IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 30, 2016, accepted September 1, 2016, date of publication September 8, 2016, date of current version December 8, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2607719

CRATER: A Crowd Sensing Application to Estimate Road Conditions

FARIA KALIM[1], JAEHOON (PAUL) JEONG[2], AND MUHAMMAD U. ILYAS[3], (Senior Member, IEEE)
[1]Department of Computer Science, University of Illinois at Urbana–Champaign, Champaign, IL 61801, USA
[2]Sungkyunkwan University, Suwon 110-745, South Korea
[3]Department of Electrical Engineering, School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan

Corresponding author: M. U. Ilyas (usman.ilyas@seecs.edu.pk)

ABSTRACT We developed a crowd sensing application to estimate road conditions (CRATER). CRATER is a smartphone application that opportunistically measures acceleration when it finds itself on the road in order to map and measure the locations of potholes and speedbumps. It does not require input from users, but can report detected potholes and speedbumps to a cloud-hosted application engine, which stores partially processed data received from smartphones of participating users and jointly processes it to obtain a better estimate of road conditions. The information is published in map form on the web. This map allows both citizens and municipal authorities to localize potholes, road segments in need of repair, and imbalances in infrastructure maintenance efforts across cities. Road tests demonstrate that CRATER succeeds in correctly detecting roughly 90% of potholes and 95% of speedbumps, while generating false alarms only about 10% and 5% of the time, respectively.

INDEX TERMS Application of wireless sensor networks, smartphone applications, pattern recognition, mobile systems, crowd sensing.

I. INTRODUCTION

A. BACKGROUND AND MOTIVATION

Road conditions in Pakistan vary from excellent conditions to dilapidated conditions. Urban infrastructure funds are limited and are spent 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. 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, their response often is that citizens need to report the locations with poor conditions in the road network so that the municipality can act. In addition to suffering from disrepair, people sometimes construct illegal speedbumps on public roads to slow down traffic near their homes and businesses. The lack of reporting of such illegal speedbumps means that sometimes it can be years before such speedbumps are actually removed. If the municipality has the data of legally constructed speedbumps, it can identify all others as illegally constructed. Municipalities can use this information to allocate more funds to areas that are desperately in need of repair. Conversely, the public can keep an eye on how efficiently the authorities work to repair road across the city.

At the same time, Pakistan is a country with a growing population. The number of cars in urban areas has increased manifold over the last decade, but unfortunately, the road network has not grown at the same rate. A study by the Texas A&M Transportation Institute estimates 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 [1]. Given the scale of this problem in the US, in the absence of similar studies in developing countries, we can only guess what the state of disrepair of roads is costing citizens there.

Mapping and measuring road conditions season after season is not a trivial problem — it requires extensive resources including expertise, manpower, time and transportation. An approach that the government alone should appoint people to monitor the state of our roads is not a viable solution. During the time a team surveys an area, conditions may have already changed.

B. PROBLEM STATEMENT

Mapping and measuring road conditions in developing countries is not a trivial problem. Not only do major segments of the road network require immediate repair, but the state of roads needs to be updated regularly to maintain a certain level of quality year-round. Naïve solutions such as creating government departments to handle the situation are expensive and are not sustainable. Therefore, the objective is to develop a low-cost, volunteer driven system that relies on crowd sensing to develop a reliable estimate of the current state of road conditions.

C. LIMITATIONS OF PRIOR ART

Kulkarni et al. [2] described a machine learning approach using accelerometer data from smartphones to detect potholes in real-time. However, instead of aggregating data from multiple users in a centralized database, they plotted the accelerometer data only in the app and emailed it to anyone as a comma separated values (CSV) file.

Similarly, Mohan et al. [3] designed and implemented Nericell, which is a system to monitor road and traffic conditions. They attempted to detect speedbumps, along with several other traffic events (stop-and-go traffic and honking) using a combination of (non-smart) cellphone and external GPS receiver. GSM was used for coarse-grain localization and GPS for occasional fine-grain localization. However, all detection functions were performed in-device, without using measurements from other smartphones. Whenever smartphone use was detected, data collection was suspended.

Mednis et al. [4] used smartphone accelerometers and integrated GPS to design and develop a real-time pothole detection system. However, like Nericell, their approach also used standalone smartphones and lacked a central database, i.e. it did not benefit from measurements taken by other phones at the same locations.

Eriksson et al. [5] mapped potholes, manholes, railroad crossings, crosswalk/expansion joints and smooth roads in urban areas using an approach that was more similar to ours. It also collected measurements in a central database and used them collectively to detect unevenness on roads. However, unlike our approach, their's relied on an in-car GPS system and three dedicated, calibrated in-cabin accelerometer sensors, which added to the system's cost.

D. 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 launch of 3G/4G services in Pakistan in 2014, more and more people have begun to adopt mobile broadband connections. Given these combined factors we were motivated to come up with a sustainable solution to our problem that leverages the high density of smartphone users in Pakistan's urban centers for crowd sense information about road conditions.

In this paper we present **CR**owdsensing **A**pplication **T**o **E**stimate **R**oad conditions (CRATER), a cloud-mobile smartphone application that is *zero-input*, i.e., it does not require any interaction with or input from users beyond installation and initial configuration. CRATER uses a centralized database that aggregates data from multiple users, thus realizing the benefits of crowdsourcing; updating measurements over time as different users pass over different routes every single day. Another obvious advantage is that as several users move over the same area over time, we can filter out false detection results that the in-app classifier may occasionally produce by using the second in-cloud meta-classifier at the back end.

We propose that the application should not require any other piece of hardware than the smartphone itself so that users can use this kind of application without buying an extra piece of hardware. While previous works suggest that the accelerometer sensors should be calibrated and placed in a fixed position in the car, our proposal allows the user to carry their device or use it during their journey, so it would increase the application's usability.

Our zero-input app serves the purpose of minimizing cognitive load and distraction for drivers by installing this app on their smartphone. The application runs in the background while the user is on the road, so it can gather the information about the shocks and vibrations that a vehicle experiences. This information is processed to determine what the vehicle went over and the resultant decision is uploaded to the server. Thus, a simple in-cloud meta classifier determines whether a sufficiently significant number of users passing through the same location have detected the presence of a pothole or speedbump. The decision of the meta-classifier is stored in the database, and published to the website. This website makes the road condition information available to the public.

The benefits of CRATER are as follows: (1) It helps the public by providing greater transparency, and also the government by providing it the means for making data-driven decisions. (2) CRATER does away with the need for citizens to submit official complaints to their local governments. (3) It removes the need for manual data collection. (4) CRATER does not require the acquisition, installation, and maintenance of, quite likely expensive, dedicated hardware. (5) The improved roads allow traffic to flow faster and smoother with less stop-and-go, leading to less congestion, consequent stress and road-rage, and also lower overall fuel consumption and air pollution.

E. RESULTS

CRATER has demonstrated a detection rate of approximately 90% for potholes and 95% for speedbumps with false alarm rates as low as 10% and 5%, respectively. Most importantly, however, it achieves these numbers without requiring the purchase of addition hardware, any calibration, installation or placement requirements of the phone while in a moving vehicle and without any input from users. The reduced quality of data that inevitably results from collecting data in uncalibrated settings is offset by a second level classifier that operates on data contributed from multiple users and generates the consensus view of locations of potholes and speedbumps.

F. CONTRIBUTIONS

The contributions of this study are as follows:

1) Creating a labeled dataset consisting of potholes, speedbumps and smooth road segments.[1]
2) Designed and developed a zero-input smartphone application that employs crowd sensing to map and measure road conditions.

II. RELATED WORK

Erikkson *et al.* [5] proposed a system called "Pothole Patrol" (P^2), that used the inherent mobility of participating vehicles, opportunistically gathered data from accelerometer and GPS sensors, and processed it to assess road surface conditions. Using a simple machine-learning approach, it showed that potholes and other severe road surface anomalies could be identified from accelerometer data. However, the system required calibrated three-axis acceleration sensors and GPS devices deployed on embedded computers in cars. Our proposed scheme can work only with smartphones that users already possess and would not have to purchase separately for the purpose of this application.

Mohan *et al.* [3] designed and implemented "Nericell": a system to monitor road and traffic conditions in chaotic settings. The system performed sensing on smartphones using the accelerometer, microphone, GSM radio, and an external GPS receiver to detect potholes, bumps, vehicle braking, and honking. As in the case of "Pothole Patrol", we would like to avoid the purchase of extensive hardware to motivate as many users as possible to install and use the application. "Nericell" was also a standalone application that lacked a central database and did not exploit the benefits of crowd sensing.

Mednis *et al.* [4] described a mobile sensing system for road irregularity detection using Android OS based smartphones. However, ground truth data was collected for training classifiers through specialized hardware (Tmote Mini sensor node with Texas Instruments micro-controller MSP430F1611 and Analog Devices 3-axis accelerometer ADXL335) and not smartphones. Furthermore, the smartphone was placed in a controlled position when gathering data. We do not wish users to forfeit use of their phones when using CRATER, and collect data opportunistically whenever a phone is not in use.

The "Pothole Detection System" by Kulkarni et al. [2] did not make use of dedicated hardware and used an algorithm to monitor pothole conditions on the road. The system used accelerometer sensors on Android smartphones to detect potholes, GPS data to geotag them and plotted their locations on Google Maps. Accelerometer data and pothole data could be mailed to any email address in the form of a CSV file. While designing the pothole detection algorithm, some threshold values on x-axis and z-axis were assumed. These threshold values were justified using a neural network technique which reported an accuracy of 90-95%. However, [2] did not take into account the benefits of crowd sensing: by gathering data from multiple users, it is likely that a larger road network could be covered much faster and the aggregated data would be more accurate. Additionally, speedbumps were not classified as features on the road.

Commissioned by the Mayor's Office in Boston, "StreetBump" [15] was an app that tracked bumps and potholes using smartphones' accelerometer data. Although the system did use a crowd sourcing approach to map the city's road conditions, it required users to record their trips by opening the app and pressing a button. Similarly, it asked users to place their device in a stable position and ensure that the app is running in the foreground while the trip is recorded, decreasing the application's usability.

Another application, "Waze" [6] is presently the world's largest community-based traffic and navigation app, although it is principally used on the West coast of the United States. It invites drivers to help each other share real-time local traffic data. However, the application does not focus on mapping road conditions and is not well-known in developing countries. It asks users to share updates on traffic accidents and police traps which distracts drivers during their journey. While we would like to map road conditions, we would prefer to do it without creating a driving hazard for our users.

"Potholes Hunter" [7] is an app that tracks potholes (available on the Google Play store for free) intended for use in Hungary. Users take photos and rank the "worst potholes in the country". The app also places these potholes on a layer on a map. Usability is reduced because users are reluctant to stop and actually record the potholes location. Another app like "Potholes Hunter," "Fill That Hole" [8] 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: minimizing user input. This will ensure that users actually use the app.

The city of Seattle is currently offering "Find It, Fix It" [9] 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 that others have submitted, so they know what areas to avoid.

To sum up, the previously proposed systems suffer from at least one or multiple limitations in the following list:

- Lacked crowd sensing feature.
- Required user participation in reporting potholes.
- Required dedicated hardware.
- Provided incomplete coverage of road anomalies, e.g., speedbumps were ignored.

[1]We shall release this dataset to the public on our website after the acceptance for publication of this manuscript.

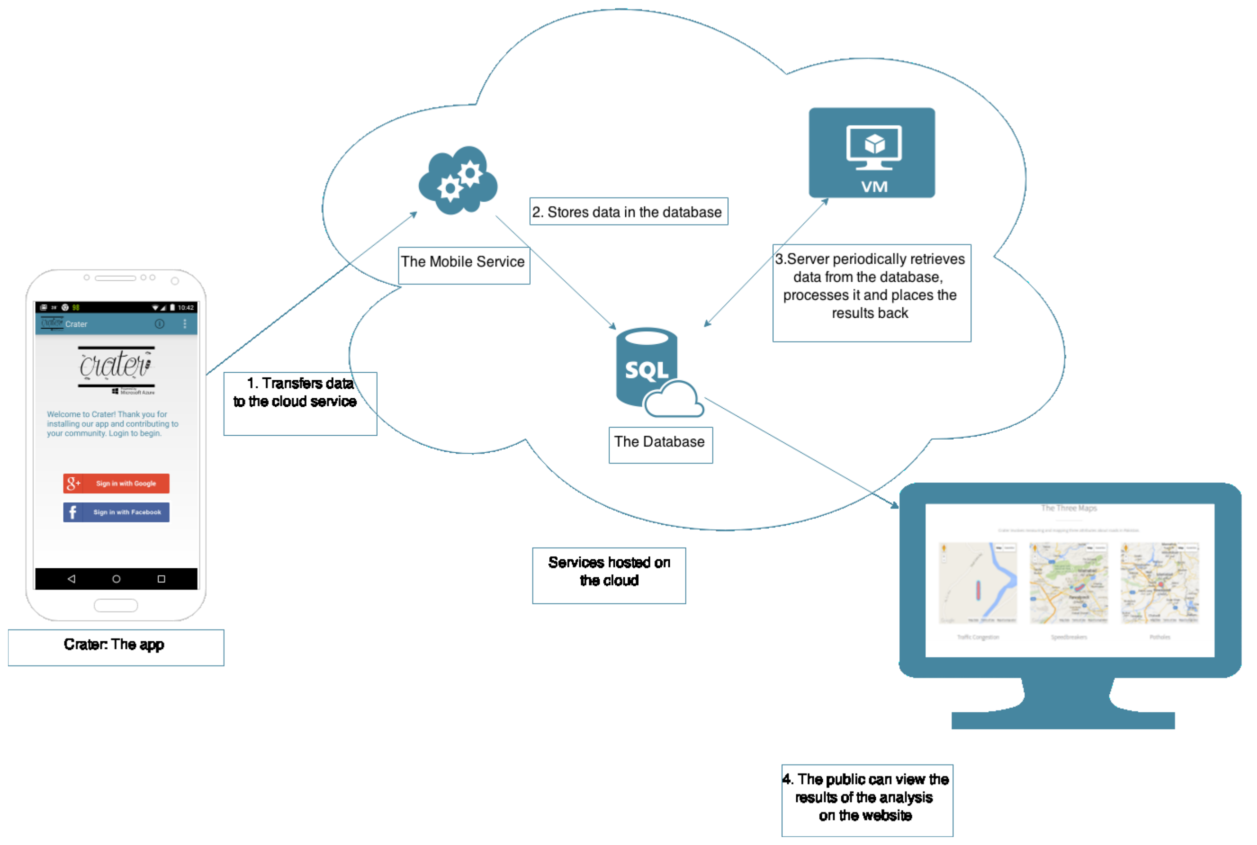

The Mobile Service
2. Stores data in the database
VM
3.Server periodically retrieves data from the database, processes it and places the results back
SQL
The Database
1. Transfers data to the cloud service
Services hosted on the cloud
Crater: The app
Welcome to Crater! Thank you for installing our app and contributing to your community. Login to begin.
Sign in with Google
Sign in with Facebook
The Three Maps
4. The public can view the results of the analysis on the website

FIGURE 1. Architectural diagram.

- Limited availability to commercial use. In some cases, mapped information was not made available on a public platform.
- Required calibration and specific placement of data collection device during data collection.

III. SYSTEM DESIGN

Figure 1 is a high-level architectural diagram of the system that depicts its major components. Figure 2 describes the flow of data between them. The following subsections describe each of these components and their working.

A. SMARTPHONE APPLICATION

The mobile app is developed for the Android OS and starts running two background processes after the user has logged in; one to gather and process data from sensors, and another to upload data. The application runs in the background unless a user explicitly chooses to close it. We used Google's Activity Recognition API for Android [16] to detect whether the user is in a moving vehicle. While the user is traveling in a car, the app tracks the phones location and collects raw data from the phones accelerometer at the highest frequency possible to get a smooth set of readings (with Android's minimum delay: `SENSOR_DELAY_FASTEST`).[2] The effective sampling rate to which `SENSOR_DELAY_FASTEST` produces varies from one phone to another phone. In our case, the resultant sampling rates ranged between $200 - 250Hz$. The app then determines the phone's orientation and performs coordinate transformation [10] to align accelerometer measurements with the downward direction (named z-axis), the direction of travel (named y-axis), and direction of lateral motion (named x-axis). We used the `SensorManager.getRotationMatrix` method in Android for linear coordinate transformation [11]. Each set of samples we collect contains the sensor readings shown in Table 1.

[2]We chose `SENSOR_DELAY_FASTEST` after trying lower sensor sampling rates of `SENSOR_DELAY_GAME`, `SENSOR_DELAY_NORMAL` after they proved insufficient for capturing a time series usable for further processing and use.

TABLE 1. A record stored in the file.

Timestamp
Linear acceleration along x-axis
Linear acceleration along y-axis
Linear acceleration along z-axis
Latitude
Longitude
Speed (estimated)
Rotation around x-axis
Rotation around y-axis
Rotation around z-axis

The data collection algorithm running in the app may be understood as described in Algorithm 1.

This yields a time series signal of the acceleration the phone experiences. This signal is processed to detect potholes

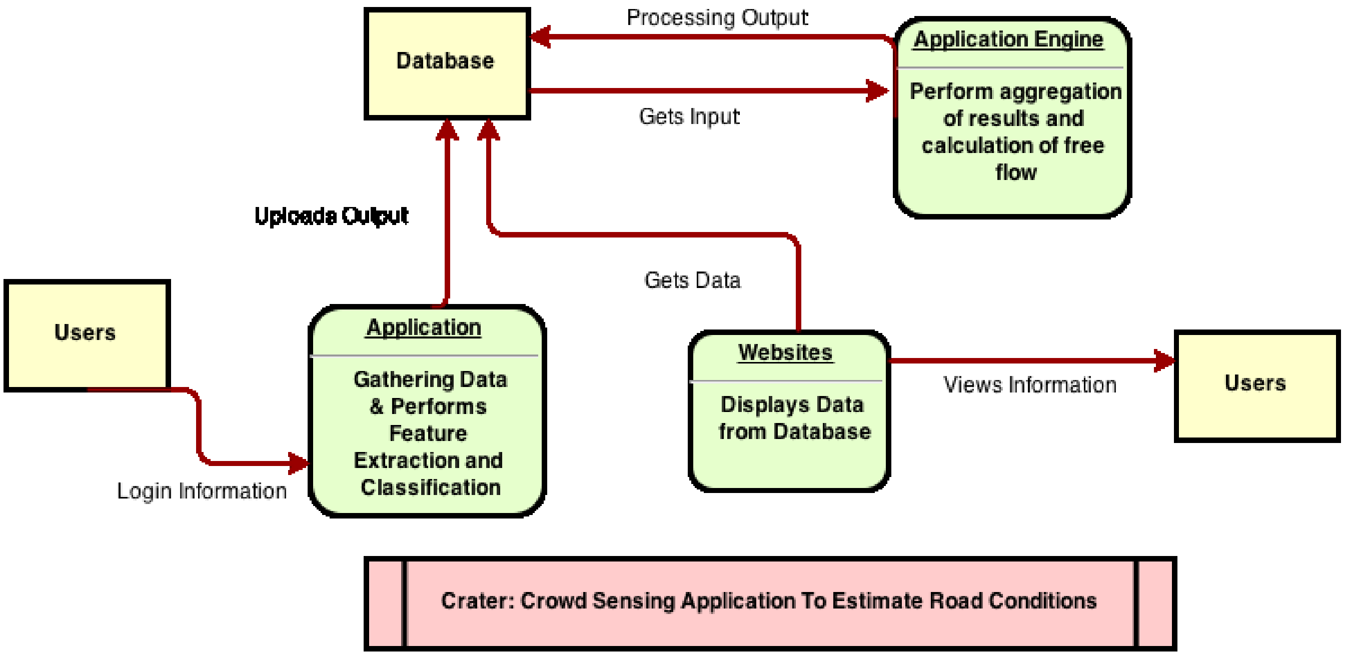

Database
Processing Output
Gets Input
Application Engine
Perform aggregation of results and calculation of free flow
Uploads Output
Gets Data
Users
Application
Gathering Data & Performs Feature Extraction and Classification
Websites
Displays Data from Database
Views Information
Users
Login Information
Crater: Crowd Sensing Application To Estimate Road Conditions

FIGURE 2. **Dataflow diagram.**

Algorithm 1 Data Collection

```
while phone has storage space do
    if phone is in a moving vehicle AND GPS available then
        turn on sensors.
        while phone in moving vehicle AND GPS available do
            log 40,000 samples ($15 - 20sec$ of data).
            transform phones coordinate axes.
            apply low and high pass filters.
            perform feature extraction.
            perform classification.
        end
        turn off sensors.
    end
    wait for 15 minutes.
end
```

and speedbumps and estimate their locations. We used binary classifiers for detecting both potholes and speedbumps that were implemented in the open source WEKA library [12] which we imported into the app. The details of the design of these classifiers will be discussed in the subsequent section.

The results of the detector are temporarily stored in a file until it is uploaded to an SQL database hosted in the Azure cloud. Since road conditions are slow to change, and in order to save user's cellular Internet usage, uploads are performed opportunistically whenever a WiFi connection becomes available. The app is allocated a maximum of 100 MB of memory to buffer data for upload.[3] When the buffer is full, the app stops collecting data until the buffer is emptied. Once the data file has been transmitted, it is removed from the users device and the app can resume logging when needed.

TABLE 2. **RESTful services per table.**

Table Name	Table Description	Services
Classified Data	Insert data received from the applications classifier into the database and retrieve it	Get, Post
Voting Results	Classified data retrieved from database for meta-classifier; results stored in the database	Get, Post

B. APP ENGINE

The app engine is deployed on the Microsoft Azure cloud. The app engine consists of custom scripts and Azure's *mobile services*: an abstraction that enables developers to easily upload data to an SQL database from a smartphone application, and retrieve the data from a web or mobile platform. We configured our mobile service to perform two essential functions:

- Upload data from the app to the database.
- Provide RESTful APIs (Table 2) that allow easy access to data for display on the website and maps in the app.
- Provide user authentication through Facebook and Google+.

[3]If the app detects an SD card installed on the phone, the app stores this data file in SD card memory, else it defaults to the phone's inbuilt memory.

The SQL database buffers data output by the in-app classifier. The meta-classifier that is executed periodically by the app engine in the Azure cloud uses this data to update speedbump and pothole locations, and then deletes it to conserve space.

The app engine is a server that periodically runs the following processes:

- Delete all old processed data from the smartphone-classified data to make room for new information.
- Get data from the classified data table, process it using the meta-classifier and push the results back. This processing involves:
  - Getting all classified data available from different users for a particular location across the map
  - For each location on the map, analyzing the data available, and computing the consensus value that describes the speedbump/pothole location and road condition. The in-cloud meta-classifier calculates whether the individual votes from the mobile applications for each category (i.e. pothole or speedbump or nothingness) are in a majority and meet a particular threshold value or not. If the threshold is met for a particular category, the location is updated to have that particular road feature.

C. THE WEBSITE

The website provides a map interface overlaid with information that is a consensus view of user reported road conditions, i.e., locations of potholes and speedbumps [13]. The same view is also provided inside in a screen inside the app as an additional feature without the need for users to switch to a browser, which serves as an additional motivation for users to install the mobile app.

D. DATABASE DESIGN

The database contains tables that store:

- User Contributed Data: This table contains data produced by the in-app classifier and contributed by all users' phones.
- Voted Consensus View: This table contains data that describes the consensus view of locations of potholes and speedbumps after processing by the app engine's in-cloud classifier. This data is then published on the website.

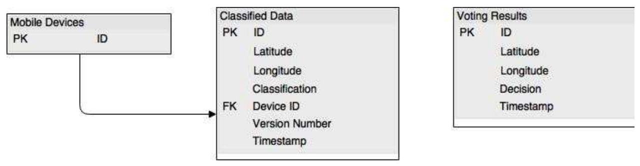


FIGURE 3. Database design.

Figure 3 shows the database architecture. The field names are all very self-explanatory. The *version number* field in the classified data table is significant as it tracks the version of the app that uploaded the data to the database. As the application is updated, we may evolve the way we process data received from various versions of the app by the app engine. Tracking the version of the app contributing data will let us process the data appropriately.

IV. DETECTOR DESIGN

A. TRAINING SET AND PRE-PROCESSING

We employed a supervised learning approach to develop the in-app classifier. This necessitated the collection of a labeled training set, which in turn required the development of a separate Android app, which we will refer to as the *data collection app*. This data collection app was designed with simplicity in mind. It lets a user start and stop collecting data by the press of a button. Specifically, the app stores readings from the three accelerometers and GPS in the background the moment it was switched on, and periodically writes them to a file. In addition, it provides two buttons to let the user mark the moment the vehicle goes over a speedbump or through a pothole. At the button press the timestamp and location of the user is stored in a file.

Note that data gathered from sensors cannot be used directly for feature extraction and classification without pre-processing first. Recall that we do not require the smartphone to be placed in a particular (calibrated) position during data collection. Therefore, we rotate the axes of the accelerometer sensors to realign them in a standard direction. Axes reorientation aligns the direction of greatest variance in accelerometer readings with the 'z-axis', which points downwards on the ground (due to the acceleration due to gravity). The direction of the second largest variance of accelerometer readings orthogonal to the z-axis is aligned with the 'y-axis,' which is usually the direction of forward motion of the vehicle. With the directions of the z and y-axes determined, the 'x-axis' is aligned with the only remaining orthogonal direction, usually sideways to the direction of a forward moving vehicle. As subsequent exploration showed, it is primarily the features of the reoriented z-axis, and to a lesser degree the y-axis, which provide useful features for detecting speedbumps and potholes. Computationally, this step involves determining the covariance matrix of accelerometer sensor readings, performing its eigen-decomposition. Each vector sample of accelerometer readings can then be rotated to align with the z, y and x-axes by performing a matrix-vector multiplication [10].

Furthermore, the Android API does not directly provide the speed of the user. To calculate the speed from successive changes in latitude and longitude, we first converted differences in geographical coordinates to distance (in km) using the Haversine formula [14]. Once we have the distance between two consecutive locations of the user, we simply divided this value by the time difference between when the user was at these two locations to approximate speed. By taking the timestamp of when a pothole/speedbump was encountered, and also studying the accelerometer and speed functions around that time, we could begin to understand what features may be worth further exploring for potential use in classifying potholes and speedbumps.

B. LABELED DATA

Ground truth gathering involved some field work – moving over the city in a car with the data collection app turned on and collecting data while marking the locations of potholes/speedbumps, as described earlier. Figure 4 is a photograph of a once paved road that is currently in a state of disrepair outside our campus that we used, among other roads, to collect data for use in the development phase.

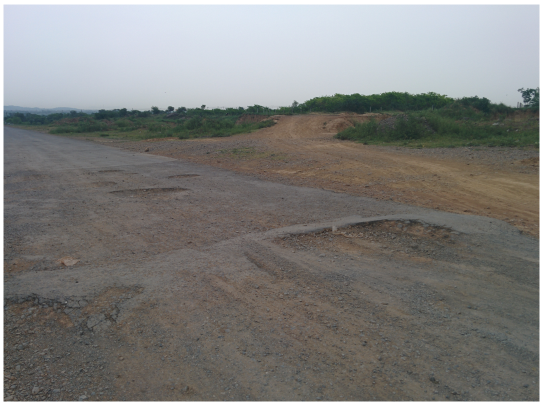


FIGURE 4. Photograph of a road in very bad condition outside NUST campus, Islamabad, located at 33.657722° N 73.001049° E.

Table 3 lists some basic properties of the data set we collected for the development of pothole and speedbump detectors. This data set was collected over many trips, driving over various roads across Islamabad and Rawalpindi cities.

TABLE 3. Distribution of labeled data.

Labeled potholes	254
Labeled speedbumps	177
Smooth road	6446
Total	6877

C. FEATURE EXTRACTION

The pothole and speedbump detectors in the app execute at one-second intervals. Each time one of them executes, it uses accelerometer readings collected in the preceding 12sec, i.e. the detectors execute on a sliding window of accelerometer samples. Figure 5b and Figure 5a are plots of accelerometer readings along x, y and z-axes over time, after performing the coordinate system rotation described in the preceding section, for a vehicle driving over a speedbump and through a pothole, respectively. As these figures show, without any further processing, this signal is quite noisy, with significant high-frequency components. We discovered during data collection that this high-frequency noise in signals is caused by two different sources.

1) The regular frequency component is vehicle vibrations due to the running engine and is present whenever the engine is running, whether or not the vehicle is moving.

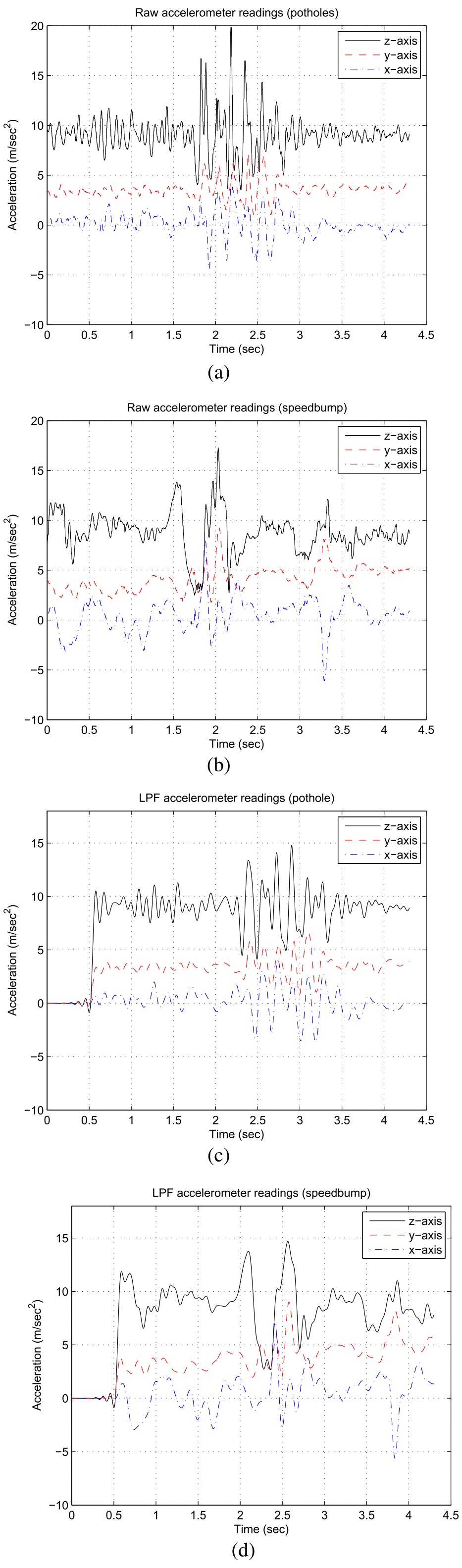


FIGURE 5. Plots of raw and low-pass filtered accelerometer readings collected while driving through a pothole and over a speedbump. (a) Raw pothole data. (b) Raw speedbump data. (c) LP filtered pothole data. (d) LP filtered speedbump data.

It is more generated when the vehicle is stopped and can be found in accelerometer signals along all three axes.

2) The high-frequency noise may also have an irregular component which is due to unevenness of the road surface and varies depending on how fine or coarse the carpeting of the road is. Obviously, this component is present only when the vehicle is in motion and its frequency depends on the vehicle's speed. This component is most pronounced along the z-axis accelerometer signal

Since neither high-frequency components are useful for our objective, we remove them from the signal(s) by passing them through low-pass (LP) filters. Figure 5(d) and Figure 5(c) plot the LP filtered signals for the examples in Figure 5(b) and Figure 5(a), respectively.

We extracted the following set of features from the LP filtered signals and assessed their usefulness for speedbump and pothole detectors. The most significant features included:

- Average, minimum and maximum speed
- Standard deviation of accelerometer readings across x/y/z-axis
- Mean of accelerometer readings across x/y/z-axis
- Standard deviation of low-pass filtered accelerometer readings across x/y/z-axis
- Mean of low pass filtered accelerometer readings across x/y/z-axis
- Number of minimas and maximas one/two standard deviation(s) away from the mean on x/y/z-axis
- Number of points one/two low pass filtered standard deviation away from the low pass filtered mean on the x axis
- Number of points one/two standard deviation(s) away from the mean on x/y/z-axis of low-pass filtered signal
- Maximum/minimum value on x/y/z-axis

We trained and considered five different types of classifiers using a supervised learning approach, which included naïve Bayes, C4.5 decision tree, SVM, decision tables and supervised clustering. The SVM classifier was trained using sequential minimal optimization (SMO).

V. RESULTS

A. IN-APP DETECTORS

The app gathers data, rotates the phone sensors' axes to conform with the direction of motion of the vehicle, preprocesses it by filtering, extracts features and and uses them to detect speedbumps/potholes on the road segment. In this section we describe our design choices and decisions for these detectors. The results of the classifiers serving as speedbump/pothole detectors and the timestamp of when the measurments were recorded and their geolocation, were stored in a file. The app opportunistically uploads this data to the app database in the cloud whenever it connects to a WiFi network. Table 4 and Table 5 show the performance of the speedbump and pothole detectors, respectively. These results were generated using the labeled training set described in Section IV-B and using 10-fold cross validation.

TABLE 4. In-app speedbump classifier performance.

Classifier	TP rate	FP rate	Precision	Recall	F-measure	ROC Area
SVM	58.8%	4.0%	81.3%	58.8%	68.2%	88.7%
Naïve Bayes	88.1%	89.6%	21.9%	9.6%	88.1%	17.4%
C4.5	60.5%	0.8%	66.5%	60.5%	63.3%	72.4%
Decision Table	63.8%	3.8%	30.6%	63.8%	41.4%	86.8%
Sup. Clustering	0.0%	0.0%	0.0%	0.0%	0.0%	50.0%

TABLE 5. In-app pothole classifier performance.

Classifier	TP rate	FP rate	Precision	Recall	F-measure	ROC Area
SVM	57.5%	4.0%	83.4%	57.5%	68.1%	90.1%
Naïve Bayes	69.3%	86.9%	14.3%	15.6%	69.3%	25.5%
C4.5	63.8%	1.2%	67.8%	63.8%	65.7%	80.5%
Decision Table	46.5%	0.2%	88.7%	46.5%	61.0%	84.4%
Sup. Clustering	79.1%	23.2%	11.6%	79.1%	20.2%	78.0%

1) CLASSIFIERS FOR SPEEDBUMP DETECTOR

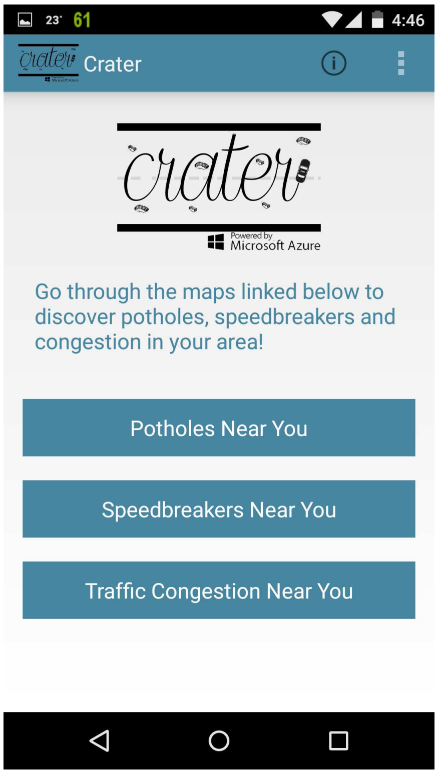
Since speedbumps are relatively rare occurrences on the road, we chose to remove any classification algorithms with high false positives (FP) rates from further consideration. Based on the FP rates in Table 4, this disqualified naïve Bayes. Supervised clustering was dropped because it delivers a true positives rate of 0%, making it completely useless. This leaves us with SVM, C4.5 decision tree and decision table which all have very similar performance in terms of TP and FP rates. Of these three, SVM clearly outperforms C4.5 decision tree and decision table in terms of precision.

2) CLASSIFIERS FOR POTHOLE DETECTOR

A similar analysis of the same five classifiers for the pothole detector rules out naïve Bayes and supervised clustering due to very high FP rates. Of the three remaining candidates, decision tables have a significantly lower TP rate than SVM and C4.5 decision tree, thus eliminating it from further consideration. Between SVM and C4.5 decision tree, SVM beats C4.5 in terms of recall, F-measure and ROC area by significant margins. Therefore, we ended up selecting two SVM classifiers for speedbump and pothole detectors.

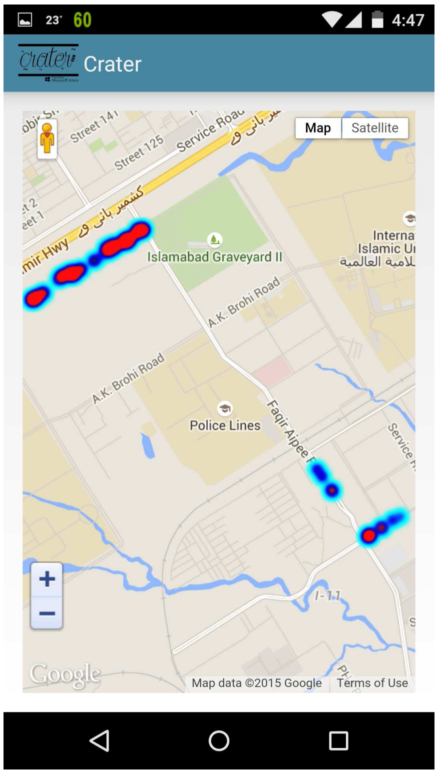
B. IN-CLOUD DETECTOR

The app engine in the cloud receives the results of the in-app speedbump and pothole detectors tagged with timestamps and GPS coordinates. The in-cloud detector assigns each detected speedbumps/potholes into a $2m \times 2m$ bin on a map grid. Presently, we are using a very simple threshold based in-cloud detector: if more than N users report detecting a speedbump (pothole) at a grid location, the location is marked as having a speedbump (pothole). We set the threshold value of N for Islamabad to 5. Thus, the final determination of whether a



Go through the maps linked below to discover potholes, speedbreakers and congestion in your area!

(a) CRATER app home screen.



(b) CRATER app pothole map screen.

FIGURE 6. Screenshot of the app's map screen showing the locations of potholes as a heatmap layer on Google maps. (a) CRATER app home screen. (b) CRATER app pothole map screen.

speedbump (pothole) is to be marked on the map is made by consensus. In this way, the in-cloud detector filters out speedbumps (potholes) detected erroneously by the in-app detector. The consensus-based results of the in-cloud detector are then published to a website and also made available through the app itself. Figure 6 shows the home screen and the map screen showing locations of potholes as a heatmap layer in the app. To test the output of the in-cloud detector, we compared its detection results to manually marked potholes and speedbumps on a roughly 10km route through the city. The in-cloud detector yielded detection rates of about 90% for potholes and 95% for speedbumps and false alarm rates of about 10% and 5%, respectively. We attribute the 10% false alarm rate to the poor quality of roads in general, even those that are carpeted and not visibly in need of repair.

VI. CONCLUSION

In this study, we developed CRATER, an opportunistic mobile crowd sensing application to estimate road conditions. The design and development of CRATER was informed by a number of systems for the same purpose that have been developed previously. CRATER distinguishes itself from prior attempts at road condition measurement systems in that it does not require any hardware beyond an Android phone with GPS and accelerometers. The Android app that is part of CRATER has been simplified to the point that contributors do not need to engage in entering any data into the app. Instead, we opted for a system design that relies on zero-input from its users. Very significantly, the system does not require that phones be placed in any particular location or orientation inside moving vehicles.

Our findings show that CRATER's final detection rates for are approximately 90% for potholes and 95% for speedbumps. We attribute the difference in these detection rates to the greater variety of potholes relative to the variety in speedbumps which, even with varying lengths and heights, are nevertheless intentionally built, unlike potholes.

A. FUTURE WORK

We realize that there are several other problems that can be attacked with the kind of data being collected by CRATER's app. First, there is the fact that it may be possible to leverage the information in the high frequency components of accelerometer readings to assess the quality and smoothness of the surface of roads, i.e., whether a road is paved or not. This could allow us to classify roads into several sub-types, i.e., carpeted/paved, unpaved, gravel, dirt roads, etc.

Another problem that could be attempted is the detection of traffic congestion on roads. Free-flow traffic conditions could be established by tracking vehicle speeds at low-traffic times during the day/night. Mobility traces from participating contributors could then be used to identify traffic congestion by identifying places where vehicle speeds significantly deviate from earlier established free-flow traffic conditions.

ACKNOWLEDGMENT

The authors would like to thank Microsoft Research for a generous grant under the *Microsoft Azure 4 Research* program which provided us with the resources that enabled this project. This research was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2014006438).

REFERENCES

[1] Z. Zhang *et al.*, "It's about time: Investing in transportation to keep Texas economically competitive: Appendices," 2011.

[2] A. Kulkarni, N. Mhalgi, S. Gurnani, and N. Giri, "Pothole detection system using machine learning on Android," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 4, no. 7, pp. 360–364, Jul. 2014.

[3] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, 2008, pp. 323–336.

[4] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using Android smartphones with accelerometers," in *Proc. Int. Conf. Distrib. Comput. Sensor Syst. Workshops (DCOSS)*, 2011, pp. 1–6.

[5] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proc. 6th Int. Conf. Mobile Syst., Appl., Services*, 2008, pp. 29–39.

[6] WAZE. (May 30, 2015). *Free Community-Based Mapping, Traffic & Navigation App*. [Online]. Available: https://www.waze.com/

[7] Plus Design & Marketing Transportation. (2016). *Pothole Hunters*. [Online]. Available: https://play.google.com/store/apps/details?id=cz.vymoly.android&hl=en

[8] Cycling UK. (2015). *Fill That Hole*. [Online]. Available: http://www.fillthathole.org.uk/iphone

[9] City of Seattle. (2015). *Find It, Fix It*. [Online]. Available: http://www.seattle.gov/customerservice-bureau/find-it-fix-it-mobile-app

[10] Nvidia. *Tegra Android Accelerometer Whitepaper*, accessed on May 15, 2015. [Online]. Available: http://developer.download.nvidia.com/tegra/docs/tegra_android_accelerometer_v5f.pdf

[11] Stackoverflow. (2013). *Calculate Acceleration in Reference to True North*. [Online]. Available: http://stackoverflow.com/questions/14963190/calculate-acceleration-in-reference-to-true-north/14988559

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

[13] F. Kalim. (2015). *Crater*. [Online]. Available: http://craters.azurewebsites.net

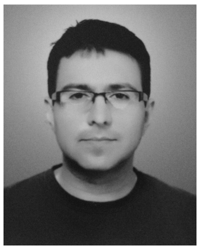
[14] C. C. Robusto, "The cosine-haversine formula," *Amer. Math. Monthly*, vol. 64, no. 1, pp. 38–40, 1957.

[15] C. Bits. *Connected Bits*, accessed on Sep. 21, 2016. [Online]. Available: http://www.connectedbits.com/

[16] Google, Aug. 2015. *Activityrecognitionapi—Google Apis for Android—Google Developers*. [Online]. Available: https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi

FARIA KALIM received the B.E. degree in software engineering from the National University of Sciences and Technology, Islamabad, in 2015. She is currently pursuing the Ph.D. degree in computer science with the University of Illinois at Urbana–Champaign, IL, USA.

JAEHOON (PAUL) JEONG received the B.S. degree from the Department of Information Engineering, Sungkyunkwan University, in 1999, the M.S. degree from the School of Computer Science and Engineering, Seoul National University, in 2001, and the Ph.D. degree from the Department of Computer Science and Engineering, University of Minnesota–Twin Cities, in 2009. He was with the Electronics and Telecommunications Research Institute as a Member of Research Staff from 2001 to 2004. He was with Brocade Communications Systems as a Software Engineer from 2010 to 2012. He has been an Assistant Professor with the Department of Software, Sungkyunkwan University, since 2012. His advisors were Professor David H.C. Du, Professor Tian He, and Professor Yanghee Choi. He was a Member of the Minnesota Embedded Sensor System Laboratory, and the Multimedia and Mobile Communications Laboratory.



MUHAMMAD U. ILYAS received the B.E. degree (Hons.) in electrical engineering from the National University of Sciences and Technology, Rawalpindi, Pakistan, in 1999, the M.S. degree in computer engineering from the Lahore University of Management Sciences, Lahore, Pakistan, in 2004, and the M.S. and Ph.D. degrees in electrical engineering from Michigan State University in 2007 and 2009, respectively. Since 2011, he is currently an Assistant Professor with the National University of Sciences and Technology, Islamabad. His research interests include wireless communication networking, statistical data analysis modeling, graph theory, information theory, applied optimization, algorithms, and social networks.