Crater: A Crowd Sensing Application To Estimate Road Conditions

By

Faria Kalim 2011-NUST-SEECS-BESE-259

Hamza Naveed 2011-NUST-SEECS-BESE-214

Advisor

Dr. Muhammad Usman Ilyas

Department of Electrical Engineering

Co-Advisor

Dr. Tahir Azim

Department of Computing

A Project Report submitted in partial fulfillment of the requirements for the degree of Bachelor of Software Engineering (BESE), Department of Computing

In

School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(May 2015)

Certificate

It is certified that the contents and form of thesis entitled "Crater: A Crowd Sending Application To Estimate Road Conditions" submitted by Faria Kalim (2011-NUST-SEECS-BESE-259) and Hamza Naveed (2011-NUST-SEECS-BESE-214) have been found satisfactory for the requirement of the degree.

Dr. Usman Ilyas

Dr. Tahir Azim

To my family and faculty.

Since I can not possibly pay you back for all that you have given me, I pray instead that may Allah SWT reward you in this world and the next, and give me the prowess and the opportunity to pay your efforts forward.

Acknowledgements

Dr. Usman Ilyas provided us with invaluable guidance and help at every step of the project. He gave us his time whenever we chose to walk into his office, and tirelessly explored opportunities to fund the project and take it to a level where it would be more than just an undergraduate final year project. His mentoring changed the way we evaluate information and allowed us to develop a more analytical approach to problems.

We would also like to thank Dr. Tahir Azim for his valuable input, mentoring and general suggestions for the project.

We will forever be thankful to our advisors for their helpful career advice and significant efforts to help us get started with our future endeavours.

We would also like to thank Asbah Ashfaq for helping us out with some graphical user interface design of the project, and Sundus Mariya for lending her ear when we were in need.

Abstract

We have designed and developed a smart phone application that measures shocks and vibrations when it finds itself on the road in order to map and measure the locations of potholes and speed breakers. Simultaneously, it also measures the vehicle's speed in order to measure traffic congestion as a function of the number of users at a particular location and their speed relative to free flow conditions (when there is no congestion).

The application does not require any input from its user and reports measurements to an Azure hosted application which stores data from cellphones of all users and jointly processes this data to obtain a better and more complete estimate of road conditions, and traffic congestion. This information is published on separate maps that is made publicly available via our website.

The uploaded information allows citizens and municipal authorities alike to spot potholes, road segments in need of repair, congestion hubs, and imbalances in infrastructure maintenance efforts across cities. If the municipality has data of approved and legally constructed speed bumps, it is able to identify all others as illegally constructed. The government can use this information to allocate more funds to areas that are desperately in need of repair and see more traffic. On the other hand, the public can keep an eye on how efficiently the authorities work to repair road conditions in their area and if politicians actually carry out the actions that they claim to do during election season.

Table of Contents

1 Introduction **1**
- 1.1 Problem Statement . . . 1
- 1.2 Proposed Approach . . . 2
- 1.3 Scope . . . 2
- 1.4 Contents of This Report . . . 3

2 Literature Review **4**
- 2.1 Existing Applications . . . 4
  - 2.1.1 Street Bump . . . 4
  - 2.1.2 Waze . . . 5
  - 2.1.3 Potholes Hunter . . . 5
  - 2.1.4 Fill That Hole . . . 5
  - 2.1.5 Find It, Fix It . . . 5
- 2.2 Published Materials . . . 6
  - 2.2.1 Pothole Detection System using Machine Learning on Android . . . 6
  - 2.2.2 The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring . . . 6
  - 2.2.3 Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smart Phones . . . 7
  - 2.2.4 Real Time Pothole Detection using Android Smartphones with Accelerometers . . . 7
- 2.3 Goals . . . 7

3 Functionality and Design **9**
- 3.1 User Requirements . . . 9
  - 3.1.1 The Application . . . 9
  - 3.1.2 The Application Engine . . . 9
  - 3.1.3 The Website . . . 10
  - 3.1.4 The Database . . . 10
- 3.2 Functional Requirements . . . 11

3.2.1 The Application 11
3.2.2 The Website 16
3.2.3 The Azure Mobile Service 16
3.2.4 The App Engine 16
3.3 Database Design 17
3.4 Data Flow Diagram 18
3.5 High Level System Architecture 18
3.6 Important Design Decisions 21
3.6.1 Google Location APIs vs GPS Sensors in Android Application : Accuracy and Activity Recognition 21
3.6.2 SMO: In the application or at the server end 23

4 Implementation and Results **25**
4.1 Software Development Life Cycle and Challenges Encountered 25
4.1.1 Planning and Requirement Gathering 25
4.1.2 Building the Ground Truth Application 25
4.1.3 Gathering Labelled Data 26
4.1.4 What do Potholes and Speed Breakers Look Like...On Paper? 27
4.1.5 Testing Classifiers for Best Performance 31
4.1.6 Building the Final Apps, the Website and the Back-end 34
4.1.7 Testing and Deployment 34
4.1.8 Implementation Details 40

5 Conclusion and Future Discussion **43**
5.1 Findings and Significance 43
5.2 Commercialization 44
5.3 Limitations 44
5.4 Improvements 44

List of Figures

3.1 Database Design 18
3.2 Data Flow Diagram 19
3.3 Architectural Diagram 19

4.1 A Plot of Raw Accelerometer Readings of A Speed Breaker . . 27
4.2 A Plot of Raw Accelerometer Readings of A Pothole 28
4.3 A Plot of low-Pass Filtered Accelerometer Readings of A Speed Breaker 29
4.4 A Plot of Low-Pass Filtered Accelerometer Readings of A Pothole 30
4.5 Naive Bayes 31
4.6 J48 32
4.7 SMO 32
4.8 Decision Tables 33
4.9 Classification via Clustering 33
4.10 Home Page of the App 35
4.11 Login Page of the App 36
4.12 A Map from the App 37
4.13 About Page from the App 38
4.14 Main Page from the Website 38
4.15 About Page from the Website 39
4.16 Maps Page from the Website 39
4.17 Calculate Free Flow Conditions 41
4.18 Convert Difference in Latitude and Longitude to Distance . . 41
4.19 Correct the Device's Orientation 42

List of Tables

3.1 RESTful Services per Table 17

4.1 Distribution of Labelled Data 31

Acronyms

- FFT: Fast Fourier Transform
- SMO: Sequential Minimal Optimisation

Chapter 1

Introduction

1.1 Problem Statement

Road conditions in Pakistan vary from excellent to dilapidated. Urban infrastructure funds are limited and are spend without giving the public any visibility. Every year, the monsoon season brings torrential rain which pushes bad conditions to worse, and sometimes to intolerable levels. It is a frequent complaint that patients Some governments try to improve the road network through efforts to build new roads, flyovers and underpasses; such efforts are limited to urban areas and the major cities of Pakistan. Furthermore, these efforts generally involve ripping up the old most-frequented routes and cordoning off these sections of the road network until the repair work is finished.

On occasions that authorities are held accountable for poor road conditions in residential areas or in a small locality, the response is that citizens need to report locations so that the municipality can act. In addition to suffering from disrepair, people sometimes construct illegal speed breakers on public roads to slow down traffic near their homes and businesses. Lack of reporting means that sometimes it can be years before such speed breakers are actually removed.

At the same time, Pakistan is a country with a growing population. Recently, the number of cars in urban areas has increased exponentially and unfortunately, the road network has not grown at the same rate. This leads to traffic congestion at central locations in the cities at peak times. This is not only a waste of time, but also of energy and money in terms of fuel. Researchers at the Texas A&M Transportation Institute estimate that congestion and rough roads will cost the average Texas household $6,100 a year in wasted fuel, vehicle repairs, and time lost sitting in traffic between now and 2035 [15]. While the conditions are such in a developed country, we can

only fathom what it must be costing the citizens of Pakistan to waste time sitting in traffic on terrible roads.

1.2 Proposed Approach

While the use of smartphones is already ubiquitous in urban areas of Pakistan, they are rapidly pervading the rural setting. On the other hand, with the advent of 3G/4G technologies in Pakistan, more and more people have begun to adopt mobile broadband connections. Given these combined factors we were motivated to come up with a solution to our problem that leverages the high density of smartphone users in Pakistan's urban centers to crowd sense information about road and traffic conditions.

Our solution consists of a zero-input crowdsensing application with a minimum cognitive load i.e. all the user should have to do is to install it on his phone. The application runs in the background while the user is on the road, and gathers information about the jerks and vibrations that a vehicle experiences. Then, this information is classified to determine what exactly the vehicle went over and is uploaded to the server. There, another classifier runs to determine what the majority of users say is present at a particular geographical location. Having determined this value, it stored in the database, ready to be published to our website. This website is available for the viewing pleasure of all. [8]

Our solution helps both the government and the public by providing them with much-needed information to make important decisions through an automated mechanism. All worries about updating information manually are removed, as the system shall take updated information from users all the time while they are driving on the roads.

Government authorities can go through this information to refine their fund allocation and policies - segments of the road network that is in desperate need of repair should be catered to first as an utmost priority. Congestion hubs in the cities should be noted, and alternate routes can be developed. The public can simultaneously check whether government efforts actually bring results or not.

1.3 Scope

The scope of this project covers mapping:

- Road conditions by measuring on a road:

  - Potholes
  - Speed breakers

- Traffic congestion by measuring:
  - the number of users at a particular spot
  - the speed of aforementioned users

Initially, the scope included describing overall road conditions in terms of their general condition, and mapping traffic conditions was only an option. However, traffic conditions are actually measured and added to the application because they help users make an important decision - choosing routes with no/little congestion - and this helps motivate more users to adopt the application. Due to lack of time, describing overall road conditions was skipped.

1.4 Contents of This Report

This report moves from the introduction of our project to related works described in the literature review. That section is used to describe how our application has a more significant contribution than the previous works. The third section describes the functionality that is to be implemented as part of the system and important design decisions that we had to make in order to use the user's device memory and bandwidth in an optimal way. The fourth section explains how we completed the implementation of the project, moved through the software development life cycle and what results we produced. Finally, in the last chapter, we conclude with the significance of our contribution and how the project can be taken forward in the future.

Chapter 2

Literature Review

Previous works on the topic generally describe research or existing applications that have the following features that we would like to avoid:

- Lack of crowd sensing.
- Required user participation in reporting potholes.
- Specialised hardware requirements.
- Incomplete coverage of road anomalies e.g. speed breakers are left out or ignored.
- Only available for commercial or regionally limited use
- The device which gathers the data is forced to be still or not used while the data gathering process is going on

2.1 Existing Applications

2.1.1 Street Bump

This app tracks bumps and potholes using the phone's accelerometer. However, this particular app struggles with speed bumps and manhole covers, and additionally does not provide the user with the output of its crowd sensing. Instead, the app sends the data to the local government's road maintenance workers. These organizations will then use this information to fix problematic road conditions after three or more bumps occur at the same location. This app is currently available on the Apple App store for free. [3]

2.1.2 Waze

Waze is supposedly the world's largest community-based traffic and navigation app, although it is mainly used in the United States of America. It invites drivers to help each other share real-time traffic and road info about the local area, saving everyone time and gas money on their daily commute. However, the application does not focus on mapping road conditions and is not well-publicized in the third world. [1]

2.1.3 Potholes Hunter

This app is a pothole tracker (available on the Google Play store for free) intended to be used in Hungary. Users take photos and rank the worst potholes in the country. The app also places these potholes on an overlay of a map. Usability is reduced because users are reluctant to stop and actually record the potholes location. [7]

2.1.4 Fill That Hole

Another app like Potholes Hunter, this app focuses on bicyclists for its target user base. Again, users must photograph and fill out forms about certain potholes on their own. However, much like Potholes Hunter usability of this app relies on its users taking time to actually record the potholes in their area. This points out the most critical aspect of our project: making an app that will be user friendly, and simple. This will ensure that users actually use the app. [6]

2.1.5 Find It, Fix It

The city of Seattle is currently offering the this app for city residents to report potholes, along with other items such as graffiti and abandoned vehicles. The reports go directly to the city, and then they are processed by the appropriate departments. There is no way for residents to see potholes others have submitted, so they know what areas to avoid. [5]

2.2 Published Materials

2.2.1 Pothole Detection System using Machine Learning on Android

This paper investigates an application of mobile sensing: detection of potholes on roads. It describes a system and an associated algorithm to monitor the pothole conditions on the road. The Pothole Detection System uses accelerometer sensor of Android smart phones for detection of potholes and GPS for plotting the location of potholes on Google Maps. Accelerometer data and pothole data can be mailed to any email address in the form of a .csv file. While designing the pothole detection algorithm some some threshold values on x-axis and z-axis are assumed. These threshold values are justified using a neural network technique which confirms an accuracy of 90%-95%. The neural network has been implemented using a machine learning framework available for Android called "Encog".

This paper does not however take into account the benefits of crowd sensing: by gathering data from multiple users, it is likely that a larger road network would be covered much faster and the aggregated data would be more accurate. No attention is paid to classifying speed breakers either. [9]

2.2.2 The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring

This paper investigates an application of mobile sensing: detecting and reporting the surface conditions of roads. A system called the Pothole Patrol (P2), uses the inherent mobility of the participating vehicles, opportunistically gathers data from vibration and GPS sensors, and processes it to assess road surface conditions. Using a simple machine-learning approach, it is shown that potholes and other severe road surface anomalies can be identified from accelerometer data.

However, the system requires three-axis acceleration sensors and GPS devices deployed on embedded computers in cars. We propose to work only with android smart phones that users would not have to purchase separately for the purpose of this application. [4]

2.2.3 Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smart Phones

To monitor road and traffic conditions in chaotic settings, this paper presents Nericell, a system that performs rich sensing on smart phones. In this paper, the writers specifically on the sensing component, which uses the accelerometer, microphone, GSM radio, and/or GPS sensors in these phones to detect potholes, bumps, braking, and honking.

This application is also standalone, lacks a central database and does not exploit the benefits of crowd sensing data from multiple users. [11]

2.2.4 Real Time Pothole Detection using Android Smartphones with Accelerometers

The paper describes a mobile sensing system for road irregularity detection using Android OS based smart-phones. Selected data processing algorithms are discussed and their evaluation presented with true positive rate as high as 90% using real world data.

However, the paper describes that ground truth data is collected for training classifiers through specialized hardware and not smart phones themselves. Furthermore, the smartphone when gathering data for testing the proposed algorithms was placed in a controlled position. We would like the user to do as they choose while the application gathers data to increase usability of the application and decrease cognitive load on the user. [10]

2.3 Goals

Based on the literature review, we aim to bring innovation to the previously described ideas by creating a mobile-cloud solution with the following features:

- It publishes all processed data online for the use of all and sundry
- It does not involve any user interaction and thus does not place any burden on the user. All they have to do is carry their phone around with them, just as they would do on any other day
- This application aggregates the data available from crowd sensing and this allows us to improve accuracy by taking the input of multiple users into account

- It does not involve using any additional piece of hardware to go with the application.
- It covers both speed breakers and potholes and also helps the user ascertain traffic congestion in the country.

Chapter 3

Functionality and Design

3.1 User Requirements

3.1.1 The Application

- The user shall be able to login to the app.
- Once logged in, the user can view maps for:
  - Traffic Congestion
  - Potholes
  - Speed breakers
- The user shall be able to turn off GPRS usage through app settings
- The app shall inform the user if their device does not support use of the application and will direct them to the website.
- The application shall provide an about page for the users to view to understand what the app does.
- The application shall gather, classify, store and upload data to the cloud service. This data shall be used there to describe the locations of potholes and speed breakers.

3.1.2 The Application Engine

- The application engine shall receive data from the users application and shall store it into the database

- The engine shall host the RESTful Web Services to get and post data from the users mobile device. The engine shall also support get calls from the website
- The server shall regularly process uploaded data to find out the locations of potholes and speed breakers.
- The server shall daily process uploaded data to find out what the free flow conditions are at particular road segments. These free flow conditions shall be used later on to discover if an area is congested or not with current traffic

3.1.3 The Website

- Displays the three maps, each showing:
  - Traffic Congestion
  - Potholes
  - Speed breakers
- Gives an overview of the application
- Provides the user with the link page for the app stores

3.1.4 The Database

The database shall contains tables that store:

- Congestion Data.

  This table shall contain all data that is uploaded by the users relevant to traffic conditions i.e. their locations and their speeds
- Free Flow Traffic

  This table shall contain all data that describes free flow conditions on roads i.e. the speed of vehicles when there is no congestion. An alternate could be speed limits
- Classified Data

  This table shall contain data that describes what each individual user has to say about individual locations. This is the result only of the processing on the app.

- Voting Results

  This table shall contain data that describes what the aggregated data says about each individual location after processing on the server. This data is then displayed on the website.

3.2 Functional Requirements

3.2.1 The Application

1. The user shall log into the app using Facebook or Google authentication as supported by Azure mobile services
2. The user shall be directed to the main page after logging in
3. The main page shall have three buttons that shall redirect the user to three separate maps:
   - Showing traffic congestion
   - Showing location of potholes
   - Showing location of speed breakers
4. The main page shall allow the user through the menu to view information about the app.
5. The main page shall allow the user to turn GPRS on or off at will. This provision is made so that if there is no WiFi available and the user does not want to expend their mobile broadband connection on the app, then they can choose to prevent the app from using the remainder of their bandwidth.
6. The application shall periodically check if the following are available, once the user has logged in, after every fifteen minutes:
   - An Internet connection (WiFi or GPRS if the user hasnt disallowed GPRS)
   - Google Play Services (For Locations API and Activity Recognition)
   - GPS Sensor, Rotation Sensor, Accelerometer
   - GPS Location is on
   - The cellphone is moving on the road in a vehicle

- The user is not using the phone for a telephone call
- If an SD Card is available, then there should be 100 MBs of space for the file that the app is writing into. If there is no SD card available, then only 50 MBs of space should be utilised. If there is already a file of such size available in memory, then data gathering, storage and processing should not take place.

If all of these conditions are true, the application can proceed to the step 7.

7. The application shall begin gathering data about road conditions and traffic congestion.

   - The application shall use the accelerometer sensor at the highest frequency in android with minimum delay i.e SENSOR_DELAY_FASTEST. This gives a frequency of 50 Hertz approximately.
   - The accelerometer readings must be scaled to a vertical axis. As the device rotates, the axis along which readings are taken also rotate. These must be scaled back to the original axis. (This can be done using the rotation sensor readings)
   - Once 40,000 accelerometer readings across all three axes are collected, calculate features from the data by dividing these readings into sets of 800 readings each. Each record of readings shall contain: (time in milliseconds, latitude, longitude, acceleration in the x direction, acceleration in the y axis direction, acceleration in the z axis direction)
   - Calculate the speed simultaneously for each record and add it to the record.
   - Pass this data across through a low pass filter and a high pass filter to generate three sets of data:
     - The first with only low frequency data
     - The second with only high frequency data
     - The original data set
   - Calculate features for each of these 800 readings. These features include:
     - Average speed
     - FFT of the accelerometer readings across the x axis (first 50)
     - FFT of the accelerometer readings across the y axis (first 50)

- FFT of the accelerometer readings across the z axis (first 50)
- FFT of the low pass-filtered accelerometer readings across the x axis (first 50)
- FFT of the low pass-filtered accelerometer readings across the y axis (first 50)
- FFT of the low pass-filtered accelerometer readings across the z axis (first 50)
- Maximum value from FFT of low pass filtered accelerometer readings across x-axis
- Maximum value from FFT of low pass filtered accelerometer readings across y-axis
- Maximum value from FFT of low pass filtered accelerometer readings across z-axis
- Minimum value from FFT of low pass filtered accelerometer readings across x-axis
- Minimum value from FFT of low pass filtered accelerometer readings across y-axis
- Minimum value from FFT of low pass filtered accelerometer readings across z-axis
- Maximum value from FFT of accelerometer readings across x-axis
- Maximum value from FFT of accelerometer readings across y-axis
- Maximum value from FFT of accelerometer readings across z-axis
- Minimum value from FFT of accelerometer readings across x-axis
- Minimum value from FFT of accelerometer readings across y-axis
- Minimum value from FFT of accelerometer readings across z-axis
- Standard deviation of accelerometer readings across x axis
- Standard deviation of accelerometer readings across y axis
- Standard deviation of accelerometer readings across z axis
- Mean of accelerometer readings across x axis
- Mean of accelerometer readings across y axis
- Mean of accelerometer readings across z axis

- Standard deviation of low pass filtered accelerometer readings across x axis
- Standard deviation of low pass filtered accelerometer readings across y axis
- Standard deviation of low pass filtered accelerometer readings across z axis
- Mean of low pass filtered accelerometer readings across x axis
- Mean of low pass filtered accelerometer readings across y axis
- Mean of low pass filtered accelerometer readings across z axis
- Number of points one standard deviation away from the mean on the x axis
- Number of points two standard deviation away from the mean on the x axis
- Number of points one standard deviation away from the mean on the y axis
- Number of points two standard deviation away from the mean on the y axis
- Number of points one standard deviation away from the mean on the z axis
- Number of points two standard deviation away from the mean on the z axis
- Number of points one low pass filtered standard deviation away from the low pass filtered mean on the x axis
- Number of points two low pass filtered standard deviation away from the low pass filtered mean on the x axis
- Number of points one low pass filtered standard deviation away from the low pass filtered mean on the y axis
- Number of points two low pass filtered standard deviation away from the low pass filtered mean on the y axis
- Number of points one low pass filtered standard deviation away from the low pass filtered mean on the z axis
- Number of points two low pass filtered standard deviation away from the low pass filtered mean on the z axis
- Number of points one high pass filtered standard deviation away from the low pass filtered mean on the x axis
- Number of points two high pass filtered standard deviation away from the low pass filtered mean on the x axis

  - Number of points one high pass filtered standard deviation away from the low pass filtered mean on the y axis
  - Number of points two high pass filtered standard deviation away from the low pass filtered mean on the y axis
  - Number of points one high pass filtered standard deviation away from the low pass filtered mean on the z axis
  - Number of points two high pass filtered standard deviation away from the low pass filtered mean on the z axis
  - Maximum on X axis
  - Maximum on Y axis
  - Maximum on Z axis
  - Minimum on X axis
  - Minimum on Y axis
  - Minimum on Z axis
  - Minimum Speed
  - Maximum Speed
  - Maximum of ratio of x to z in high pass filtered data
  - Mean of ratio of x to z in high pass filtered data
  - Median of ratio of x to z in high pass filtered data
  - Minimum of ratio of x to z in high pass filtered data
  - Standard deviation of high pass filtered data across x axis
  - Standard deviation of high pass filtered data across y axis
  - Standard deviation of high pass filtered data across z axis
  - Maximum of high pass filtered data across x axis
  - Maximum of high pass filtered data across y axis
  - Maximum of high pass filtered data across z axis
  - Minimum of high pass filtered data across x axis
  - Minimum of high pass filtered data across y axis
  - Minimum of high pass filtered data across z axis
- This data shall be passed to the classifier (an SMO trained model) and is classified as either a pothole, or a speed breaker or as no anomaly
- This classified data should be written to file
- As speed, location and time values are available, they should be passed to the mobile service for storing in the database for calculation of congestion of traffic

8. The application shall periodically upload data stored during gathering information about road conditions to the server if WiFi is available.

3.2.2 The Website

The website shall have two main functionalities:

- The website shall have an about page, explaining how the app works, how to download it and who are the creators.
- The website shall display three separate heat maps for showing:
  - Potholes (data from voting results)
  - Speed breakers (data from voting results)
  - Congestion (Live) (data from congestion data)
- An option that could also be additionally implemented would be to allow the user to choose a certain time for which they would like to see the congestion in the city.

3.2.3 The Azure Mobile Service

The mobile service provides two essential functionalities:

- Passes data from the app to the database. An optimal implementation would include batch processing the input from the user and then inserting it into the database.
- The service provides RESTful APIs to allow easy access to the data for display on the website and on the maps in the device. APIs can be customised for any use.
- The service also provides authentication through social media websites.

The following services must be implemented:

3.2.4 The App Engine

The application engine shall consist of a server that shall periodically run the following processes:

- Delete all old processed data from classified data to make room for more data. The database size is limited and more space should be made as the volume of data increases with the number of users.

Table Name	Table Description	Services
Classified Data	Used to store data received from the applications classifier	Get, Post
Voting Results	Server gets data from classified data, processes it and pushes results back to voting results	Get, Post
Congestion Data	Stores data containing user id, location, speed and time to calculate congestion	Get, Post, Get within a specified time for a specific location with unique users only
Free Flow	Contains free flow speed for each road segment	Get, Put, Post

Table 3.1: RESTful Services per Table

- Get data from classified data table, process it and push result back into voting results. The processing would involve:
  - Get data for a particular cell across the map
  - See whether there are more votes for that cell having a pothole, a speed breaker or no anomaly. This is the defined result. In the case where votes for two features (potholes—speed breakers, no anomaly) are equal, then we choose the safe option and say that there are no anomalies in that cell. If the votes for potholes and speed breakers are equal, then the decision becomes more difficult to make and due to that, we must consider implemented Naive Bayes at the server end to make the choice easier.

3.3 Database Design

Figure 3.1 shows the database design. The version numbers in the congestion data and classified data tables are significant as they help us keep track of the version of the application that uploaded the data to the database. As the application is updated, we need a method that would allow us to keep track of what data came from what version of the application so that each of the data types could be updated using their respective algorithms.

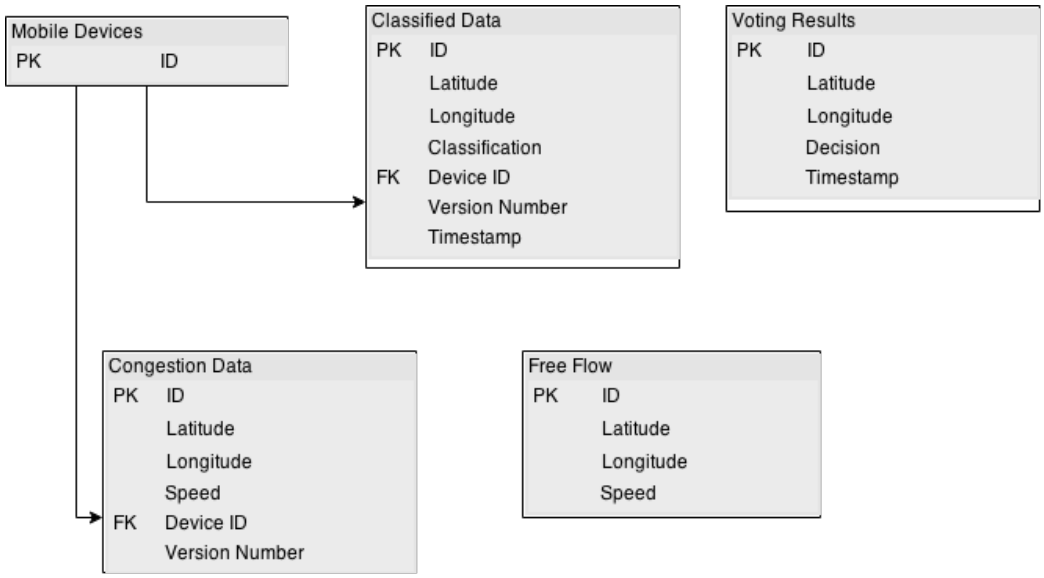

Figure 3.1: Database Design

3.4 Data Flow Diagram

Figure 3.2 is a simplified diagram showing how data flows within the system to complete functionality:

3.5 High Level System Architecture

Figure 3.3 is a very high level architectural diagram of the system. This is broken down according to the steps involved in the processing in the system.

1. The Application Gathers, Classifies, Stores and Uploads Data to the Server

   The mobile app runs in the background when the user logs in. While the user is traveling in a car, the app tracks the phones location as well as collecting raw data from the phones accelerometer sensors. This data is used to determine whether or not the phone is traveling on a road. If it is, the app determines the phones orientation and record linear acceleration / shocks along the vertical direction.

   The data collection algorithm running in the app may be understood as follows: 1) Is this phone on a road? If No, wait, then go back to 1). If Yes go to 2). 2) Is this phone in a moving vehicle? If No, wait, then go back to 1). If yes, go to 3). 3) Turn on sensors, log data. Go to 4). 4)

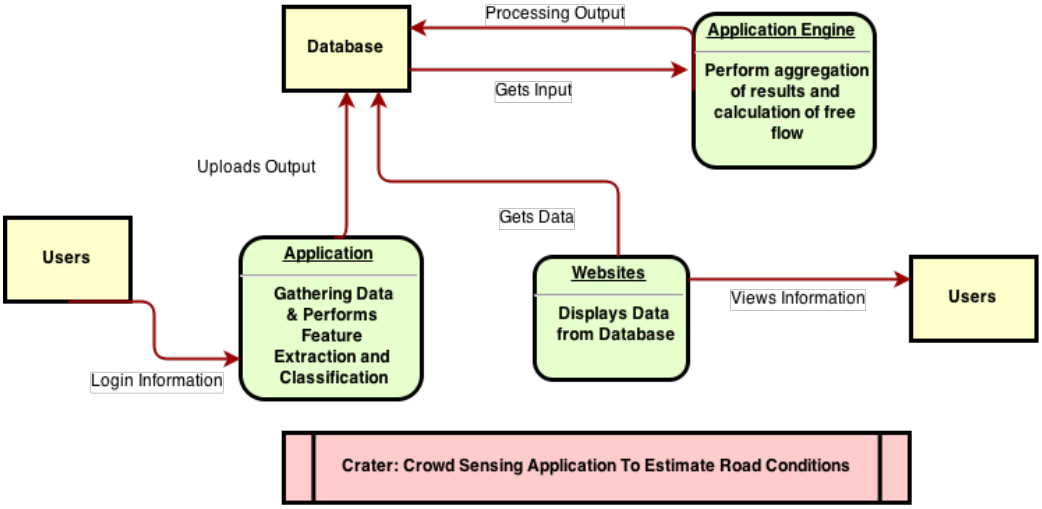

Figure 3.2: Data Flow Diagram

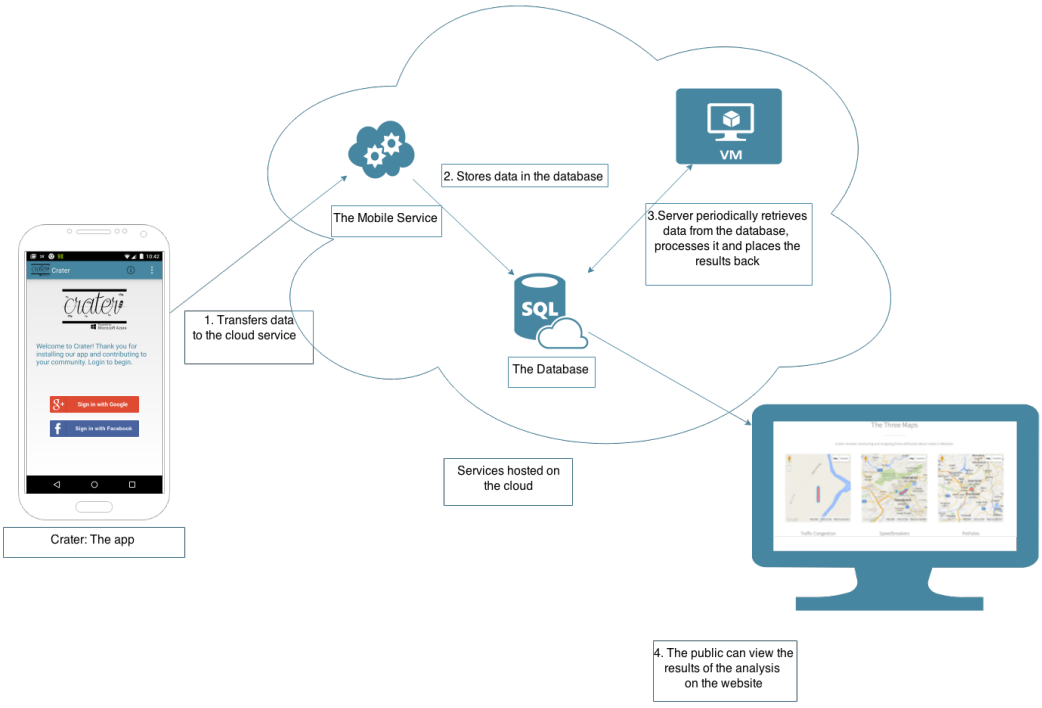

Figure 3.3: Architectural Diagram

Is the phone still on the road AND in a vehicle? If Yes, continue to log data and go to 4). If No, turn off sensors, go to 1). This yields a time series signal of the vertical acceleration the phone experiences. This signal is processed to: 1) Detect potholes and estimate the location. 2) Detect speed bumps and estimate their location. 3) Estimate the cars speed and use it to measure traffic conditions.

The mobile app transmits detected potholes and speed bumps, along with speed along the road at various points either via GPRS / 3G / 4G mobile Internet connection (if available) or through the next available WiFi connection, whenever it becomes available.

The app is allocated a finite amount of memory to buffer data that needs to be transmitted to the server. Once the buffer is full, the app may stop collecting data until the buffer is emptied. Once data has been sent to the main server, it is removed from the users device.

2. The Mobile Service Stores Data into the SQL Database

   The SQL database contains data about new mobile app measurements awaiting processing since the last run of the app engine. The app engine uses this data when it executes and updates road condition consensus, speed bump locations, pothole locations and the limited backlog of raw measurements

3. Server Periodically Retrieves Data, Processes It and Uploads it Back into the Database

   The principal task of the app engine is the periodic analysis of classified data received from mobile app instances. Every time the app engine executes it includes the new data received since the last run for the following two detectors:

   - Pothole detector.
   - Speed bump detector

   For each location on the map, the app engine periodically analyzes the data available, filters out false positives, and detects the consensus value that describes the speed bump / pothole location and road condition. The greater the number of readings for a location, the more accurate the mean value is likely to be. The updated results of the detectors is plotted on the Google Maps interface. An additional feature is to include a map view of the results of the app engines two detectors in a view of the mobile app described above. Therefore, they could see the

updates without launching the browser which could serve as additional motivation for users to install the mobile app.

4. A User can View the Results from the Website.

   The website provides a map interface overlaid with information that represents the consensus of user reported road conditions. It also maps the locations of potholes and speed bumps and the current traffic conditions. Traffic conditions are displayed as a function of the number of users at a location and the ratio of free flow conditions to the users current speeds. Each individual user plotted on the map has weight:

$$Weight = Free\ Flow\ Conditions / Current\ Speed\ of\ User \tag{3.1}$$

   Free flow conditions are the conditions across a road when there is no congestion. Ideally, this value should be replaced by the speed limits however speed limits are not available. Therefore, the free flow conditions have a default value of 60 km/hr if no values are available for a road segment.

   Free flow conditions are calculated as the median of the speed values between 1 and 5 AM. We assume that there is no congestion at this time on the roads and that the people will be travelling at speeds close to the speed limit. When we compare the current speed of the user with the free flow, we get their individual weights - so a person who is moving slowly will have more weight than a person moving at a speed close to free flow conditions. Users with more weight are shown with a more intense red colour on the map while users moving fast are shown with a light blue colour.

3.6 Important Design Decisions

3.6.1 Google Location APIs vs GPS Sensors in Android Application : Accuracy and Activity Recognition

During development, the initial method of obtaining the geographical location of the users was to simply use the GPS sensors that are available in the device. However, during testing, it was discovered that the sensors used did not always give the most accurate values of the users location, as indicated

by the accuracy field that the function result also provides. These returned values were subsequently plotted onto Google Maps by a user who knew where they had been while using the application and sometimes, the latitude and longitude pair returned by android sensors was off by several streets.

Such inaccuracy was unacceptable for us because there would be no point in finding the location of potholes and speed breakers if the location was off by a couple of streets. Therefore, we set out in search of a method that would give us better accuracy in terms of location. Since android is a product by Google, a solution was already provided to by them in the form of Google Location APIs. (https://developer.android.com/google/play-services/location.html). The website claims that its fused location provider has the following features: "The Fused Location Provider intelligently manages the underlying location technology and gives you the best location according to your needs.

- Simple APIs: Lets you specify high-level needs like "high accuracy" or "low power", instead of having to worry about location providers.
- Immediately available: Gives your apps immediate access to the best, most recent location.
- Power-efficiency: Minimizes your app's use of power. Based on all incoming location requests and available sensors, fused location provider chooses the most efficient way to meet those needs.
- Versatility: Meets a wide range of needs, from foreground uses that need highly accurate location to background uses that need periodic location updates with negligible power impact."

After rigorous testing, it was found that the service does provide much better accuracy than the sensors alone. The worst case involved the readings to be in a few meters distance of the actual location of the user. Therefore, it was decided that the APIs should be used.

This decision was corroborated by the fact that the APIs provide functionality to discover what the users current activity is. The web page describes this as:

"**Activity recognition:**

The Activity recognition API makes it easy to check the users current activitystill, walking, cycling, and in-vehiclewith very efficient use of the battery:

- Optimized for battery: Uses low-power sensors to recognize the user's current physical activity.

- Enhances other services with context: Great for adding movement awareness to location awareness. Apps can adjust the amount of location awareness they provide, based on the current user movement. For example, a navigation app can request more frequent updates when the user is driving.

- Advanced activity detection: For apps that want to do their own post-processing, the activity APIs provide confidence values for each of the activities. It also includes two activities that indicate unreliable measurements: unknown and tilt.

”

Since code reuse is always a good idea, this API has been used to discover the users activity as well. The alternate would be to use another classifier to recognise the users activity. Since this work did not take priority over classifying the road conditions, this idea was discarded.

3.6.2 SMO: In the application or at the server end

During the initial phases of the architectural design, the plan was to place the classifier at the server instead of in the app.

- Plan 1: Send Gathered Accelerometer Data to Server

  This turned out to be a poor design decision in practice because the rate at which data is gathered is 50 Hertz. This means that for each second, we have 50 records containing (timestamp, latitude, longitude,acceleration in the x direction, acceleration in the y direction, and acceleration in the z direction). This amounted to a lot of space that the application occupied in memory. Therefore, a method had to be discovered in order to make this process easier.

- Plan 2: Send features only to the server

  Approximately 350 features (all doubles) are calculated for every 800 readings/for 16 seconds of data. This allowed us to go from storing 4800 readings for every 16 seconds in Plan 1 to storing 350 readings for every 16 seconds. However, during an experimental test, this also occupied quite a lot of space on the device and consumed a lot of bandwidth when uploading the data to the server end and . Therefore, we moved to plan 3.

- Plan 3: Moving the classifier and feature extraction to the app

Once the low pass filtering, and classification portions are shifted the app, then the downside is that the application becomes large in size. On the upside however, is that for every 1600 seconds, only the following records are stored in file: (timestamp, latitude, longitude, classification, device id). This significantly decreases the amount of data that is stored on file and also helps save the users bandwidth.

Chapter 4

Implementation and Results

4.1 Software Development Life Cycle and Challenges Encountered

Our software development life cycle consisted essentially of moving from requirement gathering to deployment while going through several iterations during each stage in order to find and remove bugs right there and then, and ensure that the system was well-integrated with the rest of its components.

4.1.1 Planning and Requirement Gathering

During this stage of development, we went through defining the scope of our application, the functionalities we would implement and the constraints that we would have to face while developing the system.

4.1.2 Building the Ground Truth Application

The ground truth application was created only so that we could get data that would be used to train our classifiers, and would test different classifiers to see which one gave the best results. Its functionality included the following:

1. The app gathered sensor data in the background the moment it was switched on.

2. The user interface must provide two buttons. The first one would be for potholes and the second one would be for speed breakers. At the button press, the timestamp and location of the user would be stored in a buffer.

These two functionalities allowed us to have three arrays of information at the end of the day:

- Sensor Data
- When and where was a pothole encountered
- When and where was a speed breaker encountered

By taking the timestamp from when a pothole/speed breaker was encountered and finding out what the sensor data looked at the time, we could begin to understand what features we might use to classify potholes and speed breakers.

4.1.3 Gathering Labelled Data

Ground truth gathering involved some field work - moving over the city in a car with the app ready, pressing the buttons on the interface to mark when and where a pothole/speed breaker was encountered.

A picture of the pothole leading into NUST H-12 is shown above.

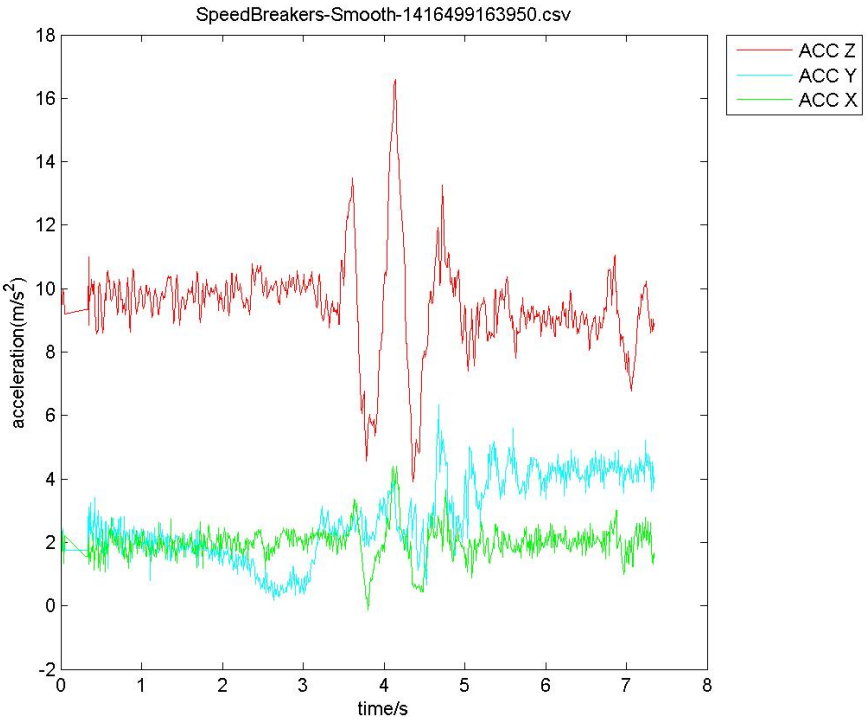

Figure 4.1: A Plot of Raw Accelerometer Readings of A Speed Breaker

4.1.4 What do Potholes and Speed Breakers Look Like...On Paper?

Once we had the data describing a pothole and a speed breaker available to us, we decided to plot this data to find out what potholes and speed breakers look like on paper.

This data was first plotted through matlab. Figures 4.1-4.4 indicate how raw values of acceleration along the x,y and z axis look when plotted with respect to time.

A low pass filter was applied to smooth the curve. A high pass filter was applied to see exactly what very high frequency values were obtained which could be used to describe general road conditions (now no longer in the scope of the project).

After looking at the smoothed curves obtained from acceleration in all three directions, we extracted the features that are described in the requirements section in detail. We used these features to classify potholes, speed breakers and no road anomaly across the road. Feature extraction was done

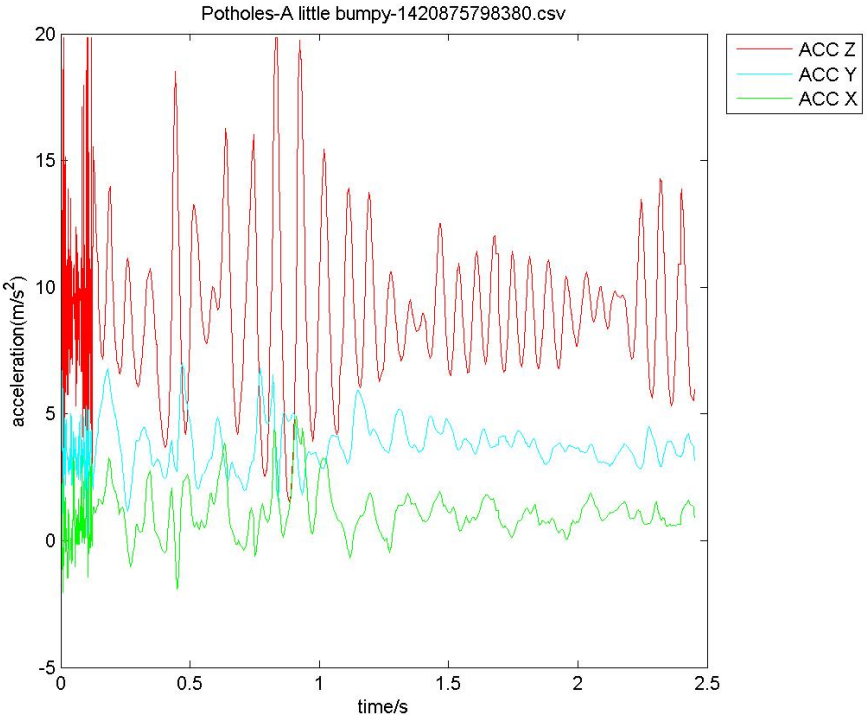

Figure 4.2: A Plot of Raw Accelerometer Readings of A Pothole

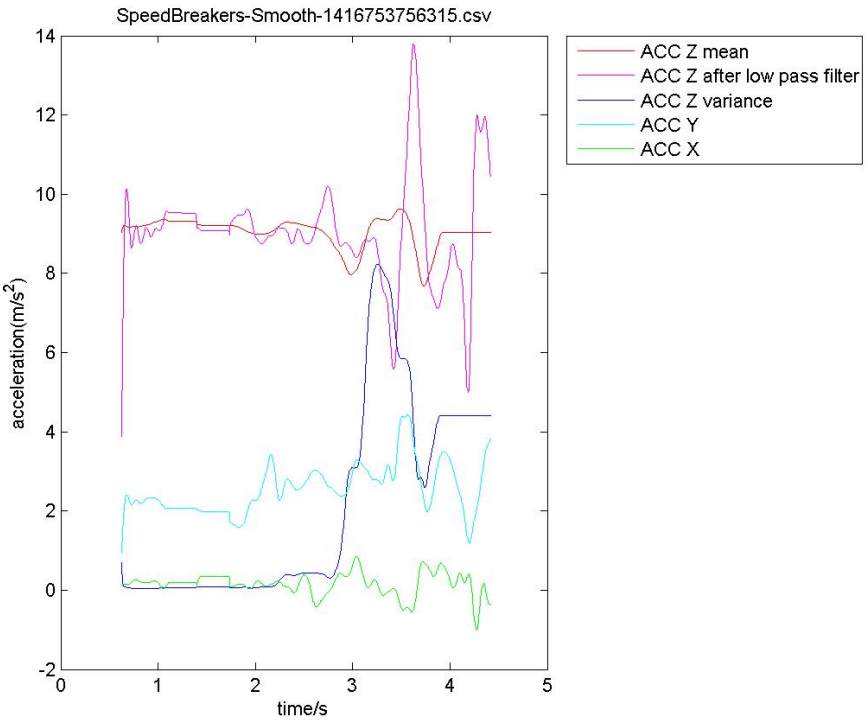

Figure 4.3: A Plot of low-Pass Filtered Accelerometer Readings of A Speed Breaker

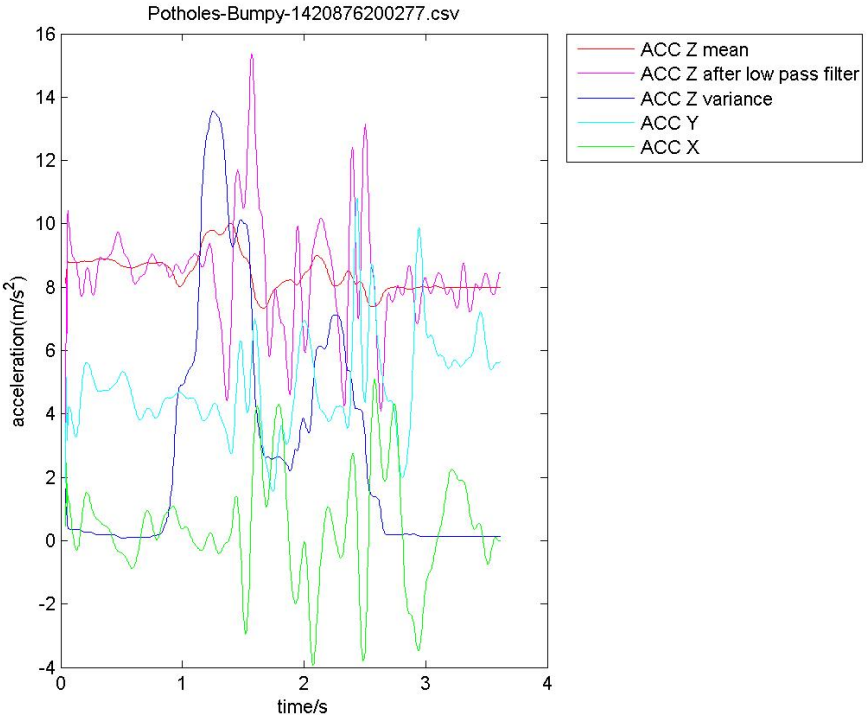

Figure 4.4: A Plot of Low-Pass Filtered Accelerometer Readings of A Pothole

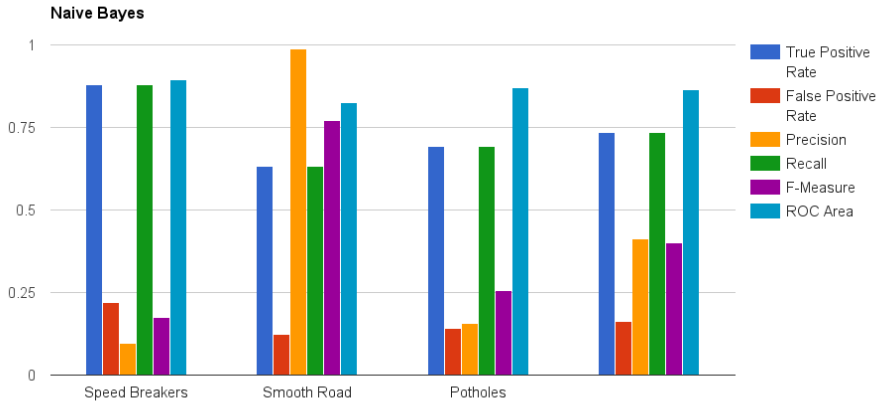

Figure 4.5: Naive Bayes

using Python scripts which made the job a whole lot easier.

4.1.5 Testing Classifiers for Best Performance

Using Weka, we tested several classifiers, given our ground truth data.

The distribution of labelled ground truth data is given in Table 4.1.

Labelled Potholes	Labelled Speed Breakers	Points Not Labelled	Total
254	177	6446	6877

Table 4.1: Distribution of Labelled Data

We trained five classifiers on the labelled data sets and performed cross-validation to test them. The performance of each classifier would help us choose which classifier to deploy in the application. The results of these classifiers are shown in Figures 4.5-4.9

After looking at the results, we decided to use SMO as our classifier of choice as it gave the gave the best Precision (a measure of the quality of results) and F-measure, and the second best Recall and ROC Area. Although there is a high false positive rate for "smooth road", we believe that using crowd sensing would allow us to compensate for this error by processing aggregate data from multiple users.

Sequential minimal optimization (SMO) is an algorithm for solving the quadratic programming (QP) problem that arises during the training of sup-

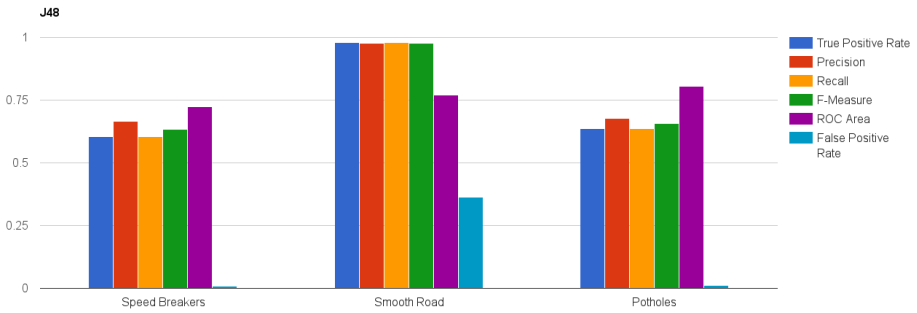

Figure 4.6: J48

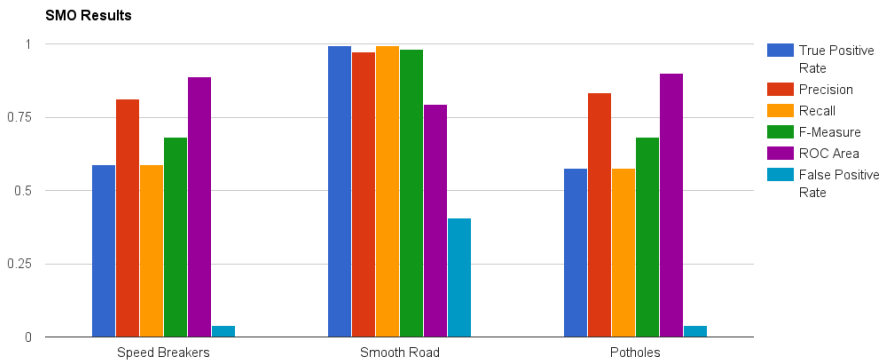

Figure 4.7: SMO

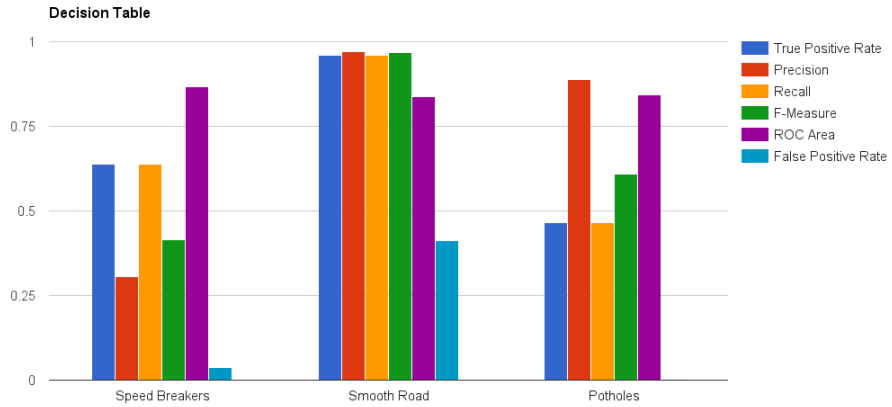

Figure 4.8: Decision Tables

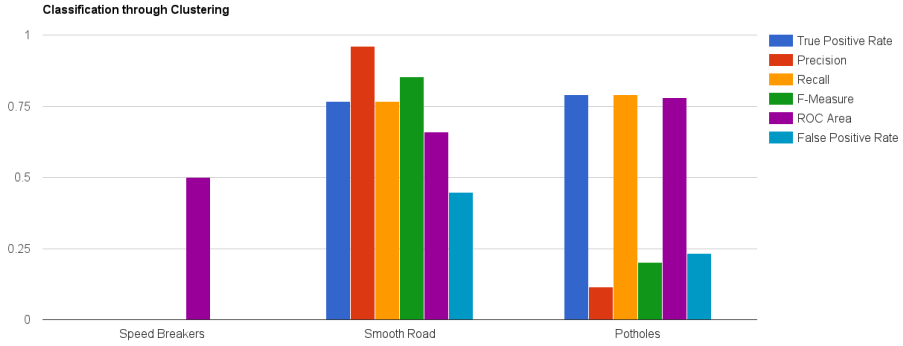

Figure 4.9: Classification via Clustering

port vector machines. SMO is widely used for training support vector machines and is implemented by the popular LIBSVM tool. [13] In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. [14]

4.1.6 Building the Final Apps, the Website and the Back-end

Once, the model was decided, all that remained was implementing the applications. This was done through an iterative model. Functionalities were added incrementally to the application, thoroughly tested and then we would move on to the next set of functionalities.

Building the website involved some work to implement the display of traffic conditions in Pakistan, especially the section where the algorithm for assigning weights to each user was developed.

As we were working on the applications, we found that each platform supported different functionalities. For example, no services were available on windows phones and iOS to detect the activity of the user. Therefore, we had to make do with whatever functionalities were available to us so that we could achieve our main goal: building and running a classifier to detect road conditions in Pakistan.

4.1.7 Testing and Deployment

Finally, once the system was complete and running, we tested the system as a whole together by taking readings along roads in Islamabad. A few bugs related to integration of the project came up but were quickly solved. Uploading the android app onto Google Play also gave us some trouble due to warnings in some external jars namely weka.jar. However, this is a link to the website that was created at the end [8]

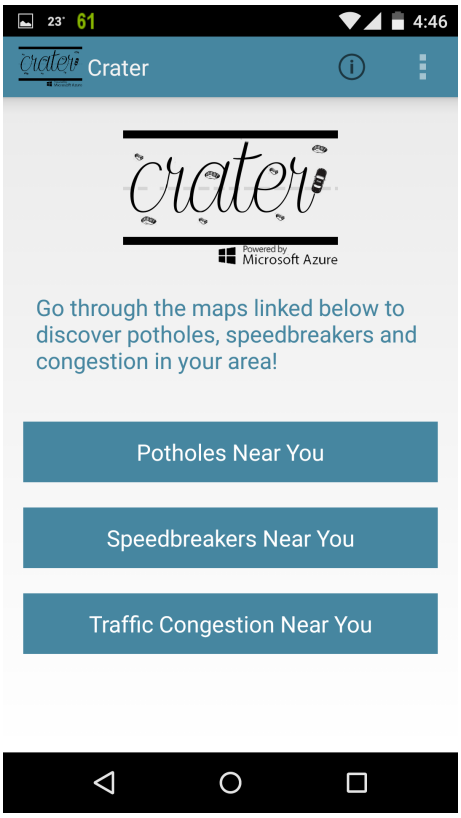

Figure 4.10: Home Page of the App

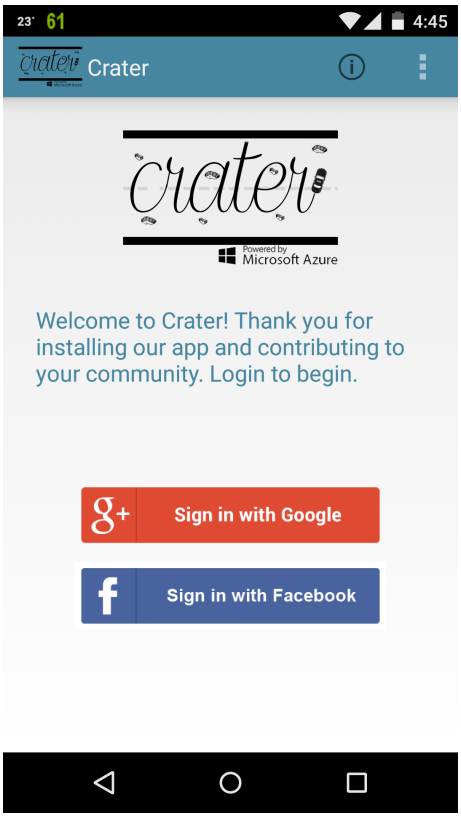

Figure 4.11: Login Page of the App

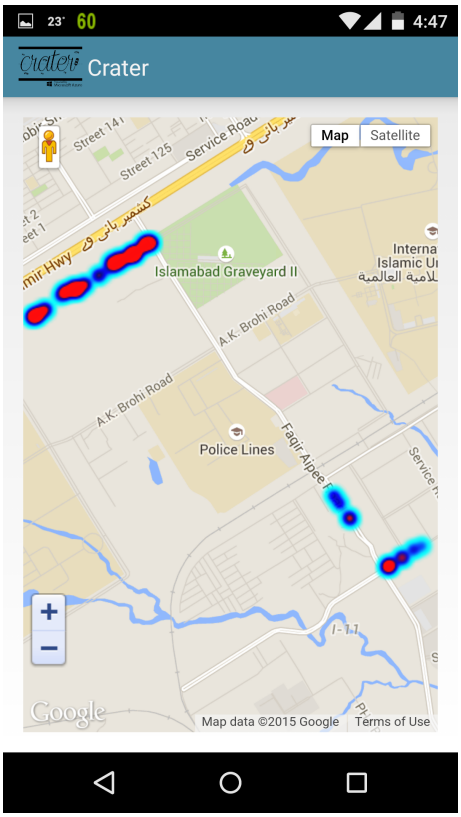

Figure 4.12: A Map from the App

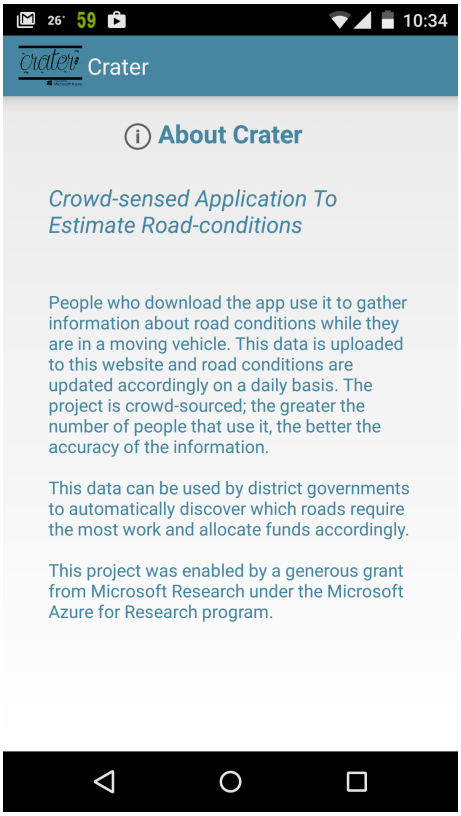

Figure 4.13: About Page from the App

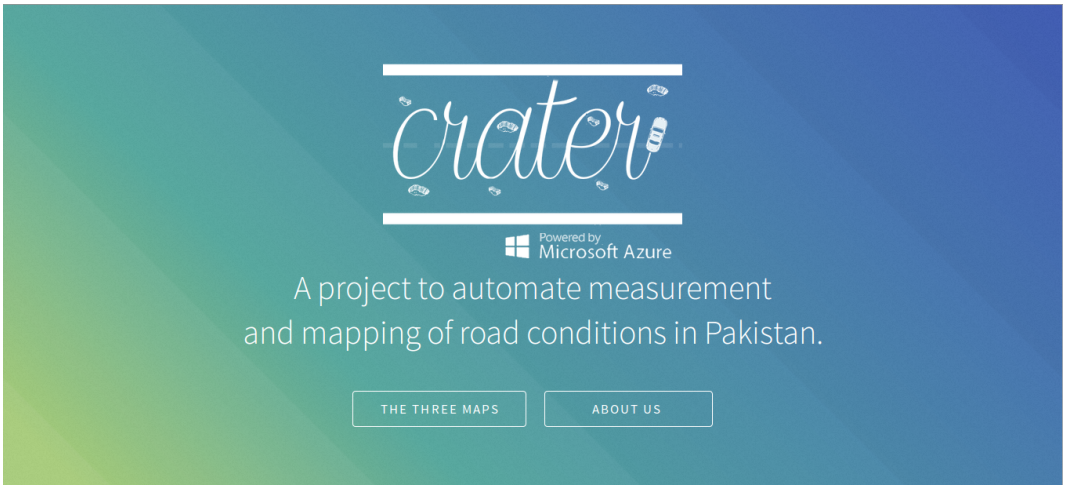

Figure 4.14: Main Page from the Website

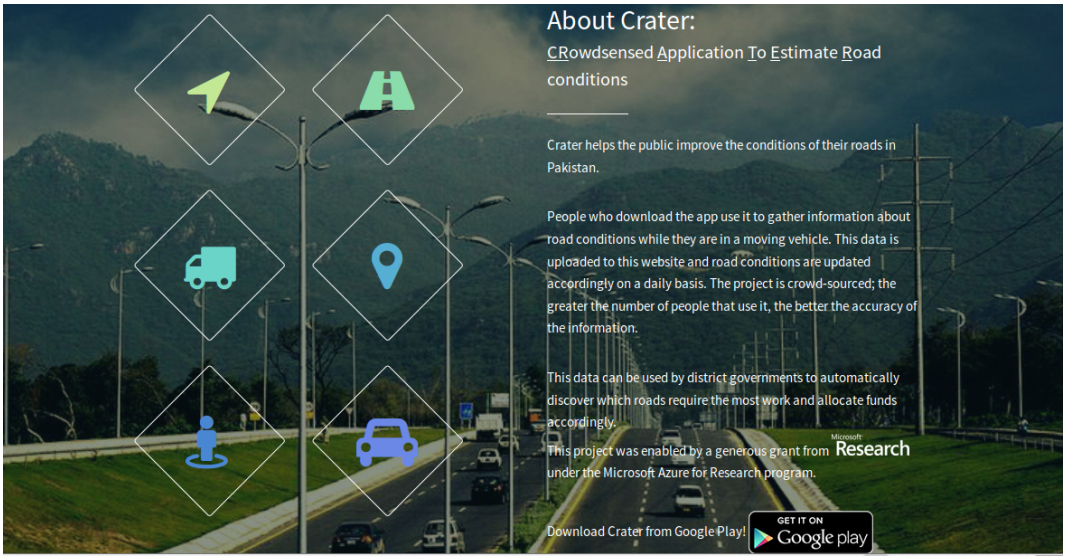

Figure 4.15: About Page from the Website

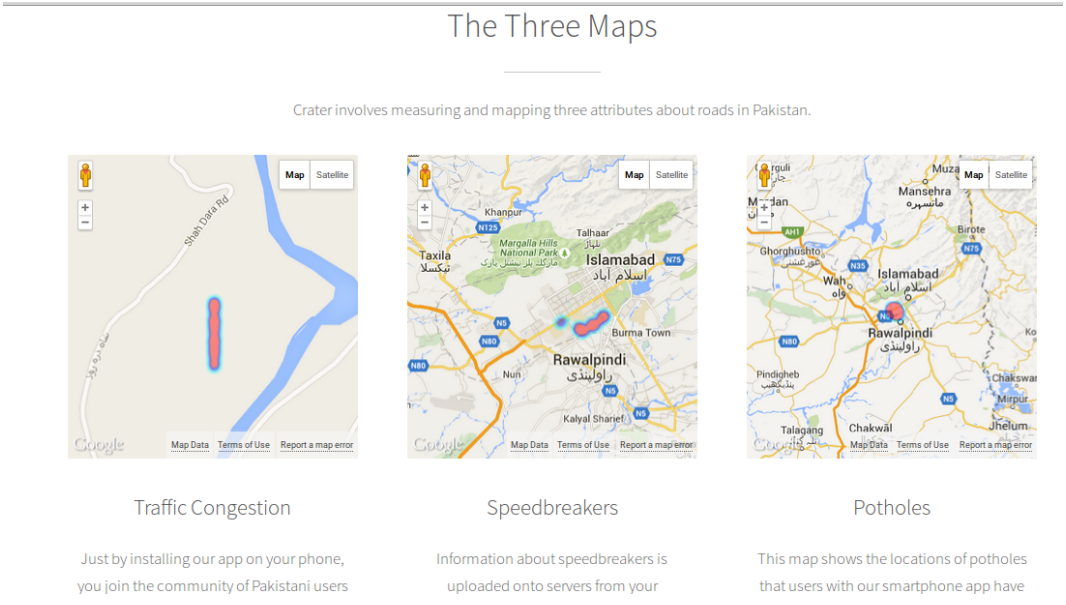

Figure 4.16: Maps Page from the Website

4.1.8 Implementation Details

4.1.8.1 Dividing the Map

In order to simplify the way the maps are viewed on the website, we divided the map into 1 meter square cells. Therefore, whenever a user would want to view the map for potholes—speed breakers, the following things would happen:

- The bounds (North east corner and the south west corner of the map) would be sent to the server.
- At the server, a query would be applied to retrieve all the data that fits within these bounds from the table Voting Results.
- These are displayed in the form of a heat map on the website.

However, things get a little complicated when displaying data for congestion. Challenges include:

- Displaying each user just once on the map even though there are multiple rows of data for each user.
- Each road segment can have a different free flow speed. Therefore, the drivers weight must be displayed with respect to that particular free flow speed.

The first problem is solved by grouping the users by their device IDs and then selecting the reading that is posted at the latest time. For the second problem, the map of Islamabad was divided into 1 meter square sections using Google Places API. Then, each section was tested if it was a road or not. if it was a road segment, then that section's latitude and longitude were written to file. These values were later uploaded to the server with null free flow values. The free flow values would be calculated as soon as readings from the users start coming in using a server-side script. These values would be the median of the speeds between 1 and 5 AM. If a value for a free flow of a road condition already exists in the table, then this value would be updated by the formula. Consult Figure 4.17.

4.1.8.2 Calculation of Speed

While the Google Location API is fantastic for finding the user's activity and the location, it does not give the device's speed. Therefore speed is calculated by first finding out the distance between a user's consecutive locations and dividing this value by the differences in time at those two locations.

Distance is found by using the Haversine formula. See Figure 4.18.

$$New\ Free\ Flow = (Old\ Free\ Flow + median\ \{speeds\ between\ 1:00 - 5:00\ AM\})/2$$

Figure 4.17: Calculate Free Flow Conditions

Haversine formula:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$
$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$$
$$d = R \cdot c$$

where φ *is latitude,* λ *is longitude,* R *is earth's radius (mean radius = 6,371km); note that angles need to be in radians to pass to trig functions!*

JavaScript:

```
var R = 6371000; // metres
var φ1 = lat1.toRadians();
var φ2 = lat2.toRadians();
var Δφ = (lat2-lat1).toRadians();
var Δλ = (lon2-lon1).toRadians();

var a = Math.sin(Δφ/2) * Math.sin(Δφ/2) +
        Math.cos(φ1) * Math.cos(φ2) *
        Math.sin(Δλ/2) * Math.sin(Δλ/2);
var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

var d = R * c;
```

Figure 4.18: Convert Difference in Latitude and Longitude to Distance

The accelerometer sensor returns the acceleration of the device. This is a vector in 3 dimentional space. This vector is returned in the device coordinate system. What you want is the coordinate of this vector in the world coordinate, which is simply

```
R = rotation matrix obtained by calling getRotationMatrix
A_D = accelerator vector return by sensor ( A_D = event.values.clone )
A_W = R * A_D is the same acceleration vector in the world coordinate system.

A_W is an array of dimention 3
A_W[0] is acceleration due east.
A_W[1] is acceleration due north.
```

Here is some code to compute it (assumes `gravity` and `magnetic` contain output from their respective sensors):

```
float[] R = new float[9];
float[] I = new float[9];
SensorManager.getRotationMatrix(R, I, gravity, magnetic);
float [] A_D = values.clone();
float [] A_W = new float[3];
A_W[0] = R[0] * A_D[0] + R[1] * A_D[1] + R[2] * A_D[2];
A_W[1] = R[3] * A_D[0] + R[4] * A_D[1] + R[5] * A_D[2];
A_W[2] = R[6] * A_D[0] + R[7] * A_D[1] + R[8] * A_D[2];
```

Figure 4.19: Correct the Device's Orientation

4.1.8.3 Fixing the 3D axis

One of our essential features include the fact that the device does not have to remain still during data gathering. However, the readings of acceleration change with rotation of the device as the axis for acceleration is fixed [2]

Therefore, we must get the accelerometer readings along the real-world 3D axis. In order to do this, we employ readings from the rotation matrix that is provided by android. Consult code snippet in Figure 4.19. [12]

Chapter 5

Conclusion and Future Discussion

5.1 Findings and Significance

Our finding show that we have an approximately 60% true positive rate for identifying potholes and speed breakers and a 97% true positive rate for identifying that a particular road segment has no anomaly. This is quite a significant contribution as the device is not held still and can be moved, rotated or used to play games.

Studies "Researchers at the Texas A&M Transportation Institute estimate that, unless spending increases, congestion and rough roads will cost the average Texas household $6,100 a year in wasted fuel, vehicle repairs, and time lost sitting in traffic between now and 2035." [15] However, there is still some room for improvement, as described in the section "Improvements".

Traffic conditions have never been mapped before in Pakistan. Although Google Maps does provide an overlay of traffic, there is no information available for Pakistan. This feature of the application will allow users to choose routes with minimum traffic so that they can reduce the amount of fuel that is wasted while sitting idly in traffic, and save their time.

The greater the number of people that use the application, the greater accuracy that we will achieve. Once the application becomes commonly used, our results will be sufficiently accurate to allow the government to make decisions about allocating funds in proportion to how terrible the road conditions are at a particular road segment or how badly congested the area gets and is in need of alternate routes. The public shall be able to use this information to keep an eye on how well the government authorities are doing and if they are actually fulfilling their promises they make during elections.

Since we are an objective third party, our results would be reliable and the concern that they are conditions are shown much better than they actually are would easily be assuaged.

5.2 Commercialization

This project can be easily commercialized by keeping government agencies and the general public in mind as target audiences. The government can sponsor the application as they need the information this application provides to properly plan and allocate funds for road maintenance in an efficacious manner. By adding ads to the application or charging a small amount of money for installing the application, funds can be generated from the public and these can be put into additional research for improving our classification algorithm.

Companies such as TCS whose drivers have to travel long distances also make for good target customers as they can make productive use of the traffic congestion information to take the least congested route to their destination. This benefits the overall system; as they move over the road network, the automatically map the road sections by simply keeping an installation of the smart phone application in their mobile devices.

5.3 Limitations

The limitations of the project are tied to the improvements suggested in the next section. Most significantly, the application treats all potholes and speed breakers as one class each. This may limit the accuracy that we can achieve as for example, potholes vary in size and shape from tiny, round ones to large misshapen ones. Therefore, there are several types of potholes—speed breakers in one class and this makes classification increasingly difficult.

Furthermore, our greatest current limitation is the fact that we have a very limited number of users. This application is crowd sensing; it needs to be supported by a crowd to improve accuracy of information and increase coverage of more and more locations in Pakistan.

5.4 Improvements

In the next phase of the project, classification can be done by dividing types of road anomalies into several different categories to have much more discrete classes. For example, some factors to keep in mind are:

- Potholes and speed breakers are of different sizes and are likely to give different results.
- Different roads can alter the pattern of potholes. For instance, a gravelly road with a pothole will give different readings than a smooth road with a pothole.
- Different vehicles have different shock absorbers and this can affect the amplitude of the wave form produced.
- Phones have sensors of varying quality and this factor must also be accounted for.

An option that must be explored involved mapping and measuring the general road conditions of the roads e.g is the road smooth or gravelly or is it unpaved? Answering this question would complete the overall picture that describes road conditions in Pakistan.

Bibliography

[1] Waze. www.waze.com.

[2] Tegra android accelerometer whitepaper, 2010. http://developer.download.nvidia.com/tegra/docs/tegra_android_accelerometer_v5f.pdf.

[3] Connected Bits, 2014. www.streetbump.org/.

[4] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 29–39. ACM, 2008.

[5] Fix It Find It, 2015. http://www.seattle.gov/customerservice-bureau/find-it-fix-it-mobile-app.

[6] Fill That Hole, 2015. http://www.fillthathole.org.uk/iphone.

[7] Pothole Hunters, 2015. https://play.google.com/store/apps/details?id=cz.vymoly.android hl=en.

[8] Faria Kalim. Crater, 2015. http://craters.azurewebsites.net.

[9] Aniket Kulkarni, Nitish Mhalgi, Dr Sagar Gurnani, and Nupur Giri. Pothole detection system using machine learning on android.

[10] Artis Mednis, Girts Strazdins, Reinholds Zviedris, Georgijs Kanonirs, and Leo Selavo. Real time pothole detection using android smartphones with accelerometers. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–6. IEEE, 2011.

[11] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.

[12] Stackoverflow. Calculate acceleration in reference to true north, 2013. http://stackoverflow.com/questions/14963190/calculate-acceleration-in-reference-to-true-north/14988559.

[13] Wikipedia. Sequential minimal optimization — wikipedia, the free encyclopedia, 2014. [Online; accessed 15-May-2015].

[14] Wikipedia. Support vector machine — wikipedia, the free encyclopedia, 2015. [Online; accessed 15-May-2015].

[15] Zhanmin Zhang, Michael Murphy, Robert Harrison, Jose Weissmann, Tim Lomax, David Schrank, Seokho Chi, Randy Machemehl, Khali Persad, David Ellis, et al. Its about time: investing in transportation to keep texas economically competitive: Appendices. 2011.