



# Mobile Sensing for Multipurpose Applications in Transportation

Armstrong Aboah<sup>1</sup> · Michael Boeding<sup>2</sup> · Yaw Adu-Gyamfi<sup>1</sup>

Received: 10 May 2022 / Revised: 30 September 2022 / Accepted: 3 October 2022 / Published online: 9 November 2022  
© The Author(s), under exclusive licence to Springer Nature Singapore Pte Ltd. 2022

## Abstract

Routine and consistent data collection is required to address contemporary transportation issues. The cost of data collection increases significantly when sophisticated machines are used to collect data. Due to this constraint, State Departments of Transportation struggle to collect consistent data for analyzing and resolving transportation problems in a timely manner. Recent advancements in sensors integrated into smartphones have resulted in a more affordable method of data collection. The primary objective of this study is to develop and implement a smartphone-based application for transportation-related data collection. The app consists of three major modules: a frontend graphical user interface (GUI), a sensor module, and a backend module. While the frontend GUI enables interaction with the app, the sensor modules collect relevant data such as video, gyroscope, motion and accelerometer readings while the app is in use. The backend leverages a real-time database to stream and store data from sensors, together with providing the computational resources needed to support the application. In comparison to other developed apps for transportation data collection, this app is not overly reliant on the internet enabling the app to be used in internet-restricted areas. Additionally, the app is designed for multipurpose applications in transportation. The collected data were analyzed for a variety of purposes, including calculating the International Roughness Index (IRI), identifying pavement distresses, and understanding driver's behaviors and environment. From the sensor data, we detected turning movements, lane changes and estimated IRI values. In addition, several pavement distresses were identified from the video data with machine learning.

**Keywords** Mobile sensing application · Accelerometer · Deep learning · International roughness index (IRI)

## Introduction

Among today's transportation issues are congestion, safety, equity, aging infrastructure, energy, sustainability, security among others. To find long-term solutions to these issues, there is a need for data-driven approaches that can collect and process quality, high-resolution, and high-fidelity streams of data at a reasonable cost. The current state of transportation data collection relies on expensive systems

that are not easily scalable for continuous data streaming. The Virginia Department of Transportation (VDOT) for example, spends about \$1.8 million per year on pavement data collection and evaluation using high-end machines (Sauerwein et al. 2011). The cost-prohibitive nature of this effort could lead to infrequent infrastructure assessment for agencies with limited funding. Low-cost data collection alternatives that are able to produce information comparable to these high-end systems are needed. The goal of this study is to tap into advancements in consumer-level technologies such as smartphones to scale the collecting of transportation data for continuous performance evaluation and decision-making.

The current generation of smartphones are enabled with a plethora of sensors such as accelerometers, gyroscopes, compass, and cameras. These sensors can be used to capture various dynamics of the transportation system. The accelerometers for example, detects vibrations caused by a moving vehicle. These vibrations are used as a surrogate to estimate the roughness of the road. Numerous studies have

---

✉ Armstrong Aboah  
aa5mv@umsystem.edu

Michael Boeding  
michaelboeding@mail.missouri.edu

Yaw Adu-Gyamfi  
adugyamfi@missouri.edu

<sup>1</sup> Civil and Environmental Engineering Department, University of Missouri-Columbia, Columbia, USA

<sup>2</sup> Computer Science Department, University of Missouri-Columbia, Columbia, USA

been conducted to estimate the IRI using smartphone data, and the results are comparable to those obtained using high-end machines with an acceptable margin of error (Aboah and Adu-Gyamfi 2020; Aleadelat et al. 2018; Sauerwein et al. 2011; Zeng et al. 2018). The limitation of these approaches is that the type of vehicle used to collect the data influences its accuracy, as different vehicles respond differently to vibration. With the aid of computer vision and machine learning techniques, the video cameras and motion sensors can be used to understand the driving environment: extract vehicle trajectories, understand lane-changing behaviors, estimate vehicle density, weather condition, etc. (Aboah and Adu-Gyamfi 2020; Aleadelat et al. 2018; Efthymiouet al. 2019, Dhatbale et al. 2021, Miao et al. 2022, Robinson and Cook 2012; Zeng et al. 2018).

The study's aim is to develop and implement a mobile data collection application for collecting transportation data for multiple applications including road condition evaluation, naturalistic driving studies, etc. A modular software design process was followed to allow for the future expansion of the mobile app. The top module is a high-level user interface that uses different layouts, menus, buttons, and notifications to create a user-friendly interface for a wide range of users. A sensor module manages all key technologies for data collection including GPS, Camera, Accelerometer and Gyroscope. The back-end module handles communications between the different modules, data storage and requests. In comparison to previous smartphone data collection apps, this current app is not overly reliant on an internet connection to transmit data. This means that data can be gathered and stored temporarily in the app's library before being uploaded to the cloud server for storage when internet connectivity is available. This feature enables the data collection from roadways in Wi-Fi dead zones. In addition, the use of the firebase backend enables data to be streamed from multiple sensors simultaneously and instantly. Recorded data can therefore be displayed instantly via web or other apps synced with the apps' database.

The rest of the paper is divided into the following sections. The second section reviews relevant literature on smartphone applications in solving transportation-related problems. Section three contains information about the development process, including the design approach, key components, and modulus. The fourth section discusses the data collection process as it relates to the developed mobile application. Section five summarizes the quantitative findings from the collection of field data. Finally, Section six summarizes the research, results the findings, and makes recommendations for future research in section seven.

## Related Studies

This section discusses studies that examine the use of smartphone data to solve transportation-related problems. Each study is reviewed for its purpose, data collection technique, and methodology.

The technology to accurately assess pavement roughness with inexpensive sensors has improved greatly. A probe-based monitoring system for slippery and rough road surfaces was developed by MDOT in 2010 (Robinson and Cook 2012). The vehicle data was collected and transmitted to a backend server running on an Android platform. It was attached to the windshield in the same manner as a navigation device. Various sensors in the vehicle and on the phone were used to collect data, including the phone's three-axis accelerometer, the external road surface temperature and humidity, and the vehicle's Controller Area Network. During a two-year period, the system was installed in two vehicles driven by MDOT personnel. Over thirty thousand miles and more than 13 gigabytes of data were captured. To represent the pavement's surface irregularities, the vertical accelerometer signal's variation was employed. The accelerometer's sample rate is 100 Hz. Data collection was followed by calibration of the accelerometer readings using a PASER system curve fitting algorithm to the Pavement Surface Evaluation and Rating (PASER) scale. Future iterations of the curve fitting algorithm may incorporate data from the MDOT's annual PASER rating study. Researchers at Auburn University examined the use of vehicle-mounted sensors to determine the condition of road pavements (Dawkin et al. 2010). The study's primary objective was to compute the IRI through the use of automotive sensors. The IRI was calculated using information gathered from a variety of vehicle sensors, including suspension deflection meters, accelerometers, and gyroscopes. On a 1.7-mile (2750-m) long test track, the National Center for Asphalt Technology conducted controlled speed tests (NCAT). The total number of vibrations in a particular section can be determined by calculating the Root Mean Square (RMS) of a signal measurement (i.e., vertical acceleration, gyroscopes, or suspension deflection). Acceleration includes a section on RMS (Root Mean Squared) (Sect. 3.2). Following that, the aggregated vibrations were compared to the pavement segment's true IRI. The root mean square of vertical accelerations was found to be the most accurate representation of the true IRI. It followed the same general trend as the well-known IRI, with the exception of a few expected magnitude changes. In summary, this study established that the most practical method for calculating the IRI is to use an RMS algorithm on vertical acceleration readings. In a pilot study, Flintsch et al. (2012) quantified

road ride quality and roughness using probing vehicles. Once again, vertical acceleration data was used to estimate vehicle vibration. At the Virginia Smart Road facility in Blacksburg, Virginia, a smoothness profile was created using an inertial-based laser profiler, and vertical accelerometer measurements were taken using an instrumented car. The frequency of the accelerometer was chosen to be 10 Hz. Additionally, GPS coordinates were collected. Acceleration data was collected during four runs on the test track. The study discovered that acceleration runs are highly reproducible. The smoothness profile and acceleration measurements are highly correlated, as determined by the coherence function analysis.

Papadimitriou et al. (2018) used smartphone data to detect and analyze risky driving behaviors. The study examined critical risk indicators such as the number of aggressive driving incidents and cell phone use while driving. The study gathered data on vehicle speed, distance traveled, accelerations, turning maneuvers, braking events, and cell phone use. The study's findings indicate that distraction caused by smartphone use has a significant effect on the number of severe events occurring per kilometer and, consequently, on the relative crash risk. Additionally, smartphone sensor data can be used to accurately detect mobile phone use while driving in more than 70% of cases. Another application of mobile apps is to enhance traffic control devices and to alleviate confusion among motorists passing through work zones. For example, a smartphone-based audio warning message (AWM) was proposed and tested in driving simulators to supplement conventional traffic controls and increase worker safety in work zones (Li et al. 2016). The National Highway Traffic Safety Administration (NHTSA) and state transportation departments have implemented a variety of safety countermeasures aimed at reducing forward collisions in work zones. Traditional countermeasures, on the other hand, frequently fail to prevent crashes in work zones due to the complexity of traffic. Craig et al. (2017) conducted a study in Minnesota to determine the effect of in-vehicle messages on drivers' perceptions of work-zone events. Researchers at Texas Southern University's Innovative Transportation Research Institute developed a warning system application that alerts drivers to hazardous traffic situations through a variety of warning messages, including sound, visual, and voice (Dutzik et al. 2013; Rahman et al. 2016). In addition, researchers at the University of Minnesota developed a smartphone app that uses embedded sensors and Bluetooth technology to provide pedestrians with routing instructions when upcoming work zones are detected. When a work zone is detected, the smartphone vibrates to alert users, and the app then broadcasts an audible message to them (Liao 2014). Azadi et al. (2020) developed and deployed a work zone application for collecting, reporting, and storing real-time work-zone activity information. The study indicated that the

precision of sensors such as GPS was within appreciable accuracy of work zone geolocation.

## Design Approach

The current app was designed to collect data from the onboard IOS sensors and video from the onboard camera and then transfer the data instantly to a cloud-based, real-time database. The current design is made up of three major modules: a frontend user interface module, a sensor module, and a backend module. Figure 1 shows the interactions and information flow between these modules.

### Frontend Design

The user interface for this app was designed to provide a streamlined view that exposed key aspects of the application via a tab bar controller. This enables us to compartmentalize the application's various critical components into distinct views. As illustrated in Fig. 2, the tab bar contains a Video Tab, an Uploads Tab, a Library Tab, a Graph Tab, and a Settings Tab. The Video Tab enables the collection of video and sensor data. The Uploads Tab enables the user to keep track of the upload status of each package gathered in the Video Tab. The Library Tab displays all of the packages that have been collected and are currently stored on the device. The Graph Tab displays a live graph of the device's acceleration in the z direction as measured by its sensors. The Settings Tab displays all of the current settings and enables them to be modified as necessary.

### Backend Design

The primary function of the backend is to provide computational resources that can be used to accelerate front-end user query responses. The analytics performed on the front end of the application can be computationally expensive. To enable such sophisticated analytics on the front end of the app, we built a scalable, cloud-based backend using cutting-edge big data analytics techniques. The current study's backend is powered by Google Firebase. Firebase is made up of several parts, including a Realtime Database, a Cloud Firestore, and Cloud Storage. The Firebase Realtime Database is a relational database management system that runs in the cloud. The data is stored in JSON format and is synchronized with each connected client in real time. Similarly, the Cloud Firestore maintains data consistency across client applications and enables offline support for mobile and web applications. Cloud Storage enables massive scalability of file storage. It allows users to upload and download files directly to and from the Cloud Storage "bucket". The developed application stores all sensor data

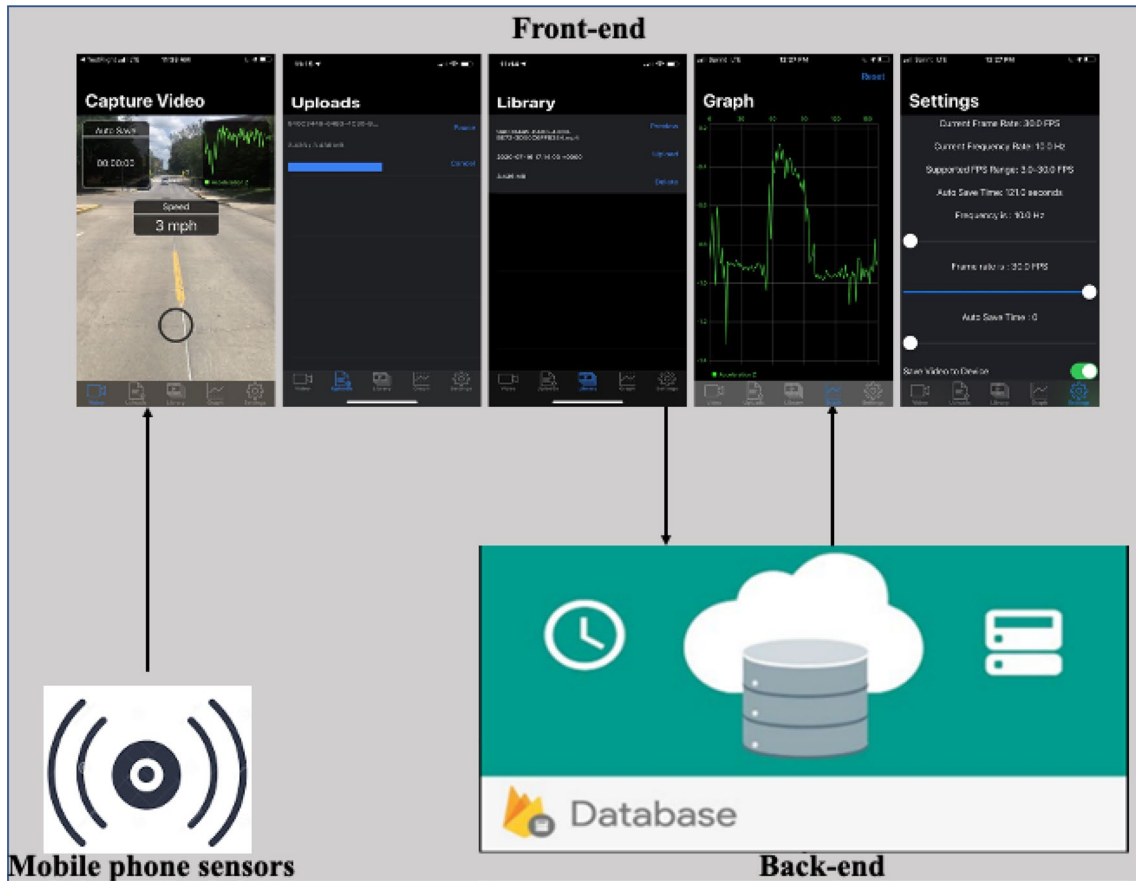


Fig. 1 Frontend, and backend modules of the App

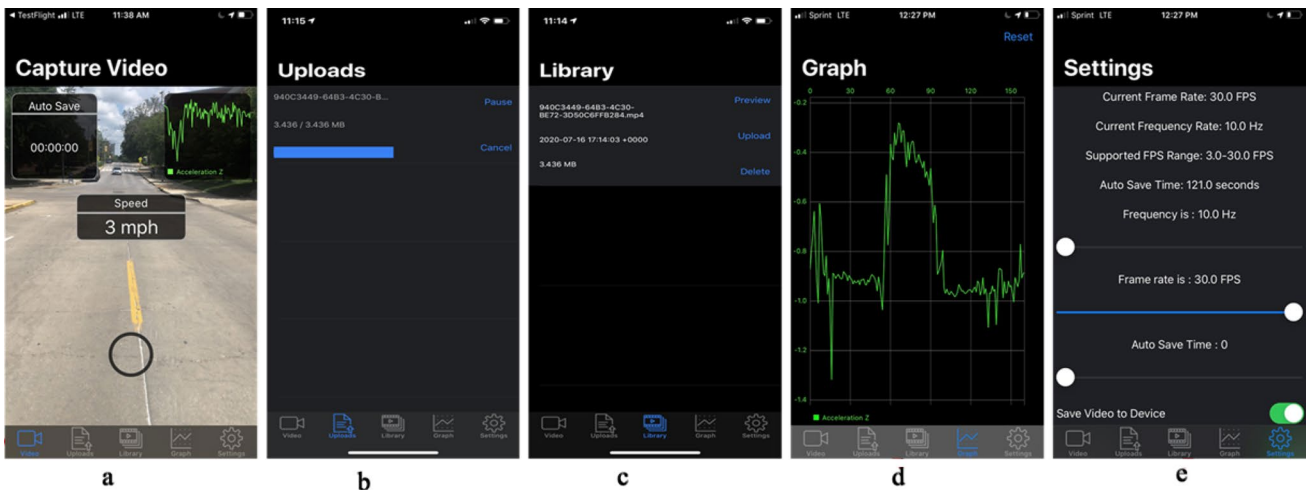
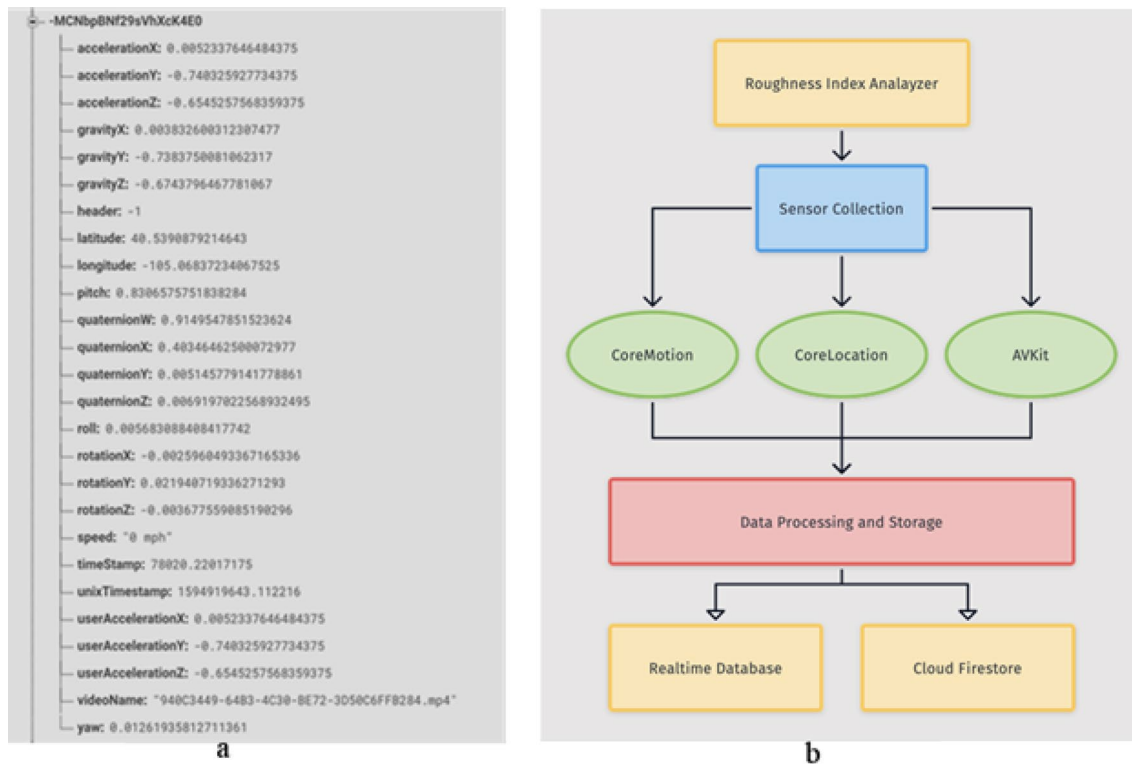


Fig. 2 Key components of the app: a video tab, b upload tab, c library tab, d graph tab, e settings tab

in a Firebase Real-time Database and the videos in Cloud Firestore. The use of both the Firebase Realtime Database and Cloud Firestore ensure that uploaded data are made instantly accessible to other app users. Additionally, they allow multiple users to simultaneously push data to the

cloud storage in real time. Finally, the Cloud Firestore caches data that your app is actively using, allowing the app to write, read, listen to, and query data even when the device is not connected to the internet. The structure of the backend real-time database is shown in Fig. 3a.



**Fig. 3** a Real-time posting of data collected, b sensor module

## Sensor Module

The developed app leverages multiple modules to allow for sensor collection. These modules include CoreMotion, CoreLocation, and AVKit as shown in Fig. 3b. The CoreMotion module collects motion and environmental data from the onboard hardware of iOS devices, including accelerometers and gyroscopes, as well as the pedometer, magnetometer, and barometer. This framework enables the access and utilization of data generated by the hardware. The CoreLocation determines the altitude, orientation, and geographical location of a device. The framework collects data by utilizing all available hardware components on the device, including barometer, GPS, Wi-Fi, magnetometer, and Bluetooth. In addition, the AVKit provides a high-level interface for video content playback. These various modules were utilized in this current app to access the device motion data (accelerometers and gyroscopes), the GPS location, and the video data, respectively.

## Data Collection Using Developed App

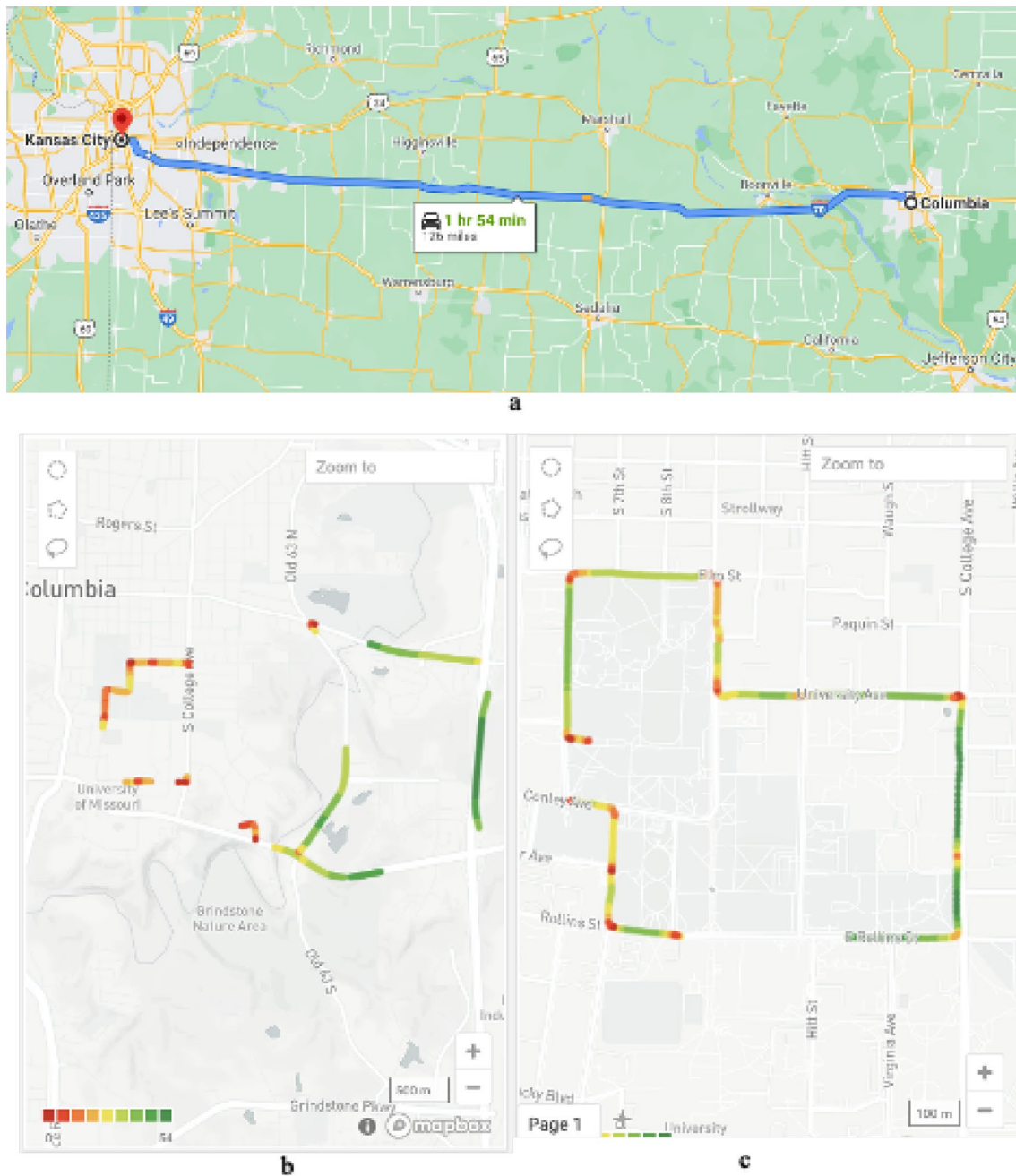
In order to test the various components of the app, we collected data on both freeways and local routes for a variety of applications. Figure 4a shows the study areas for which data

was collected. The first study was conducted on I-70 West, which connects Columbia, Missouri to Kansas City, Missouri. The road segment considered for this test was approximately 45 miles in length. The second and third study were conducted in the city of Columbia as shown in Fig. 4b, c.

To collect data with the app, the mobile phone must first be mounted on the car's windshield (see Fig. 5b) to record vehicle accelerations, rotations and some other pertinent information. The video data were sampled at a frame rate of 10 frames per second, whereas the accelerometer and vehicle location data were collected at a frame rate of 30 samples per second (30 Hz). As illustrated in Fig. 5a, the vehicle used to collect data was a 2007 Nissan Sentra. The study was conducted by a single driver and lasted for a month. A total of over 100 h of video data were collected from both highway and arterials.

## An Interactive User Interface for Data Querying using Streamlit

It is critical for the app to be able to sync the numerous pieces of information it collects. To accomplish this, the study used Streamlit to create an interactive user interface for integrating all of the various sensor data (Streamlit 2020). Streamlit is a Python library for web application development that is completely free and open-source. Streamlit



**Fig. 4** a–c The study area where the app was used to collect pavement roughness information

is compatible with a number of well-known libraries and frameworks, including Keras, OpenCV, Vega-Lite, Pytorch, Tensorflow, and Python. The study used this library because of its ease of use and rapid deployment to create an interactive dashboard that enabled effortless data querying by timestamp.

The dashboard is divided into three sections: the home-page, data querying, and data visualization. Users can query both raw data and specific video frames by timestamp using the data querying page. Users can view changes in

accelerometer readings while the vehicle is in motion on the data visualization page.

### Evaluating the Accuracy of the GPS Coordinate Points in the App

The GPS coordinates must be precise to synchronize the video and accelerometer data from the app. This will allow us to map pavement distresses in real time at the exact locations where they occurred. Issues arise when GPS

**Fig. 5** **a** 2007 Nissan Sentra vehicle used for the data collection. **b** The mounting position of the smartphone on the windshield



information is not accurate. Figure 6 is a dashboard showing the synchronization of the various data. The dashboard is divided into five sections labeled A-E. Label A displays GPS data colored according to the vehicle's speed. The changes in the accelerometer readings are depicted in Label B, while the changes in the gyroscope readings are depicted in Label C. Label D displays the various trip identifiers, and

Label E displays the speed profile. The accuracy of the GPS was determined by calculating the absolute mean deviation between the actual road center coordinates and the GPS-recorded coordinates at the same log mile. The GPS coordinates were found to be approximately 6.6 m off, which may affect the data collection's synchronization. Even though the study was only concerned with the average deviation, it was



**Fig. 6** A dashboard showing data synchronization

found that significant deviations occur when there is a gradient change in road alignment (i.e., turning movement, curve negotiation, etc.) and in areas with high-rise buildings. The spikes in the z-coordinate of the gyroscope data correspond to the various turns made during the journey. Additionally, the orientation of the spikes indicated whether the turn was a left or a right. The spikes in the accelerometer reading (accelerometer z) correspond to extremely rough sections of the roadway.

## Applications of Data Collected from the App

The app collects a variety of data, including accelerometer and gyroscope readings, roadway video, and GPS location data. This section discusses some of the possible uses for the app's data.

### Estimating Road Roughness Index (IRI)

The accelerometer, gyroscope, and speed data collected by the app were used in the study to forecast IRI values for road segments. The smartphone was mounted on the vehicle's windshield during data collection. The study also obtained ground truth IRI values from the MoDOT's ARAN viewer portal shown in Fig. 7. The portal includes three tabs for pavement information: Condition, IRI, and Rut. Each tab contains records spanning the years 2009 to 2019. The portal includes a search box for locating

roadways with relevant pavement information. To select a section of a roadway, the information for the road section's Start log (Begin log) and End log is entered in the search boxes shown in Fig. 7.

The goal is to learn IRI values from telemetry data. The independent variables used in this application include accelerometer reading, gyroscope reading, and vehicle speed. The dependent variable on the other hand was the IRI values extracted from MoDOT's ARAN viewer portal.

The study utilizes a TabNet model from the Fastai libraries. Fastai is a deep learning framework built on top of PyTorch's libraries. The model was trained on an NVIDIA GTX 1080ti GPU. The number of training sample was 432. Due to the small size of the dataset, it was divided into ratios of 0.80:0.10:0.10, with each ratio representing the training, validation, and testing datasets, respectively. The root-mean-squared-percentage-error (RMSPE) metric was used to evaluate the performance of the developed model. The RMSPE was calculated by subtracting the predicted IRI value from the actual IRI value for each prediction and expressing the result as a percentage. Figure 8 shows a convergence plot of the training and validation loss for the selected attributes. The plot demonstrates that the developed model is capable of rapidly learning the underlying relationships in the dataset. The RMSPE was approximately 0.174 after 18 iterations. This indicates that the error associated with predicting with this model was approximately 17.4%. The developed model had training and validation losses of 0.015 and 0.032, respectively.

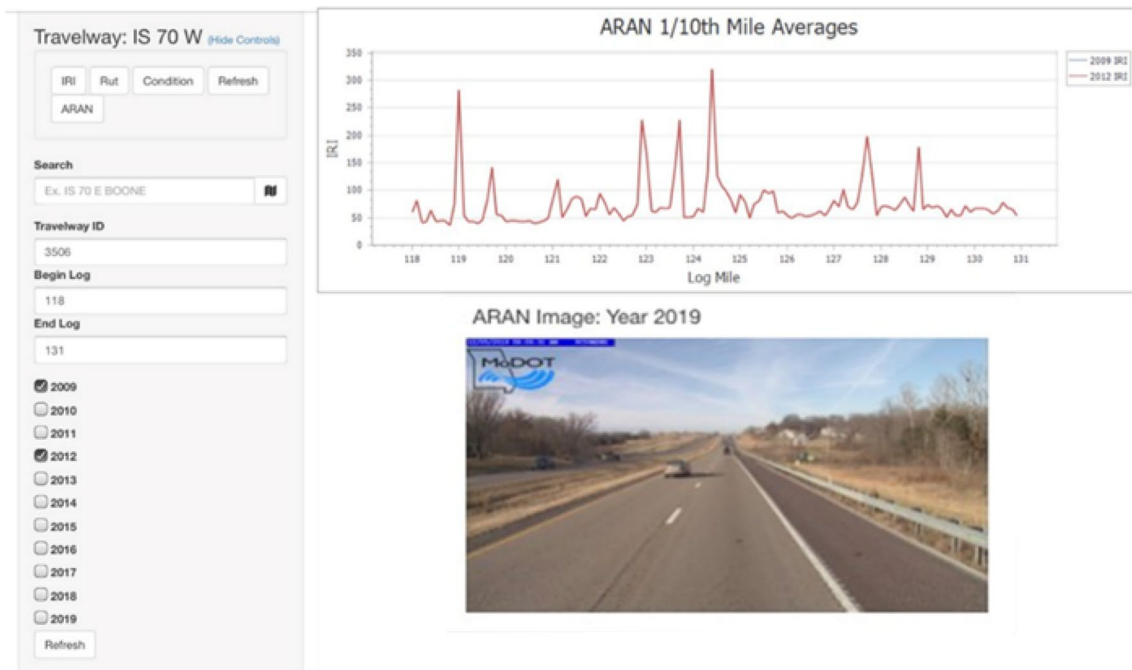
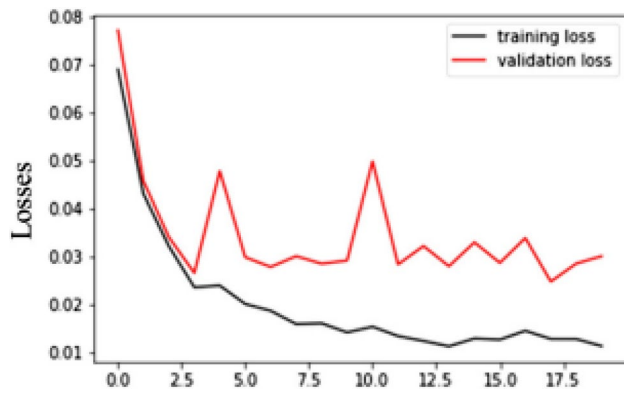


Fig. 7 The MoDOT ARAN viewer portal [8]





**Fig. 8** A plot of training and validation losses of the proposed model

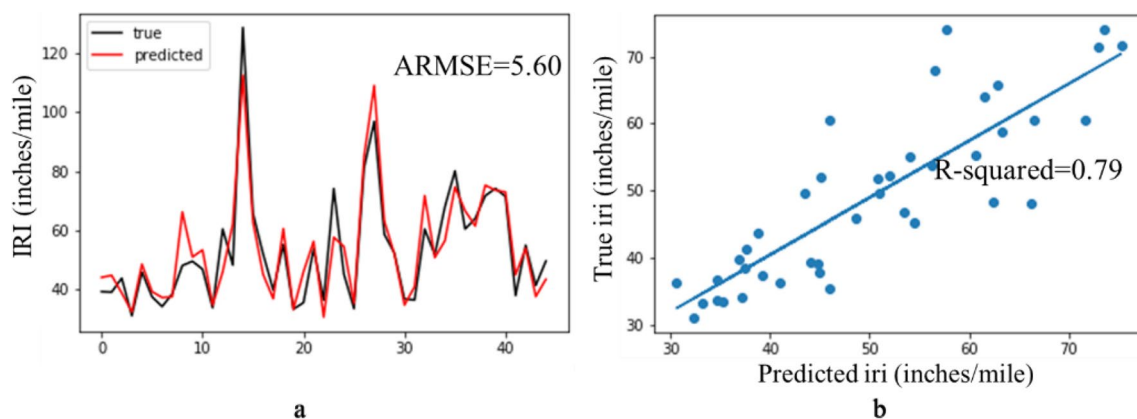
The model was used to estimate the IRI values of the test data. The average-root-mean-squared-error (ARMSE) of the test prediction was 5.6. This means that our predictions are 5.6 units off the mark when compared to the actual IRI estimates provided by road profilers. Furthermore, the root-mean-square-percentage-error (RMSPE) metric indicates that the model's predictions were approximately 17.4% off target IRI values which is within the acceptable range of error (Adeli et al. 2021). As illustrated in Fig. 9a, the road conditions' trends and amplitudes are correctly matched. As illustrated in Fig. 9b, there is a linear relationship between the true and predicted IRI values. The r-squared value of the linear plot was 0.79. This indicates that the true and predicted IRI values are highly correlated.

### Pavement Distress Detection

Pavement distresses pose a potential threat to the safety of road users. As a result, detecting distresses in a timely manner is regarded as one of the most important steps in limiting further degradation of pavement surfaces. To

make the best use of financial resources, it is necessary to assess the condition of pavement surfaces on a regular basis and keep up with maintenance. As a result, the study developed a pavement distress detection model using video images collected by the developed app. We learned the visual and textual patterns associated with the various types of distress using a single-stage object detection algorithm, You Only Look Once (YOLOv5) algorithm.

YOLOv5 is the most recent version of the YOLO series and is a state-of-the-art single-stage object detection algorithm. The YOLOv5 network is divided into three distinct components: the Backbone, the Neck, and the Head. The Backbone is a convolutional neural network that bundles and shapes image representational features at varying granularities. The neck of the architecture is composed of a series of layers that combine and integrate image representational features to advance to prediction. Similarly, the head makes use of neck-derived features and acquires box and class prediction functionality. Within YOLOv5, the CSPDarknet53 backbone contains 29 convolutional layers  $3 \times 3$ , a receptive field size of  $725 \times 725$ , and an overall parameter count of 27.6 M. Additionally, the SPP block attached to YOLO's CSPDarknet53 increases the proportion of receptive fields without impairing its operation. Similarly, feature aggregation is accomplished via PANet, which makes use of multiple backbone levels. YOLOv5 pushes the envelope of efficiency by incorporating features such as weighted residual connections, cross-stage partial connections, cross mini-batch, normalization, and self-adversarial training. In this study, we used the PyTorch [28] framework to train and deploy our YOLOv5 model. To improve the YOLOv5 model's performance in detecting vehicles, the following hyperparameters are adjusted: 64-batch size, 0.0005 decay rate for the optimizer's weights, 0.01 initial learning rate, and 0.937 momentum.



**Fig. 9** Proposed model performance: **a** plot of true IRI values and predicted IRI values and **b** scatter plot of true IRI values against predicted IRI values

The model was trained using 3,000 images obtained from the developed app and tested on 800 images. Four pavement distress types were annotated in this study. They are longitudinal crack (D00), transverse crack (D10), alligator crack (D20), and pothole (D40). The 3,000 images were distributed as follows: D00-900 samples, D10-836 samples, D20-614 samples, and D40-650 samples. We assessed the performance of the model using precision (P), F1 score (F1), and recall value (R). The F-1 score is the harmonic average of the recall and precision values. Precision is defined as the ratio of true positives (tp) to all predicted positives (tp+fp). Similarly, recall is the ratio of true positives to all true positives (tp+fn). The model was trained for 4-h using an NVIDIA GTX 1080ti GPU.

After 1000 iterations, the results summary is shown in Fig. 10. The training and validation losses converged after 500 iterations as shown in Fig. 10a, b, respectively. In addition, the best precision and recall values were obtained around the 500th iteration shown in Fig. 10c, d, respectively.

As shown in Table 1, the precision scores for all distress types range between 0.65 and 0.82, while the recall values range between 0.58 and 0.61. Distress type D00 had the highest precision score of 0.82 and recall value of 0.61. The overall F1 score for the developed model was 0.68.

Figure 11 shows detected pavement distresses from road images. Three pavement distresses were detected in this figure. They are designated as longitudinal crack (D00-red bounding box), alligator crack (D20-blue bounding box), and pothole (D40-green bounding box). The results show that the app can produce high-resolution videos images that can be used by the current generation of machine learning algorithms for pavement evaluation.

## Naturalistic Driving Studies

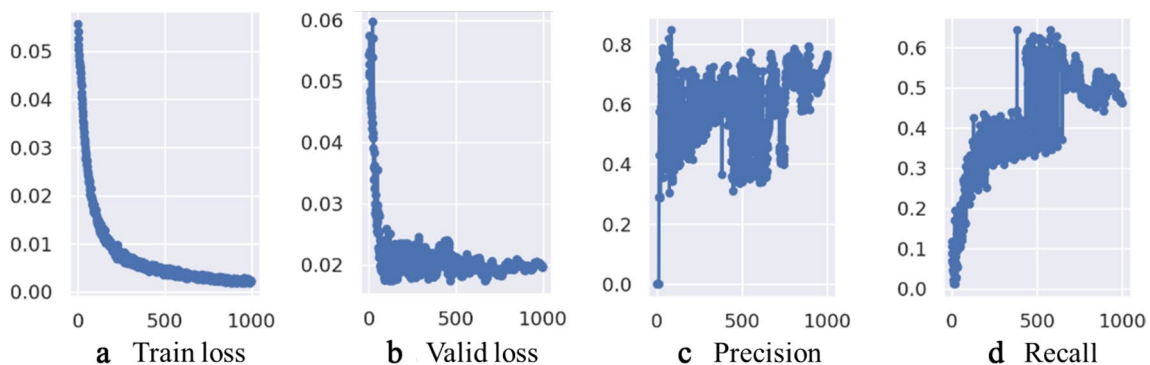
Naturalistic driving studies (NDS) are cutting-edge research techniques that involve continuously recording driving data in real-world driving conditions using advanced

**Table 1** Precision, recall, F1-score values for the various distress types

Distress type	Precision	Recall	F1-score
D00	0.82	0.61	0.70
D10	0.73	0.59	0.65
D20	0.65	0.60	0.62
D40	0.67	0.58	0.62

instrumentation. NDSs enable the assessment of driving risks that would otherwise be impossible to assess using traditional crash databases or experimental methods (Guo 2019). The NDS findings have a significant impact on policy-making, safety research and the development of safety countermeasures. The developed application in this study collects NDS data using iPhone's dual camera system. One camera monitors the driver's environment outside the vehicle, while the other is used to record the driver's activities inside the vehicle as illustrated in Fig. 12a,b. This setup enables us to visually identify traffic incidents and correlate them to specific driving behaviors (driver sleeping, distracted, using phone, etc.).

The objective of this task is to develop a model for automatically detecting driving events from naturalistic driving videos. To achieve this objective, we formulated the problem as a time series segmentation and classification problem. The segmentation task was achieved by developing a novel segmentation algorithm that utilizes the principle of energy maximization to detect the start and end of any driving event from the telemetry data (gyroscope in the z-direction). This step was necessary to facilitate and expedite the annotation of driving events. All annotated driving events were standardized to a fixed length before feeding it to the model. Seven main classes of events were defined: lane changing (left and right), driver stopped, left turns, left curves, right turns, right curves, and lane-keeping. The model was trained with 86,000 training samples using NVIDIA GTX 1080ti



**Fig. 10** Results from model training **a** training loss, **b** validation loss, **c** precision and **d** recall



**Fig. 11** Pavement distresses detected from video data

GPU. We used a 50:50 split for model training and testing. The training data was fed through a multi-layered LSTM consisting of 30 hidden layers and 5 LSTM layers.

A convergence plot of the training and validation loss is shown in Fig. 13. After 4000 iterations, the model's accuracy was approximately 94%. This implies that 94% of the time, the model can predict accurately the driving maneuvers of a vehicle. The training and validation losses of the developed model were 0.1402 and 0.2703, respectively. This means that the model is overfitting the training data, but it is not able to generalize correctly to validation data.

The developed models were evaluated using accuracy and precision. Accuracy was computed as the ratio of the true positive (TP) predictions to the sum of true positives (TP) and true negative (TN) predictions expressed as a percentage. Precision, on the other hand, was computed as the number of true positives divided by the total number of true positives and false positives predictions.

From Table 2, precision scores range between 0.889 and 0.978 for all types of driving maneuvers, while recall values range between 0.898 and 0.962. Left turns achieved the highest precision of 0.978, while right turns achieved the highest recall of 0.962. Right lane changes, on the other hand, had the lowest precision score of 0.889, while left lane changes had the lowest recall score of 0.898. Overall, the model received an F1 score of 0.936. The precision, recall,

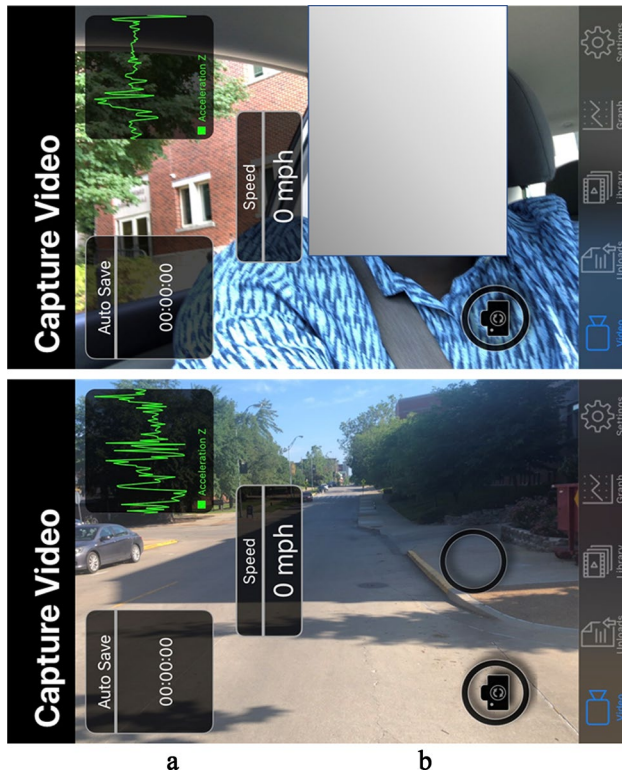
and f1 score of all driving maneuvers are summarized in Table 2.

## Conclusion

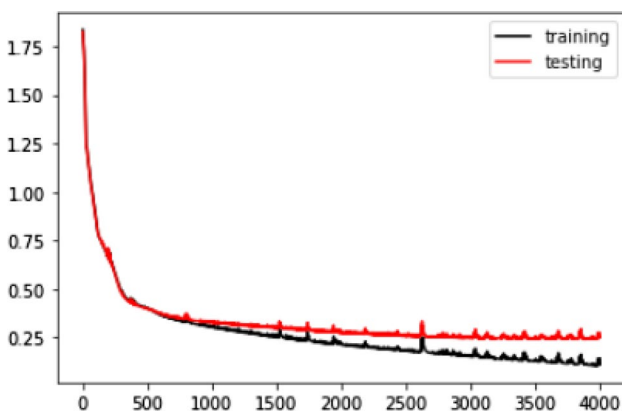
This study developed and deployed a mobile application for collecting data for multiple applications including road condition evaluation, naturalistic driving studies, etc. The app collects road surface data via in-built smartphone sensors. The data can be streamed directly to a cloud-based database or stored in the app's library and uploaded to cloud storage at a later time if Wi-Fi is unavailable. The app's settings allow users to change the sampling frequency and frame rate during data collection. Additionally, while driving, the app is able to display the accelerometer readings on the video tab interface.

The application was developed using a modular approach. The design framework of this app is composed of three major modules: a frontend user interface (UI) component, a sensor component, and a backend component.

The app's functionality was evaluated by collecting data on the road surface on the I-70 W highway connecting Columbia, Missouri and Kansas City. The collected data was used to build a predictive model for estimating IRI values using a deep learning architecture. The accuracy of the



**Fig. 12** **a** Back camera recording activities outside the vehicle. **b** Front camera recording activities inside the vehicle (face of driver masked)



**Fig. 13** A plot of training and validation losses of the Proposed model

model was used as a proxy for the quality of data collected by the app. ARMSE, R-squared, and RMSPE were used as metrics for evaluation. The model predictions indicate that the predicted IRI values from the smartphone data are comparable to those estimated using high-end machines such as the ARAN van. When the predicted IRI values were compared to the ground truth IRI values, a goodness-of-fit value of 0.79 was obtained. This demonstrates a high degree of

**Table 2** Precision, recall, F1-score values for the various driving events

Driving maneuvers	Performance matrix		
	Precision	Recall	F1 score
Right turn	0.945	0.921	0.932
Left turn	0.978	0.942	0.960
Right curve	0.969	0.962	0.965
Left curve	0.943	0.929	0.936
Right lane change	0.889	0.901	0.895
Left lane change	0.907	0.898	0.902
Lane keeping	0.949	0.938	0.943
Stop	0.961	0.944	0.952

correlation between them. Also, the video information from the app was used to identify pavement distresses. Finally, information from the gyroscope and the accelerometer readings were used to determine the turning and lane-changing maneuvers of vehicles.

In a nutshell, when tested, the developed mobile application for collecting road surface information and estimating road surface roughness demonstrated a high degree of potential for producing accurate and reliable results.

## Limitations and Recommendations

The developed roughness app's future updates should address the following limitations.

- The first constraint is that the app was designed exclusively for the iOS operating system. Given a large number of Android users, the app's next update should include support for the Android operating system.
- Additionally, when used, the app does not provide IRI values directly. Future updates should incorporate the developed deep learning model into the backend, allowing the app to directly predict the IRI value of road sections in real-time while in use.
- Another limitation observed is that the GPS coordinates were approximately 6.6 m off. Next update of the app will have an improved GPS.

**Author Contributions** Each of the three authors made equal contribution to: conceptualization, conducting experiments, analysis of the experimental data, writing, and reviewing the manuscript.

**Funding** This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

**Data Availability** Some or all data, models, or code that support the findings of this study are available from the corresponding author upon reasonable request.

## Declarations

**Conflict of interest** On behalf of all authors, the corresponding author states that there is no conflict of interest.

## References

- Aboah A, Adu-Gyamfi Y (2020). Smartphone-based pavement roughness estimation using deep learning with entity embedding. World Scientific Publishing Company 12(03n04):2050007. <https://doi.org/10.1142/S2424922X20500072>
- Aleadelat W, Asce SM, Ksaibati K, Cameron, Wright HG, Saha P (2018) Evaluation of pavement roughness using an android-based smartphone. *ascelibrary.org*, Polskie Towarzystwo Stomatologiczne 144(3)
- Azadi F, Adu-Gyamfi Y, Sun C, Edara P (2020) Mobile application development and testing for work zone activity real-time data collection. *Transp Res Rec* 2674(6):351–362
- Adeli S, Najafi moghaddam Gilani V, Kashani Novin M, Moteshareei E, Salehfard R (2021) Development of a relationship between pavement condition index and international roughness index in rural road network. *Adv Civ Eng*
- Craig C, Achtemeier J, Morris N, Tian D, Patzer B (2017) In-vehicle work zone messages
- Dawkin J, Bevely D, Powell B, Bishop R (2010) Investigation of pavement maintenance applications of intellidrive SM, CTS PFS Report
- Dhatbale R, Chilukuri BR (2021) Deep learning techniques for vehicle trajectory extraction in mixed traffic. *J Big Data Anal Transp* 3(2):141–157
- DOT Datazone. Accessed 05/05/2020. Available at <https://datazoneap.ps.modot.mo.gov/aran/Home/Index?twid=3506&log=118>.
- Dutzik T, Madsen T, Baxandall P (2013) A new way to go: the transportation apps and vehicle-sharing tools that are giving more Americans the freedom to drive less
- Efthymiou A, Barmponakis EN, Efthymiou D, Vlahogianni EI (2019) Transportation mode detection from low-power smartphone sensors using tree-based ensembles. *J Big Data Anal Transp* 1(1):57–69
- Flintsch GW, Valeri SM, Katicha SW, de Leon Izeppi ED, Medina-Flintsch A (2012) Probe vehicles used to measure road ride quality: Pilot demonstration. *Transp Res Rec Natl Res Council* 2304:158–165
- GitHub link: <https://github.com/ultralytics/yolov5>. Accessed 4 Nov 2021
- Guo F (2019) Statistical methods for naturalistic driving studies. *Annu Rev Stat Appl* 6:309–328
- Li Q, Qiao F, Proc LY-Congress, 23rd ITS World, and 2016 (2016) Performance measures of smartphone warning messages in work zones and intersections. *researchgate.net*
- Liao CF (2014) Development of a navigation system using smartphone and bluetooth technologies to help the visually impaired navigate work zones safely
- Mandal V, Mussah AR, Adu-Gyamfi Y (2020) Deep learning frameworks for pavement distress classification: a comparative analysis. *arXiv preprint arXiv:2010.10681*
- Miao H, Zhang S, Flannagan C (2022) Driver behavior extraction from videos in naturalistic driving datasets with 3D ConvNets. *J Big Data Anal Transp* 4(1):41–55
- Pal M, Sutradhar R (2014) Pavement roughness prediction systems: a bump integrator approach. *Int J Civ Environ Eng* 8(12):1258–1261
- Papadimitriou E, Tselentis DI, Yannis G (2018) Analysis of driving behaviour characteristics based on smartphone data. *Proceedings of 7th Transport Research Arena TRA*, 16–19
- Rahman R, Fengxiang Qiao M, Li Q, Yu L (2016) Developing a smartphone based warning system application to enhance the safety at work zones
- Robinson R, Cook SJ (2012) Slippery road detection and evaluation Sauerwein PM, Smith BL, Studies U. of Virginia. C. for T. (2011) Investigation of the implementation of a probe-vehicle based pavement roughness estimation system. Virginia Transportation Research Council
- Streamlit. Available at <https://docs.streamlit.io/en/stable/api.html>. Accessed 01 Oct 2020
- Zeng H, Park H, Smith B, Engineering EP-KJ (2018) Feasibility assessment of a smartphone-based application to estimate road roughness. *Springer Verlag* 22(8):3120–3129

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.