient power usage as well. The mobile app has developed to achieve maximum user-friendliness. Also, due to the automatic motion detection capability of the mobile app user only needs to turn on the mobile app when he or she is driving. The mobile app can be run as a background app. Hence, the user has no burden of always keeping the mobile app on the screen.

5.2. Anomaly Detection

Anomaly detection is done inside the backend servers. We have tried several signal processing approaches to identify the difference between a pothole and a bump. We have tried average filtering, first derivative, second derivative, Fourier transform, and Wavelet transform.

After performing a signal processing step we performed anomaly detection. In order to detect anomalies, we used two approaches. The first approach is based on machine learning. In the first approach, we train a classification model to identify anomalies. The second approach is based on a threshold. In the second approach, we use a threshold to identify spikes in vertical acceleration. We considered these spikes as anomalies on the road.

After detecting anomalies we perform a clustering process. Since our solution is a crowd-sourced one same anomaly detected by multiple times. Hence, by performing a clustering step improved the accuracy of predicted anomaly locations.

5.2.1. Signal Processing Application

Using signal processing techniques we tried to differentiate road anomalies from normal road segments. Also, we tried to differentiate potholes from bumps using the observations we got from signal processing. We considered the vertical acceleration signal and horizontal acceleration signal which is perpendicular to the vehicle's moving direction as our main data sources. We did not use accelerations along the vehicle's moving direction because those accelerations were affected by braking and accelerating actions done by the driver. Followings are the signals we got for a pothole and a bump.

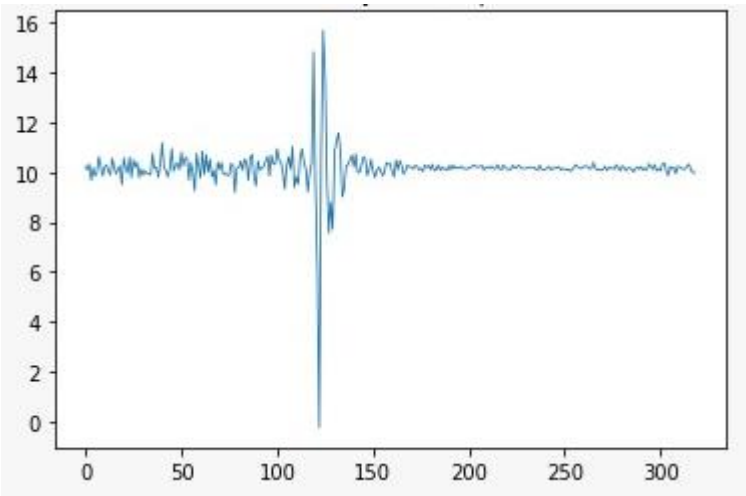


Figure 5.1. Raw y-axis acceleration for the bump.

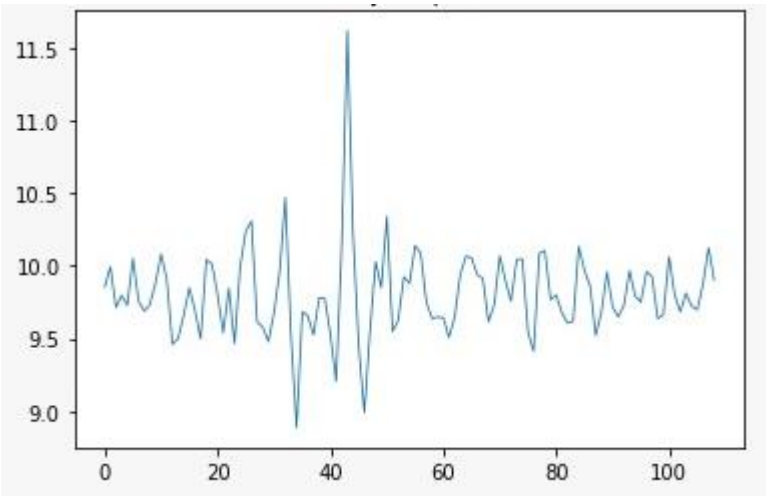


Figure 5.2. Raw y-axis acceleration for the pothole.

As we can observe there is no difference between the two vertical acceleration signals. But we can clearly identify an anomaly from a normal road segment. Hence, we tried different signal processing techniques to differentiate potholes from bumps. Also, we tried to check whether there is a relationship between horizontal accelerations and potholes. Followings are the signals we observed for bumps and potholes along the horizontal axis which is perpendicular to vehicle's moving direction.

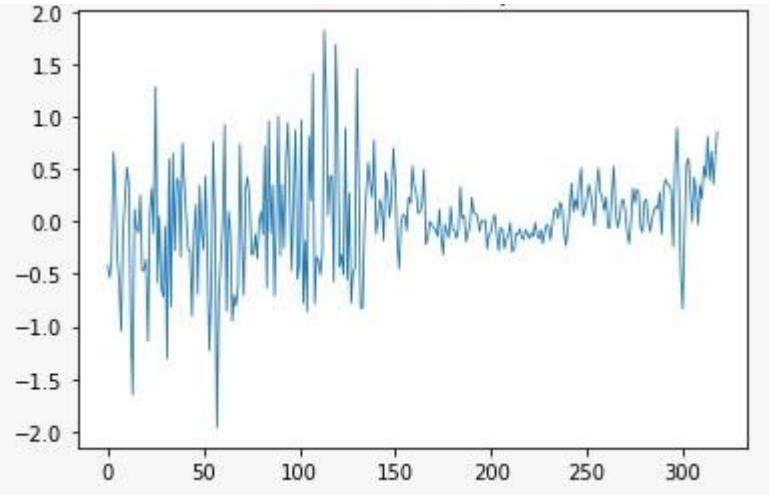


Figure 5.3. Raw x-axis acceleration for the bump.

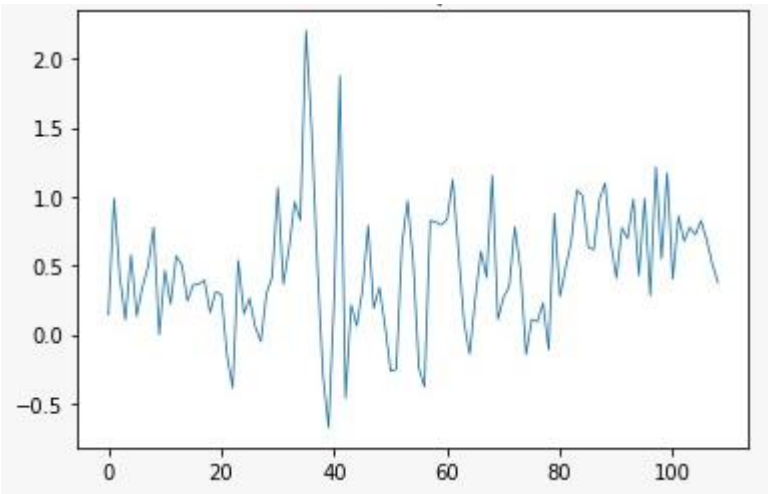


Figure 5.4. Raw x-axis acceleration for the pothole.

As we can see there is no clear evidence to differentiate potholes from bumps in accelerations along the horizontal axis which is perpendicular to vehicle's moving direction.

5.2.1.1. Average filtering

We tried a simple moving average to the raw signal to check whether we can differentiate a pothole form a bump. Followings are the signals we obtained after applying a moving average filter.

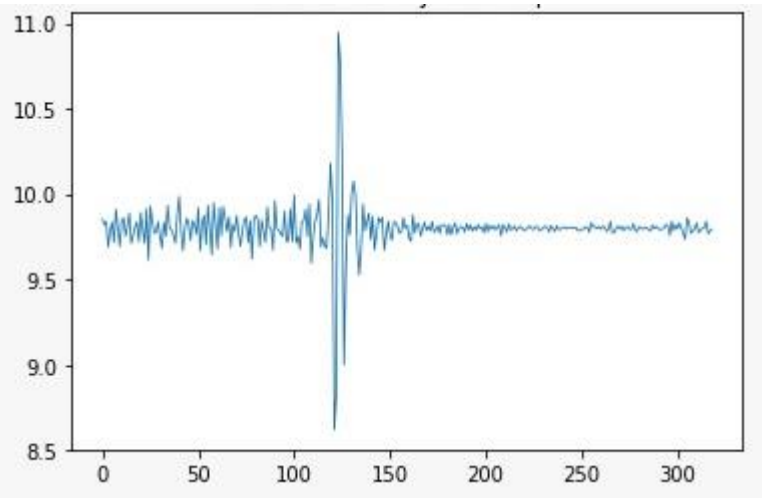


Figure 5.5. Reoriented y-axis acceleration for the bump.

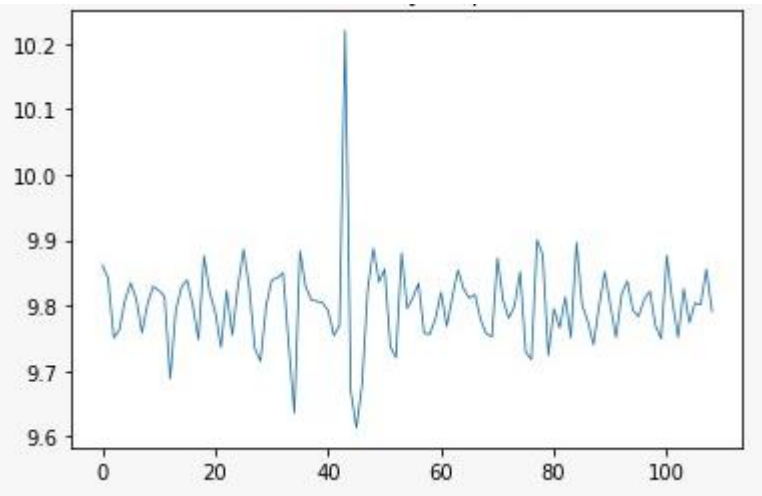


Figure 5.6. Reoriented y-axis acceleration for the pothole.

As we can see that there is no clear difference in vertical acceleration between a pothole and a bump. This is similar to the acceleration along the horizontal axis which is perpendicular to vehicle's moving direction. But we have identified that by applying a moving average filter helps to easily identify anomalies from vertical accelerations.

5.2.1.2. First derivative

With the assumption that in a pothole vertical acceleration first goes down and then up. And in a bump vertical acceleration first goes up and then down. We tried the first derivative of the vertical acceleration. According to the assumption, we should get a positive first derivative value for a pothole and a negative first derivative value for a bump. Followings are the signals we obtained for the first derivative.

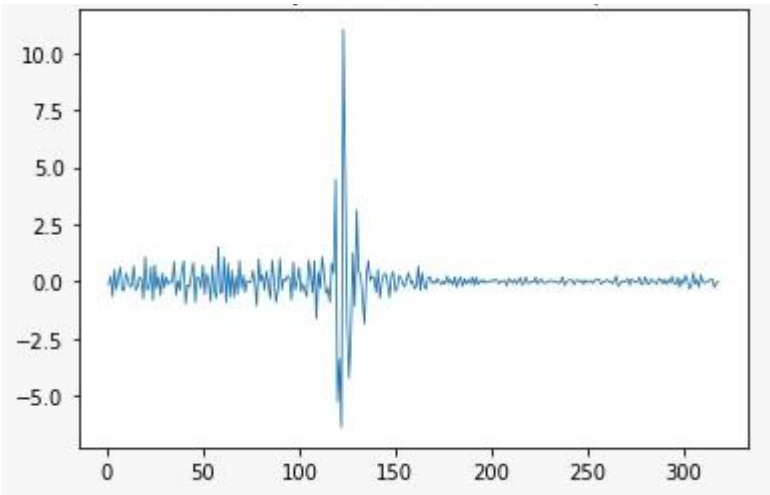


Figure 5.7. The first derivative of raw y-axis acceleration for the bump.

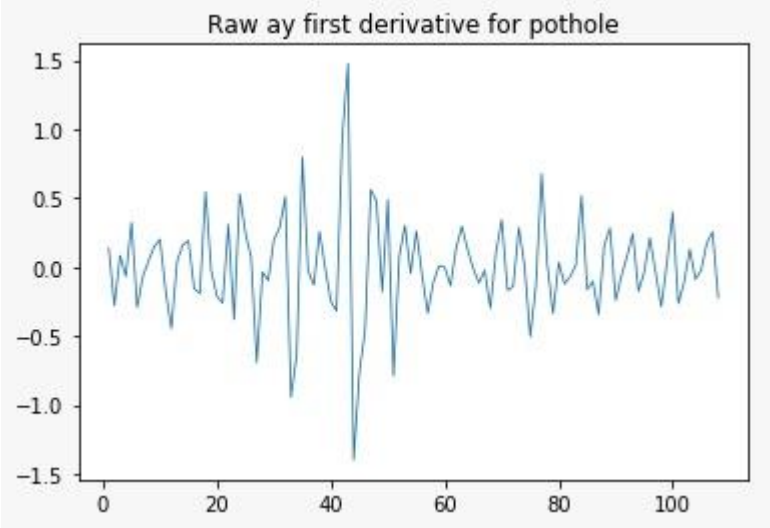


Figure 5.8. The first derivative of raw y-axis acceleration for pothole.

As we can observe there is no clear difference between in the signals for potholes and bumps. This is similar for accelerations along the horizontal axis which is

perpendicular to vehicle's moving direction. Hence, we cannot differentiate potholes and bumps using the first derivative.

5.2.1.3. Second Derivative

We tried the second derivative to check whether there is a clear evidence to differentiate a pothole form a bump. These are the signals we obtained for the second derivative for vertical acceleration.

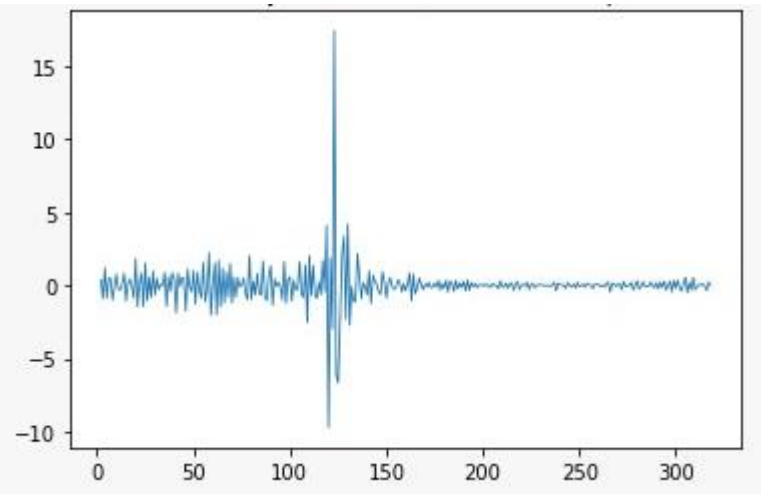


Figure 5.9. The second derivative of raw y-axis acceleration for the bump.

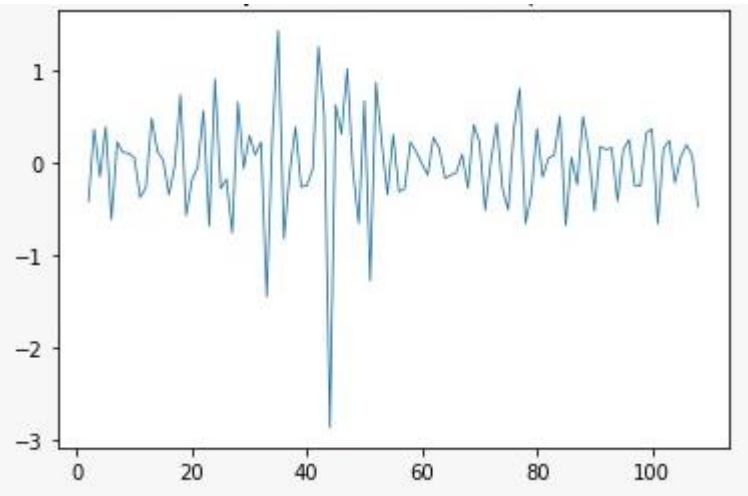


Figure 5.10. The second derivative of raw y-axis acceleration for pothole.

As we can see there is no clear evidence for a pothole or a bump in the second derivative of the vertical acceleration. This is similar for the acceleration which is

perpendicular to the vehicle's moving direction. Hence, there is no direct relationship between the second derivative and pothole bump differentiation.

5.2.1.4. Fourier Transformation

We checked whether we can use Fourier transformation to differentiate potholes and bumps. We applied Fourier transformation for vertical acceleration and horizontal acceleration which is perpendicular to the vehicle's moving direction. Followings are the results we obtained.

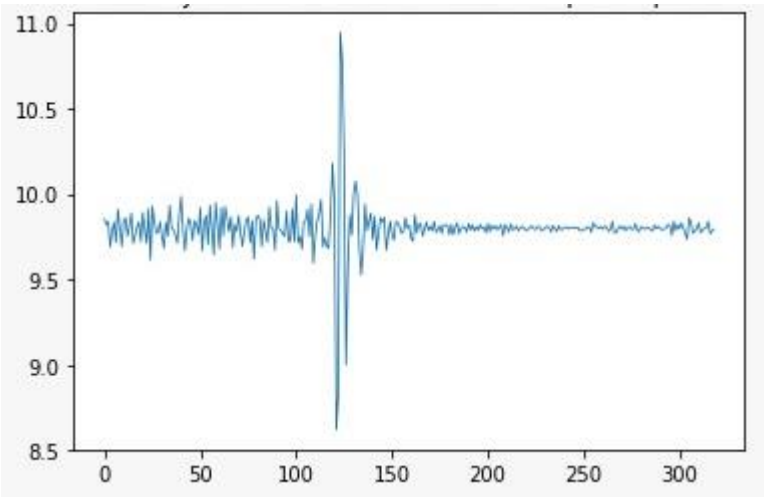


Figure 5.11. Fast Fourier transform of raw y-axis acceleration for the bump (real part).

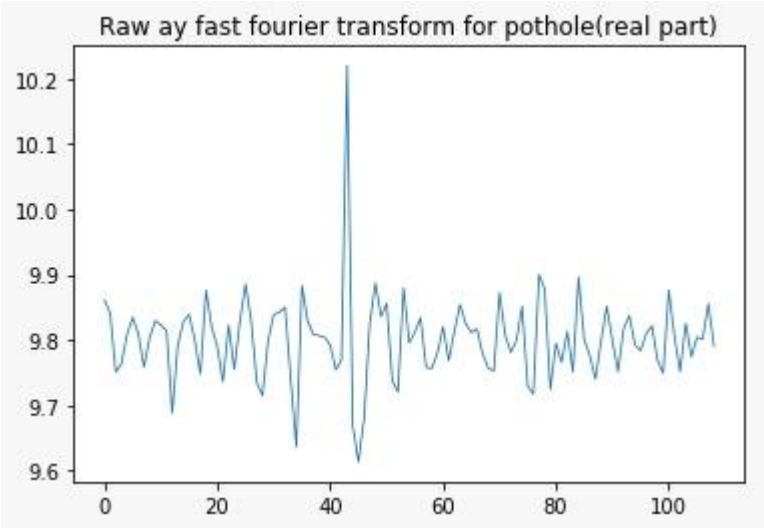


Figure 5.12. Fast Fourier transform of raw y-axis acceleration for pothole (real part).

As we can see there is no real difference between the two real parts of vertical accelerations for pothole and bump. This is similar to imaginary parts as well. Also, we observed that this is similar for the Fourier transformation for the horizontal acceleration which is perpendicular to the vehicle's moving direction. Hence, we decided that we cannot differentiate potholes form bumps using Fourier transformation.

5.2.1.5. Wavelet transformation

We analyzed acceleration y signals of potholes and bumps and wavelet transformed signals of those signals. The following figure shows original and wavelet transformed Y acceleration signals for two bumps.

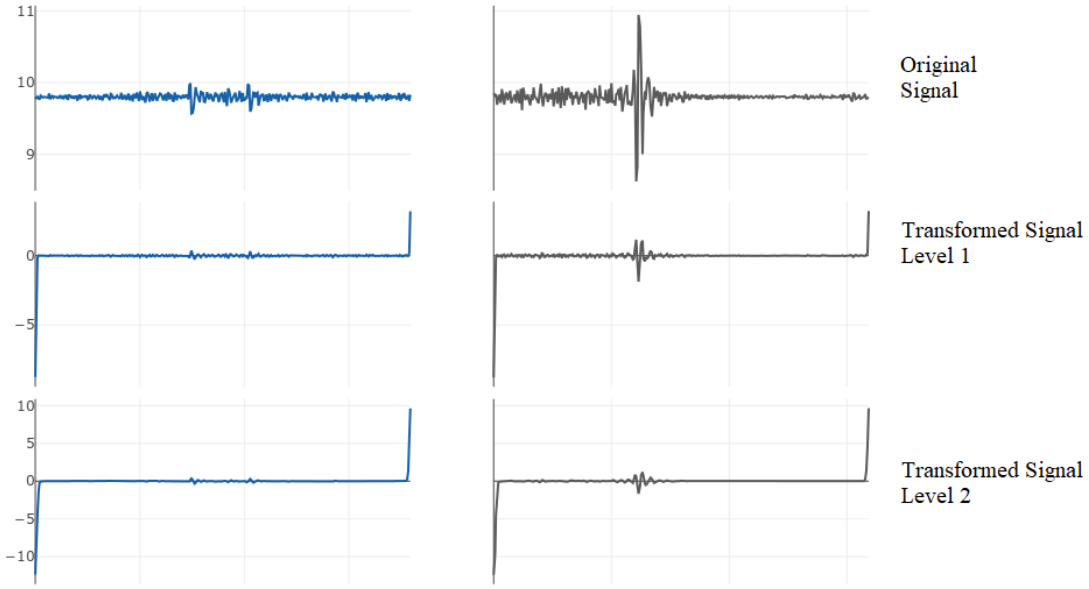


Figure 5.13. Wavelet transformed signals for bumps.

Figure 5.14 shows original and wavelet transformed y-axis acceleration signals for two potholes. So we can see that there is a fixed signal pattern in both original and wavelet transformed signals. So we decided that we cannot use wavelet transformation for classification of potholes and bumps.

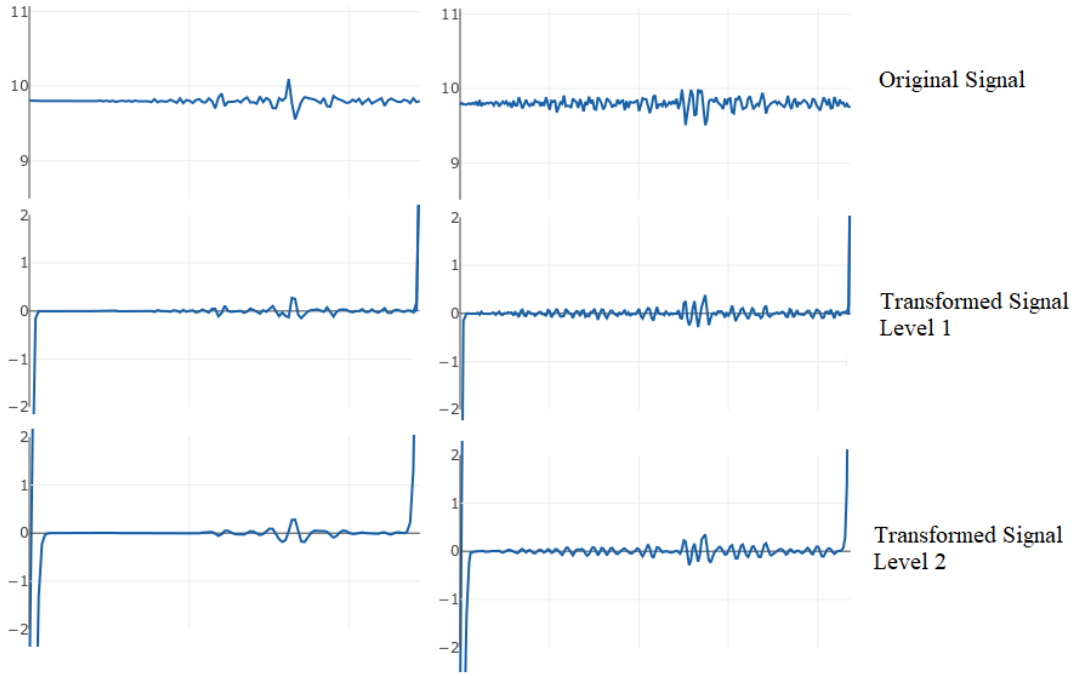


Figure 5.14. Wavelet transformed signals for potholes.

5.2.2. Threshold-based Approach

By using a threshold value, the system can identify anomalies on roads. Here we used average filtered vertical acceleration to identify anomalies. We used 10 as the threshold value and identified GPS coordinates where vertical acceleration is greater than the threshold value. Then we removed duplicated GPS coordinates from the selected anomalies because we need to give a sample size for the clustering model. By removing duplicates we can give the number of times that data have collected for a particular road segment as the sample size for the clustering model. What we have observed from the threshold-based approach is that we cannot find a logic behind deciding the threshold value. Threshold value depends mainly on the vehicle. Therefore the threshold-based approach is not very much suitable for a crowdsourced anomaly detection. The following figure shows the results we obtained after clustering anomalies identified from the threshold-based approach.

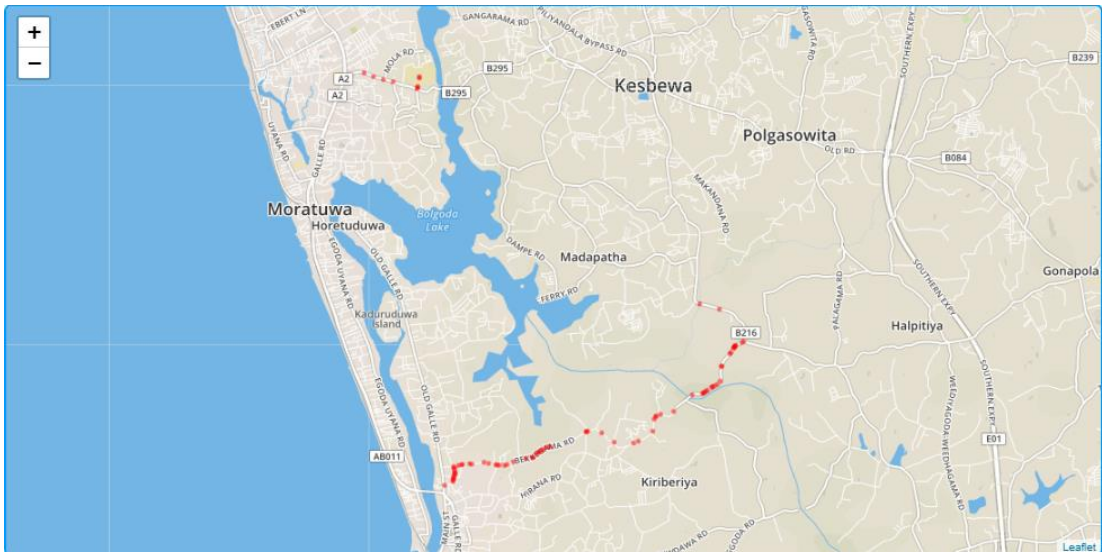


Figure 5.15. Resulted anomalies from clustering

The actual number of anomalies on the road: 109

Table 5.1. Results of Threshold based approach.

Scenario	Count
System correctly predicted an anomaly	70
System incorrectly predicted an anomaly	12
System missed an anomaly	39
Precision	0.854
Recall	0.642

:

5.2.3. Machine Learning Approach

In the machine learning approach, we detected anomalies using a classification process. Classification process was done using a Random Forest Classifier [16]. Here we used vertical acceleration, horizontal acceleration which is perpendicular to vehicle's moving direction and vehicular speed as the features for the machine learning model. Also, we used moving average filter for every feature to smooth them.

When preparing the training dataset. Anomalies were tagged manually while other data points were tagged as normal data points. Timestamps of the anomalies were recorded to increase the efficiency of the tagging process. Then those timestamps and the sudden peak values visible in the acceleration graph were used to identify the data points containing anomalies. We tagged a single point as an anomaly. Also, we make sure that this training data set contains data from different vehicles, different roads, and different phones. Hence, it has a good combination. Since we got a skewed training dataset, we had to do oversampling to remove the skewness. Otherwise, we observed that the classification model does not identify anomaly points in the training dataset.

Then using this training dataset we trained our machine learning model. Using this trained model we predicted results for a road segment in which we periodically collected data. Then we sent these predicted anomaly coordinates to the clustering process.

The actual number of anomalies on the road: 109

Table 5.2: Results of anomaly detection using Random Forest Classifier.

Scenario	Count
System correctly predicted an anomaly	38
System incorrectly predicted an anomaly	3
System missed an anomaly	71
Precision	0.926
Recall	0.349

5.2.4. Clustering

For the clustering process, the system uses GPS coordinates of anomalies identified through the classification process. Here system clusters GPS coordinates within a radius of five meters to a single cluster. We used five-meter range because of GPS error. In the end, the system identifies these cluster centers as the GPS coordinates with road anomalies. The following figure shows the result we obtained for a road segment which we collected data seven times. Here we use sample size as seven for the DBSCAN [17] clustering model.

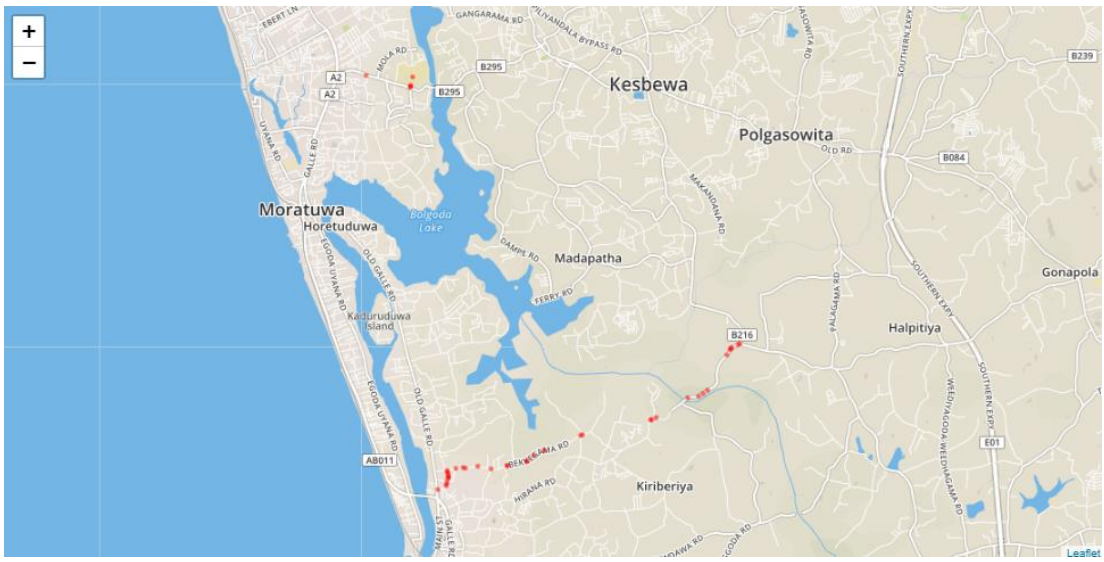


Figure 5.16. Resulted anomalies from clustering.

5.2.5. Summary

During our research, we were able to successfully identify anomalies using accelerometer readings of a mobile phone. Also, using a clustering approach we were able to identify GPS coordinates of anomalies more accurately. Since this is a crowdsourced solution system capable of more accurately identifying anomalies with time. In our research, we didn't differentiate bumps from potholes because we didn't observe any relationship to differentiate them. We use several signal processing techniques to identify a special characteristic to differentiate a pothole form a bump. Since this research is very much connected to practical scenarios it is very hard to differentiate a pothole from a bump because in Sri Lankan roads drivers always try to

bypass anomalies. Therefore, an actual bump may give a sensor reading that cannot be clearly distinguished as a bump.

5.3. IRI Prediction

5.3.1. Linear Regression Analysis

When considering the IRI calculation in miniROMDAS it mainly depends on bump integrator raw roughness value. This raw roughness value is a pulse count which indicates movement of the vehicle body with respect to the vehicle axle. In our application, the vehicle body movement measures are accelerometer readings. Following graph shows how acceleration along Y-axis changes in different scenarios.

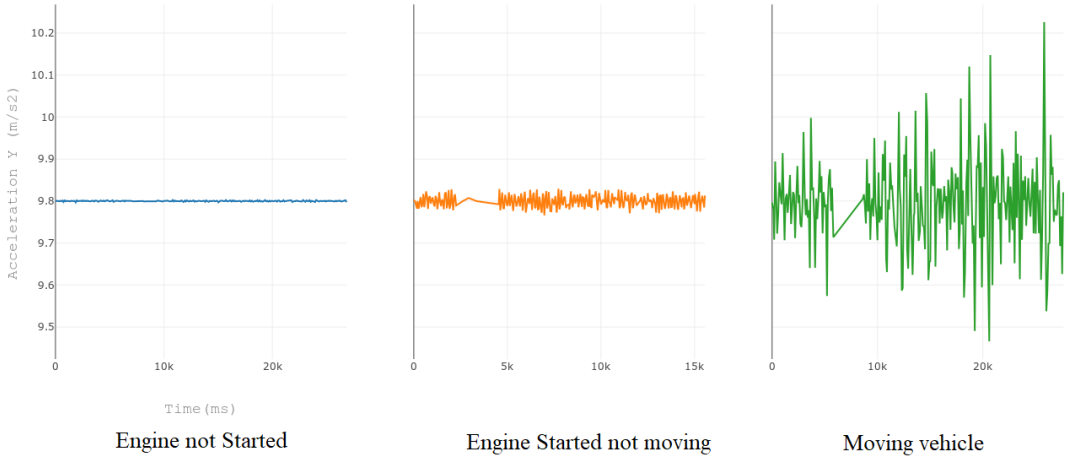


Figure 5.17: y-axis acceleration in different scenarios.

According to Figure 8.17, we can see a significant change in acceleration of Y-axis when the vehicle is moving. But still, there are y-axis acceleration changes when the vehicle is not moving. But in romdas, bump integrator is not increasing raw roughness(pulse count) when the vehicle is not moving. Because there is no relative movement in the vehicle body with the vehicle axle. So to match this pulse count to the acceleration y values we need threshold based pulse counting mechanism.

Our Pulse counting algorithm is a very simple one. If acceleration value is greater than the threshold value that data point considered as a pulse(spike). To analyze the

collected data we did correlation analysis for different features. In Table 5.1, we can see that the system gives roughness values for every 100m road segments. So we segmented rides into 100m road segments and analyzed the pulse count. Following graphs show how correlation coefficient between, bump integrator pulse count and iRoads calculated pulse count changes with accelerometer threshold value.

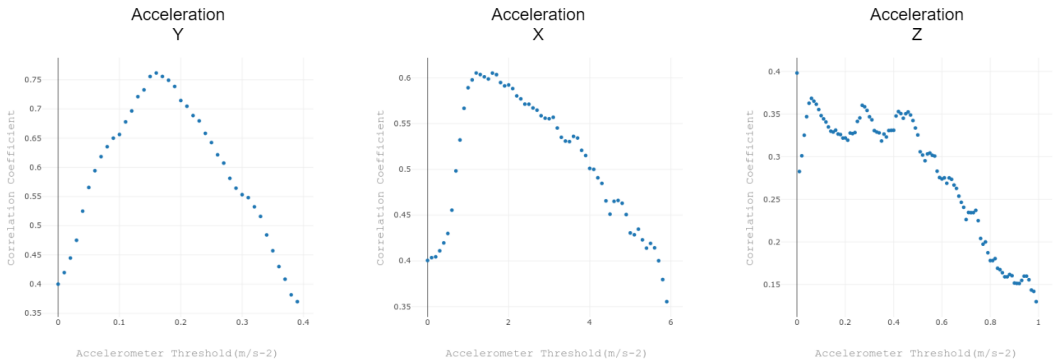


Figure 5.18. Acceleration threshold vs. correlation coefficient.

Following table shows, at what threshold value these correlations give the highest value.

Table 5.3. Highest correlation values of three axes for 100m road segments.

Axis	Highest correlation Value	Threshold(ms-2)
Y	0.7616	0.16
X	0.6052	1.2
Z	0.3982	0

Then we increased the distance of road segments and check the correlations. From that, we identified that correlations are much better for the 500m road segments. Following table shows those obtained correlation values.

Table 5.4. Highest correlation values of three axes for 500m road segments.

Axis	Highest correlation Value	Threshold(ms-2)
Y	0.9135	0.15
X	0.7686	1.2

Z	0.5833	0

Then we divided data set into sections according to speed (average speed) of the vehicle. Figure 5.19 shows ROMDAS pulse count vs iRoads calculated pulse count.

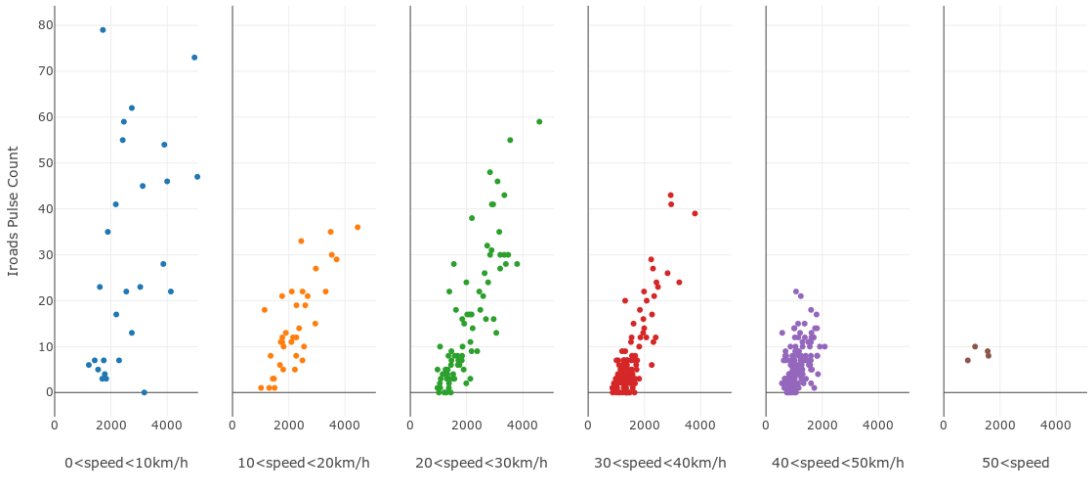


Figure 5.19: ROMDAS pulse count vs iRoads calculated pulse count for 100m road segments.

The correlation coefficients for the above speed ranges are included in below table. According to these results, we can see that when speed is in 10km/h - 40km/h range acceleration Y pulse count is leading to better correlation.

So according to above results we can see that acceleration y pulse count and acceleration x pulse count have better correlation with ROMDAS pulse count, 500m road segments give higher correlation than 100m road segments and dividing data into speed regions give different correlation for them.

Table 5.5: Correlation with speed ranges for 100m road segments.

Speed Range(km/h)	**Correlation Coefficient**
0 - 10	0.4588
10 - 20	0.7840
20 - 30	0.8393
30 - 40	0.8326
40 - 50	0.4770
50 -	0.2698

So we obtain the best fit line for the calculate IRI value for:

- 100m road segments
- 300m road segments
- 500m road segments

The ROMDAS pulse count is a raw value obtained from bump integrator. That pulse count depends on the vehicle. But using the calibration profile in ROMDAS software we can obtain vehicle independent IRI values for road segments. This IRI value and ROMDAS pulse count have a linear relationship. So we used iRoads pulse count to calculate IRI values of road segments.

Figure 8.20 shows results obtained from best-fit line calculations. Table 5.6 shows slope, intercept and mean absolute error for above three lines. This mean absolute error is calculated using the same dataset that used for calculating best-fit lines.

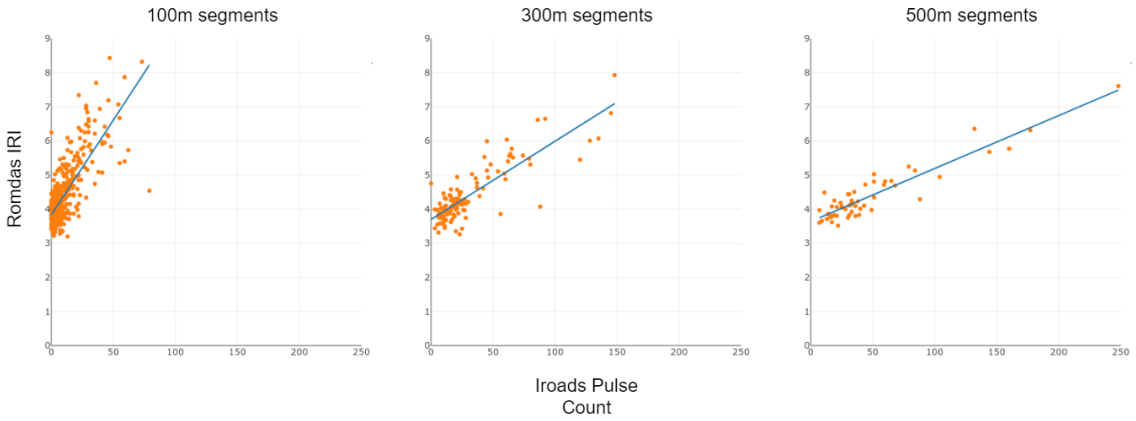


Figure 5.20: Best fit line calculation results.

Table 5.6: Attributes of best fit lines.

Segmentation	Slope	Intercept	Equation	Mean Absolute Error
100m	0.05558	3.8329	y=0.05558x + 3.8329	0.4312
300m	0.02293	3.6975	y=0.02293x + 3.6975	0.3186
500m	0.01553	3.6380	y=0.01553x + 3.6380	0.2285

Then we calculated mean absolute error using new data set collected from same delica vehicle. Following table shows MAE of those data. To reduce this MAE further we calculated best-fit lines for speed ranges. Following Table 5.7 shows those best fit lines calculation results.

Table 5.7: MAE of best-fit lines.

Segmentation	Mean Absolute Error
100m	0.6430
300m	0.5798
500m	0.4970

Table 5.8: Attributes of best-fit lines with speed ranges.

Segmentation	Speed Range(km/h)	Slope	Intercept	Mean Absolute Error	Mean Absolute Error (Different Data)
100m	0 - 10	0.02409	4.9104	0.9066	1.4033
	10 - 20	0.07085	4.1012	0.4143	0.5633
	20 - 30	0.05501	4.07506	0.3858	0.4339
	30 - 40	0.05790	3.8860	0.2772	0.4227
	40 - 50	0.04132	3.5925	0.2711	0.4054
	50 -	0.08400	3.321	0.35	0.4270
300m	0 - 10	-	-	-	-
	10 - 20	0.01774	4.2725	0.4513	0.3199
	20 - 30	0.01629	4.1677	0.4128	0.8760
	30 - 40	0.02254	3.7641	0.1457	0.3697
	40 - 50	0.01579	3.5481	0.2432	0.3734
	50 -	-	-	-	-
500m	0 - 10	-	-	-	-
	10 - 20	0.01325	4.2369	0.2403	-
	20 - 30	0.01173	3.9747	0.1995	0.3056
	30 - 40	0.01385	3.6488	0.1987	0.3779
	40 - 50	0.01697	3.4254	0.1587	0.4981
	50 -	-	-	-	-

Table 5.9 shows the total MAEs when best fit lines calculated for speed ranges.

Table 5.9: Total MAE for different segment lengths.

Segmentation	Total MAE	Total MAE(Different Data)
100m	0.2593	0.4408
300m	0.2688	0.4347
500m	0.1911	0.3977

According to these results, we can see that we can reduce error by dividing dataset into speed ranges and calculate IRI for speed ranges using unique equations for a speed range.

5.3.2. Machine Learning Prediction

Figure 5.21 shows the prediction results when using actual vehicle speed and y-axis spike count as the training features. This trained model gives MAE score of 0.3224.

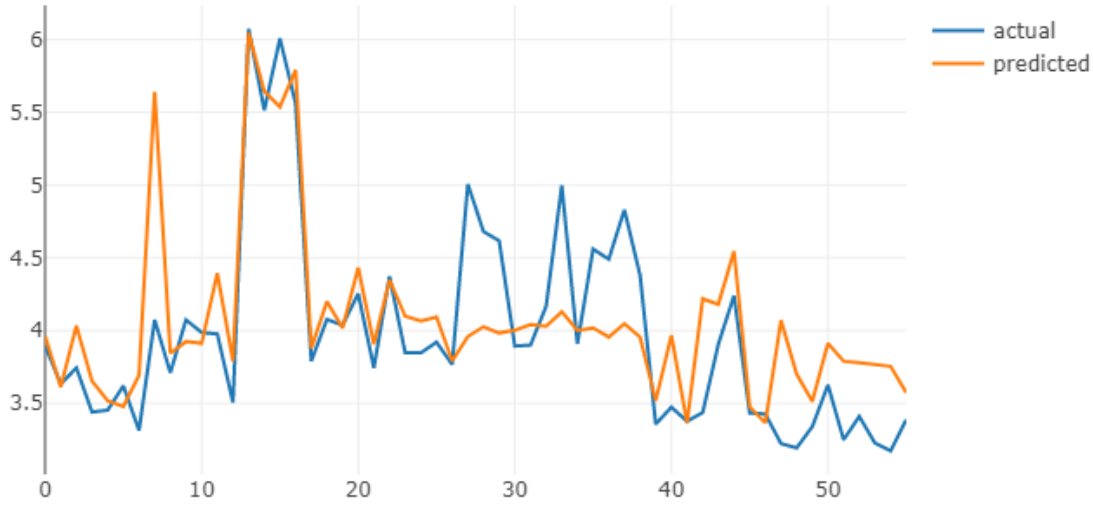


Figure 5.21: Prediction using actual vehicle speed.

Figure 5.22 shows the results when using GPS speed and y-axis spike count as the training features. This trained model gives MAE score of 0.4316.

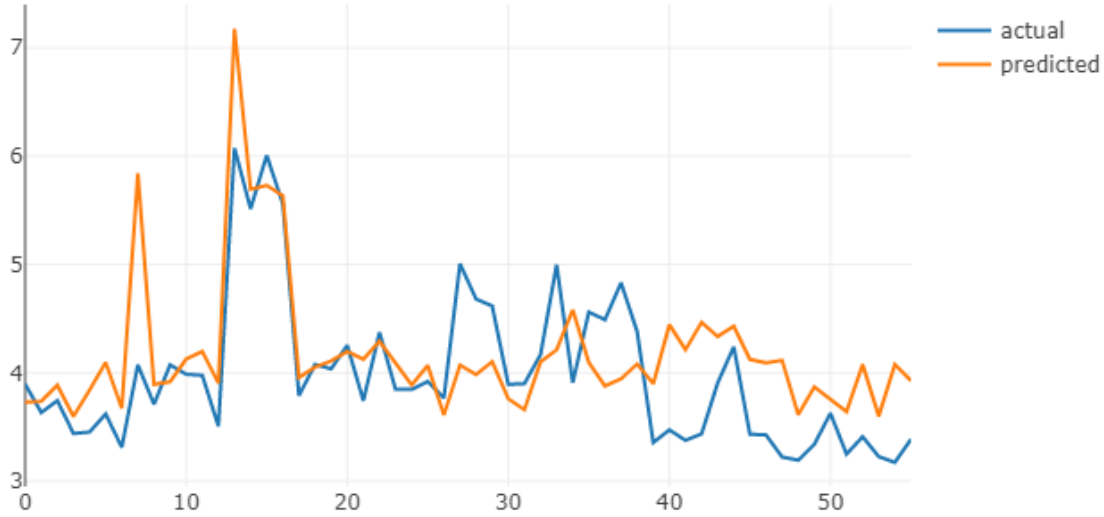


Figure 5.22. Prediction using GPS speed.

When comparing the above results from Machine Learning process, predictions are more accurate when using actual vehicle speed. So using OBD adapters actual vehicle speed could be collected and used in prediction for better results. MAE score of 0.4316 is considerably enough since the predicted values have only varied from the actual values averagely from +0.4 or -0.4 units.

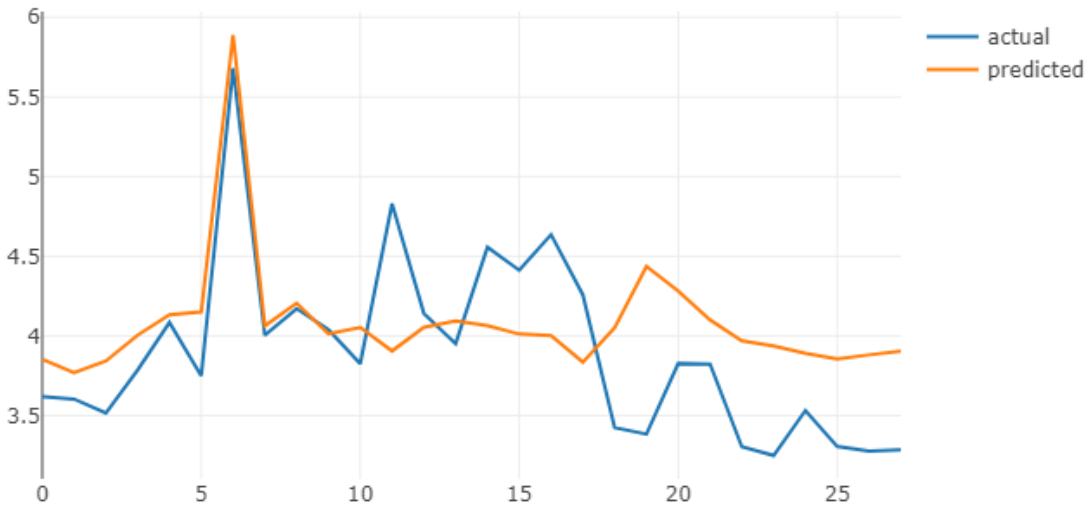


Figure 5.23. Prediction for 500m road segments.

In both of the above mentioned scenarios, the segment size of 300m was used in the training prediction model. Figure 8.23 shows the prediction results obtained from a model trained with 500m road segments. Here MAE has reduced to 0.3913 which is

less than the value given from 300m road segments. According to the results from models trained using different road segment sizes, we could see the MAE reduces with the increasing of road segment size as shown in table 8.10.

Table 5.10: Total MAE by segment size.

Segmentation	Total MAE
100m	0.6508
300m	0.4316
500m	0.3913

5.3.3. Conclusion

Following table 8.11 illustrate the comparison between the linear regression method and the machine learning approach used for prediction IRI values.

Table 5.11: Total MAE by segment size.

Segmentation	MAE from Machine Learning	MAE from Linear Regression
100m segments	0.6430	0.6508
300m segments	0.5798	0.4316
500m segments	0.4970	0.3913

According to the MAE values, we could conclude that both Machine Learning approach and Linear regression method are more accurate in IRI prediction when road segment size is increasing. Also the overall accuracy of the Linear regression method is lower than the Machine Learning approach.

6. Conclusion

6.1. Summary

As an outcome of this research and development project, we were able to develop a system to measure and monitor road conditions in Sri Lanka. This is a low-cost solution compared to existing solutions because this solution uses only a smartphone. This developed system is capable of detecting anomalies on roads and visualize them

on a map. Also, this system is capable of calculating IRI values in a more simpler and cheaper way compared to the currently existing method in Sri Lanka. Backend servers of the system can collect data related to the road network in Sri Lanka for a longer time period. Hence, this would be a good data source to analyze traffic patterns, road deterioration in Sri Lankan roads. Since this is a crowdsource solution where the general public also can contribute to the system by collecting data. And it would be a good initiative for a better road network in Sri Lanka. Though we mentioned that we are going to measure the relationship between roughness and fuel consumption in Sri Lankan roads, we didn't research on that aspect due to time limitation. As the final conclusion, we can say that this system can be further calibrated to gain more accurate results when predicting anomalies and calculating IRI values.

6.2. Limitations

Dependency evaluation needs more data according to each scenario. We faced limitations on data collecting. Creating quantitative analysis on depending factors needs more data and precise concern on tools that are used throughout the data gathering process such as mounting brackets.

6.2.1. Dependency Evaluation

As iRoads a crowdsourced solution there are several factors to be considered when collecting data from different environments. We have identified those scenarios as,

1. different vehicles
2. different phones
3. different speeds

Our project focused on doing research in these scenarios by collecting data for all these situations as much as possible.

6.2.1.1. Experimental Evaluation & Discussion of Results

There is a notable difference in the graphs in Figure 6.1 and 6.2. Data point density has changed between these two scenarios within one minute (in Figure 6.1 Point count is 597 and in Figure 6.2 Point count is 1042). So, different phones have different

frequencies for their sensors. This provides different densities in different devices and it affects the calculation of IRI than anomaly detection.

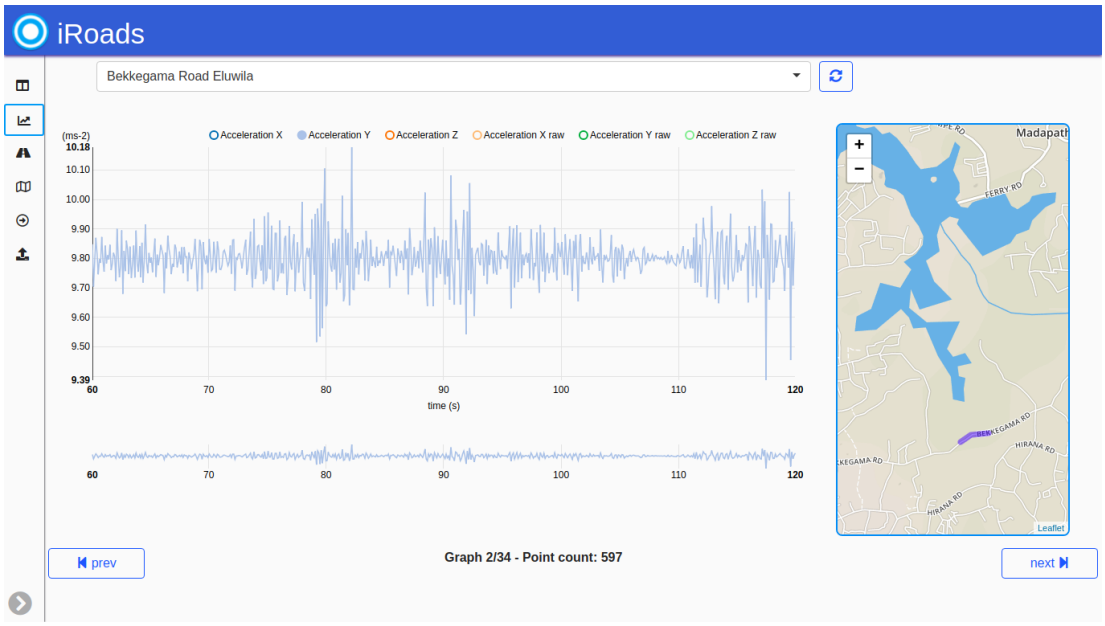


Figure 6.1. Acceleration data from HTC device in a Honda Vezel.

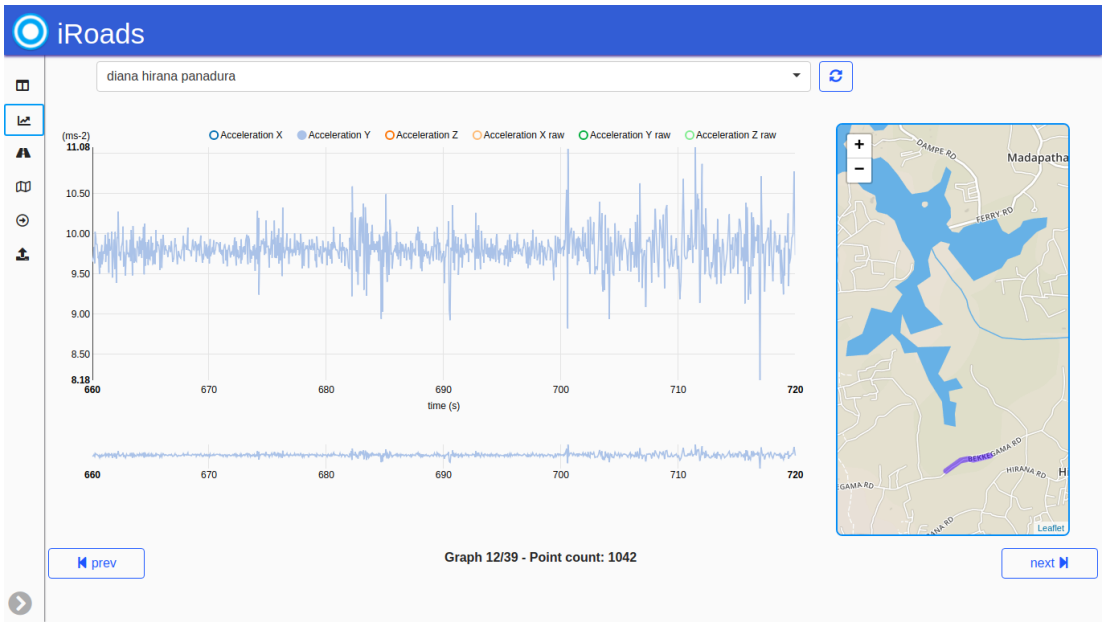


Figure 6.2. Acceleration data from Xiaomi device in a Mitsubishi Delica.

When checking the similarity of the graphs with same speeds, same vehicle, and different phone within one minute in Figure 6.3 and Figure 6.4, different data point

densities were noticed and it shows different peak points and we noticed that value of gravity line (approximately 9.8) has changed.

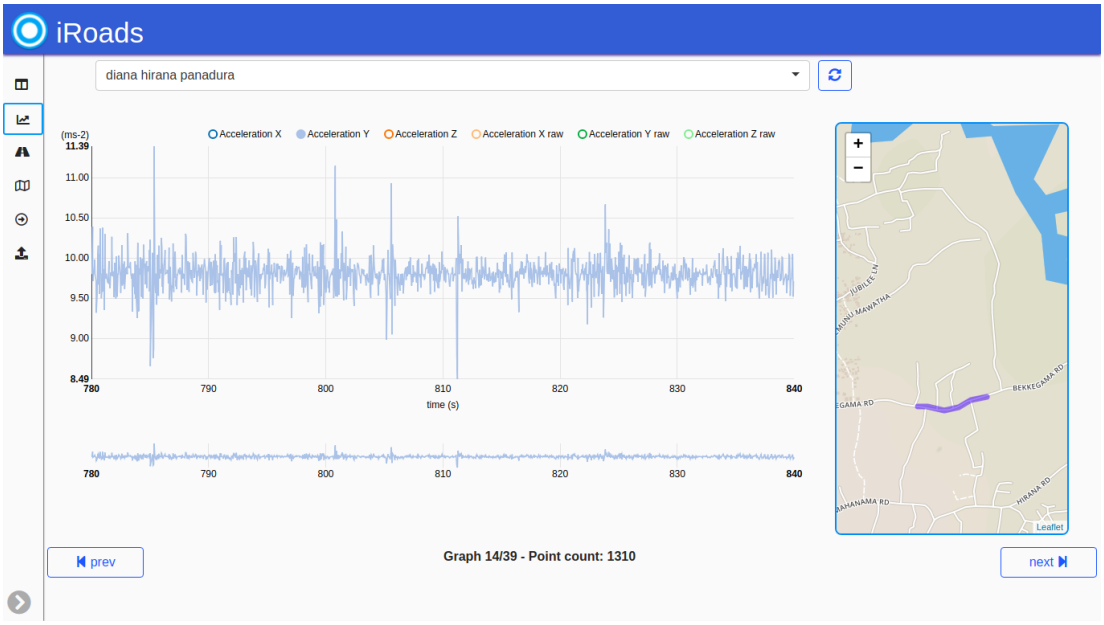


Figure 6.3. Acceleration data from Xiaomi device in a Mitsubishi Delica.

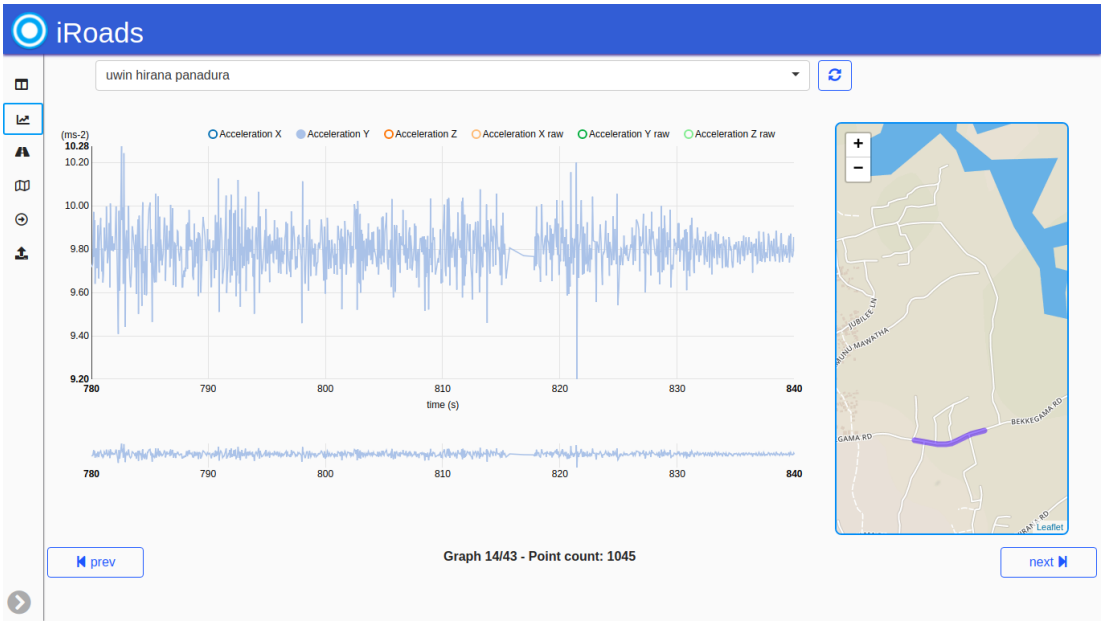


Figure 6.4. Acceleration data from HTC device in a Mitsubishi Delica.

6.3. Future Work

Dependency evaluation and calibrate the system accordingly is yet to be done. As this mainly affect in IRI calculation, the system needs to be improved to remove vehicle dependency and phone dependency. At the moment we can calibrate IRI for different vehicles by manually calculating correlation with bump integrator values. Speed dependency may affect in several ways because correlation scores well in some speed ranges. So, removing speed dependency may require more research work.

Current system identifies the potholes and bumps as anomalies. So, classifying anomalies into bumps and potholes is yet to be done. Bumps and potholes identification became a complex task because the expected behavior of potholes and bumps were not actually existed as ups and downs wise.

To create more precise IRI estimation we can train a model with more IRI data including a wide range of IRI values.

Table 6.1. Limitations and Future Work

Limitation	**Future work**
IRI calculation is dependent on vehicle & smartphone used for the analysis	Collection of a large dataset covering multiple vehicles & smartphones
Device errors affected accuracy of calibrator	Use Class 1 & 2 IRI measuring devices
Estimate relationship between fuel consumption & IRI	Gather fuel consumption data from OBD and analyze
Differentiation of anomalies was not implemented	Look for advanced techniques to differentiate potholes & bumps

References

[1] "Road Profiler, Non Contact Profilometer | International Cybernetics", Intlcybernetics.com, 2018. [Online]. Available: http://www.intlcybernetics.com/road_profiler.html. [Accessed: 15- Jul- 2018].

[2] "*International Roughness Index - LGAM Knowledge Base*", (2010). [Online]. Available: http://www.lgam.info/international-roughness-index. [Accessed: 30- Jan- 2018].

[3] Romdas.com. (2018). *Bump Integrator | Road Roughness Measurement , Road Survey*. [online] Available at: https://romdas.com/romdas-bump-integrator.html [Accessed 13 Jul. 2018].

[4] P. V. P. Tonde, A. Jadhav, S. Shinde, A. Dhoka, and S. Bablade, "Road Quality and Ghats Complexity analysis using Android sensors," *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 4, Issue 3, 2015, pp. 101–104.

[5] P. Mohan, V. N. Padmanabhan and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones", *6th ACM conference on Embedded network sensor systems SenSys '08*, Raleigh, NC, USA, 2008, pp. 323-336.

[6] R. Bhoraskar, N. Vankadhara, B. Raman and P. Kulkarni, "Wolverine: Traffic and road condition estimation using smartphone sensors," *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, Bangalore, 2012, pp. 1-6.

[7] 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", *The Sixth Annual International conference on Mobile Systems Applications and Services (MobiSys 2008)*, Breckenridge, CO, USA, 2008, pp 29-39.

[8] A. Ghose, P. Biswas, C. Bhaumik, M. Sharma, A. Pal and A. Jh,. Road condition monitoring and alert application: Using in-vehicle Smartphone as Internet-connected sensor. *PerCom Demos 2012*, Lugano, 2012, pp.489-491.

[9] D. Gamage, H.R. Pasindu, and S. Bandara, "Pavement roughness evaluation method for low volume roads", *8th International Conference on Maintenance and Rehabilitation of Pavements*, Singapore, 2016, pp. 976-985.

[10] L. Forslöf and H. Jones, "Roadroid: Continuous Road Condition Monitoring with smart phones", *Journal of Civil Engineering and Architecture*, 2015, pp. 485-96.
[11] "AndroSensor for Android", Fivasim.com, 2018. [Online]. Available: http://www.fivasim.com/androsensor.html. [Accessed: 15- Jul- 2018].
[12] Spring.io. (2018). Spring Projects. [online] Available at: https://spring.io/projects/spring-boot [Accessed 13 Jul. 2018].
[13] M. Amarasinghe, A. L. Arachchi, S. Muramudalige and H. M. N. D. Bandara, "Vatichub OBD2-Core", *GitHub*, 2016. [Online]. Available: https://github.com/vatichub/obd2-core. [Accessed: 15- Jul- 2018].
[14] "Developer Portal | A Better Location Stack for iOS and Android", developer.pathsense.com, 2018. [Online]. Available: https://developer.pathsense.com. [Accessed: 04- Nov- 2018].
[15] Community.esri.com. (2018). *Distance on a sphere: The Haversine Formula | GeoNet*. [online] Available:
[16] Rdocumentation.org. (2018). *randomForest function | R Documentation*. [online] Available at: https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest [Accessed 13 Jul. 2018].
[17] Rdocumentation.org. (2018). *dbscan function | R Documentation*. [online] Available: https://www.rdocumentation.org/packages/dbscan/versions/1.1-2/topics/dbscan [Accessed 13 Jul. 2018].
[18] *Michigan.gov, "International Roughness Index (IRI)*," Nov-2017. [Online]. Available: http://www.michigan.gov/documents/mdot/International_Roughness_Index_605990_7.pdf [Accessed 13 Jul. 2018].
[19] "3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.20.0 documentation", Scikit-learn.org, 2018. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. [Accessed: 07- Nov- 2018].
[20] "SharedPreferences | Android Developers", Android Developers, 2018. [Online]. Available:

https://developer.android.com/reference/android/content/SharedPreferences. [Accessed: 06- Nov- 2018].
[21] "Save data using SQLite | Android Developers", Android Developers, 2018. [Online]. Available: https://developer.android.com/training/data-storage/sqlite. [Accessed: 06- Nov- 2018].
[22]N. Soroka, "nolia/Noodle", GitHub, 2018. [Online]. Available: https://github.com/nolia/Noodle. [Accessed: 06- Nov- 2018].
[23] A. Masny, "pilgr/Paper", GitHub, 2018. [Online]. Available: https://github.com/pilgr/Paper. [Accessed: 06- Nov- 2018].
[24] "Firebase Realtime Database | Firebase Realtime Database | Firebase", Firebase, 2018. [Online]. Available: https://firebase.google.com/docs/database/. [Accessed: 06- Nov- 2018].
[25] "Couchbase Lite", Docs.couchbase.com, 2018. [Online]. Available: https://docs.couchbase.com/couchbase-lite/1.4/index.html. [Accessed: 06- Nov- 2018].
[26] "Anaconda Documentation — Anaconda 2.0 documentation", Docs.anaconda.com, 2018. [Online]. Available: https://docs.anaconda.com/. [Accessed: 12- Nov- 2018].
[27] "Python Data Analysis Library — pandas: Python Data Analysis Library", Pandas.pydata.org, 2018. [Online]. Available: https://pandas.pydata.org/. [Accessed: 13- Nov- 2018].
[28] "NumPy — NumPy", Numpy.org, 2018. [Online]. Available: http://www.numpy.org/. [Accessed: 13- Nov- 2018].
[29] "pyplot — Matplotlib 2.0.2 documentation", Matplotlib.org, 2018. [Online]. Available: https://matplotlib.org/api/pyplot_api.html. [Accessed: 13- Nov- 2018].
[30] "seaborn: statistical data visualization — seaborn 0.9.0 documentation", Seaborn.pydata.org, 2018. [Online]. Available: https://seaborn.pydata.org/. [Accessed: 13- Nov- 2018].
[31] F. Wasilewski, "PyWavelets - Wavelet Transforms in Python — PyWavelets Documentation", Pywavelets.readthedocs.io, 2018. [Online]. Available: https://pywavelets.readthedocs.io/en/latest/index.html. [Accessed: 13- Nov- 2018].

[32] "SciPy — SciPy v1.1.0 Reference Guide", Docs.scipy.org, 2018. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/index.html. [Accessed: 13- Nov- 2018].
[33] "scikit-learn: machine learning in Python — scikit-learn 0.20.0 documentation", Scikit-learn.org, 2018. [Online]. Available: https://scikit-learn.org/stable/index.html. [Accessed: 13- Nov- 2018].
[34] "imblearn.over_sampling.SMOTE — imbalanced-learn 0.4.3 documentation", Imbalanced-learn.org, 2018. [Online]. Available: https://imbalanced-learn.org/en/stable/generated/imblearn.over_sampling.SMOTE.html. [Accessed: 13- Nov- 2018].
[35] "sklearn.model_selection.GridSearchCV — scikit-learn 0.20.0 documentation", Scikit-learn.org, 2018. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. [Accessed: 13- Nov- 2018].
[36] "MiniROMDAS System | Road Roughness Measurement , Road Survey", Romdas.com, 2018. [Online]. Available: https://romdas.com/miniromdas.html. [Accessed: 06- Nov- 2018].
[37] "Z-250 Reference Profiler | Road Roughness Measurement", Romdas.com, 2018. [Online]. Available: https://romdas.com/romdas-z250.html. [Accessed: 06- Nov- 2018].
[38] OpenStreetMap. (2018). *OpenStreetMap*. [online] Available: https://www.openstreetmap.org [Accessed 13 Jul. 2018].
[39] Shodhganga.inflibnet.ac.in, 2018. [Online]. Available: http://shodhganga.inflibnet.ac.in/bitstream/10603/138740/8/08_chapter%202.pdf. [Accessed: 16- Nov- 2018].
[40] "codemogroup/iRoads-Mobile", GitHub, 2018. [Online]. Available: https://github.com/codemogroup/iRoads-Mobile. [Accessed: 16- Nov- 2018].
[41] "codemogroup/iRoads_Anomaly_Detection_using_Machine_Learning", GitHub, 2018. [Online]. Available: https://github.com/codemogroup/iRoads_Anomaly_Detection_using_Machine_Learning. [Accessed: 16- Nov- 2018].

Appendix

Appendix A - Extended Abstract published in TRF 2018

iRoads: Smartphone-Based Road Condition Monitoring

H.M.A. Abeywardana[1] , U.M.J. Abeywickrama[2], P.T. Amarasinghe[3],
R.P.D. Kumarasinghe[4] , H.M.N. Dilum Bandara[5], and H.R. Pasindu[6]

Abstract

Measuring and monitoring road conditions are essential to ensure public and vehicle safety, promptly maintenance, as well as fuel and time savings. While developed countries use sophisticated devices installed on specialized vehicles to measure and monitor the road conditions, it is cost prohibitive for countries like Sri Lanka. Moreover, diversity of road types and non-standard physical properties make it impractical for specialized vehicles to travel on roads in Sri Lanka and many other countries. Therefore, a system that is low cost and practically usable on roads with non-standard physical properties will be a useful solution for road condition monitoring.

Sensors such as 3-axis accelerometer, gyroscope, GPS, and magnetometer in most smartphones could be used to detect potholes and bumps, as well as estimate International Roughness Index (IRI) at a much lower cost. Related work has shown that acceleration data from smartphones have a linear relationship with road roughness. Hence, it opens the way to develop a system to measure road roughness using smartphone sensors. While the accuracy of such a solution is relatively low, with the increasing number of motorists with smartphones, crowdsourcing could be used to collect data at a high spatial and temporal resolution that has been hitherto possible. Such massive volume of data collected through crowdsourcing could be processed using machine-learning and signal processing algorithms such that the limitations and low accuracy of a single smartphone could be overcome by data analytics of the same road condition again and again.

A crowdsourced mobile app is proposed to measure the road conditions such as potholes, bumps, speed breakers, and estimate IRI at a high spatial and temporal granularity. The proposed solution collects data over a broadband connection to a

cloud-computing-based backend where machine-learning and signal processing algorithms are used to determine different road conditions and estimate IRI. Moreover, the solution provides visualization of this information using a map-based dashboard.

3-axis accelerometer is used as the main source for road profile monitoring. However, in a crowdsourced model, many practical problems need to be solved in addition to technical problems, as motorists may use vastly different types of smartphones with varying features and accuracy. For example, they may mount the smartphone in various orientations or orientation may change as the trip progresses. Therefore, a reorientation mechanism is essential to convert accelerometer data from any arbitrary smartphone position to the vehicle's axis. The solution implements two reorientation mechanisms. The first mechanism is using Euler angle-based algorithm.

The second mechanism uses magnetometer and GPS bearing readings to reorient the acceleration vectors of the mobile device. Signal processing techniques are used to filter out the sensor noise for more accurate data gathering. Moreover, the magnitude of sensor reading tends to correlate with acceleration and deceleration of the vehicle. Thus, vehicle speed data are also needed to capture road conditions accurately. Therefore, the proposed app connects to an OBD2 (On-Board Diagnostic) ELM327 adapter to collect vehicular data such as fuel consumption and speed of the vehicle. OBD2-based vehicle speed estimation is more effective than GPS-based estimation due to low resolutions and slow sampling in GPS.

Random-forest algorithm is used in the backend to detect road anomalies (e.g., pothole or bump), while the pulse calculating algorithm is designed for estimating IRI values. Road segments are classified based on IRI and a map is annotated based on the IRI values. Due to varying accelerometer accuracy levels, as well as low resolution and slow sampling in GPS, it is difficult to estimate the exact location of the road anomaly. Therefore, a clustering algorithm is used to identify the location of an anomaly by clustering GPS locations estimated from different trip data provided by users. Moreover, vehicular data will be used in the future to estimation the relationship between fuel consumption and IRI of Sri Lankan roads. Furthermore, the visualization

of bad road segments could provide insights for drivers to bypass bad road segments while the authorities could use the dashboard to prioritize maintenance and policy marking.

An Android-based mobile app, namely iRoads, is developed and already used with a few data collection trails. The research currently focuses on calibrating the mobile app and related algorithms to accurately estimate IRI and detect road anomalies. For example, efforts are currently underway to calibrate estimated IRI values with the IRI readings from a ROMDAS Bump Integrator. The goal is to improve the accuracy to such a level that iRoads could measure roughness like a class-3 road profiling instrument. Another app is also developed to label road anomalies on the go such that a large, labeled training dataset could be gathered for training and evaluation of machine-learning and signal processing algorithms. Based on this dataset model parameters are to be tuned to more accurately estimate road anomalies.

keywords: IRI, road anomaly, accelerometer, signal processing, machine learning

---

[1] Undergraduate Dept. Computer Science & Engineering, aminda.14@cse.mrt.ac.lk

[2] Undergraduate Dept. Computer Science & Engineering, uwin.14@cse.mrt.ac.lk

[3] Undergraduate Dept. Computer Science & Engineering, pivithuru.14@cse.mrt.ac.lk

[4] Undergraduate Dept. Computer Science & Engineering, dushan.14@cse.mrt.ac.lk

[5] Senior Lecturer, Coordinator Industrial Training & MBA in IT, Director Engineering Research Unit Dept. Computer Science & Engineering, University of Moratuwa, Sri Lanka, dilumb@cse.mrt.ac.lk

[6] Senior Lecturer, Transportation Engineering Division, Department of Civil Engineering, University of Moratuwa, Sri Lanka, pasindu@uom.lk