WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Road Watch: Real-Time Pothole Detection and Rider Alert System Using a Website and a Pretrained Deep Learning Model

B Varun Kumar[1], J Renugadevi[2], R Swaruppa[3], P Supriya[4]

Sri Shakthi Institute of Engineering and Technology, Chinniyampalayam, L&T Bypass, Coimbatore, India

A B S T R A C T :

Potholes are a major cause of two-wheeler accidents and vehicle damage, yet riders usually notice them only at the last moment, and road authorities depend on slow manual complaints and periodic surveys. This paper presents Road Watch, a website that turns an ordinary smartphone into a pothole sensor. While the rider is travelling, the phone's back camera is accessed through the browser and one frame is captured every three seconds instead of processing continuous video, which keeps battery and data use low. Each frame is checked by a pretrained YOLO-based pothole detection model obtained from Roboflow, so no model training is needed. When a pothole is found, GPS coordinates are attached, the rider is alerted instantly, and the detection is saved in a Supabase database and shown on a shared Leaflet hazard map so that other riders approaching the same spot are warned in advance. The system is built with React (Vite), Node.js with Express and Supabase. The login, dashboard, ride-start and live camera screens with three-second frame capture are implemented, while model integration, GPS tagging, the hazard map and proximity alerts are in progress. The paper describes the design, the implementation status and the planned evaluation.

Keywords: Pothole detection; Website; YOLO; Road safety; Crowdsourced hazard map; Supabase

Introduction

Road surface damage, and potholes in particular, is a persistent safety problem on Indian roads. A rider travelling at normal speed often sees a pothole only a moment before reaching it, especially at night, in low light, or when the hole is filled with rainwater. The result can be sudden braking, loss of control, damage to tyres and suspension, and in the worst cases an accident. Road authorities, meanwhile, depend on complaints filed by citizens and on periodic manual surveys, which are slow, costly and quickly become outdated.

A smartphone already contains a camera, a GPS receiver and a capable web browser, which together are enough to detect and report road damage automatically. If the phones of many riders contribute what they see, a continuously updated map of road hazards can be built without any special hardware. This idea motivates Road Watch, a real-time pothole detection and alert system delivered as a website that runs in the browser. The scope of the present work is limited to potholes; other forms of road damage such as cracks and waterlogging are left for future work.

Nomenclature

API	Application Programming Interface
GPS	Global Positioning System
YOLO	You Only Look Once (object detection model family)

1.1. *Problem statement*

Riders receive no advance warning of potholes. The existing options are manual complaint portals, which depend on a person typing a report after the event, and occasional government surveys. Neither gives real-time, location-tagged information to the rider who is about to reach the damaged spot. There is therefore a need for an automatic system that detects potholes while riding, warns the rider immediately and shares the detection with other riders.

1.2. *Objectives*

The objectives of this work are:

- To capture road images from the phone's back camera inside a web browser, without installing any software.
- To detect potholes in the captured frames using a pretrained deep learning model.

- To tag each detection with GPS coordinates and alert the rider instantly.
- To store detections in a database and display them on a shared hazard map.
- To warn other riders when they approach a previously detected pothole.

Related work

Early pothole sensing systems relied on vibration data. The Pothole Patrol project mounted accelerometers and GPS units on taxis to detect and map road anomalies (Eriksson et al., 2008), and Mednis et al. (2011) showed that the accelerometer of an Android smartphone can be used for real-time pothole detection. Such sensor-based methods are inexpensive and work in the dark, but they respond only when the vehicle has already driven over the pothole, so they cannot warn the rider in advance.

Vision-based approaches use images instead. Maeda et al. (2018) trained deep neural networks on smartphone images captured from a moving vehicle to detect and classify road damage, and Arya et al. (2021) extended this work to road damage data from several countries. These studies show that camera images are sufficient for damage detection; their focus, however, is detection accuracy rather than a complete rider-facing service with alerts and a shared map.

Single-stage detectors such as YOLO (Redmon et al., 2016) predict object locations and classes in a single pass, which makes them fast enough for near real-time use. Maintained implementations such as Ultralytics YOLOv8 (Jocher et al., 2023) and public pretrained models on Roboflow Universe (Roboflow, n.d.) make it practical for a small team to use such a detector without training one from scratch. Road Watch combines browser-based frame sampling, a pretrained detector, rider alerts and a shared hazard map in a single website.

Proposed system

1.3. *System architecture*

Road Watch follows a client–server architecture, shown in Fig. 1. The rider's phone runs the website front end, which accesses the camera through the getUserMedia API (Mozilla, n.d.) and the position through the browser Geolocation API. Captured frames are sent to a Node.js and Express backend, which passes them to the pretrained detection model and returns the result. Confirmed detections are stored in Supabase (Supabase, n.d.), and the hazard map and the proximity alerts read from the same database.

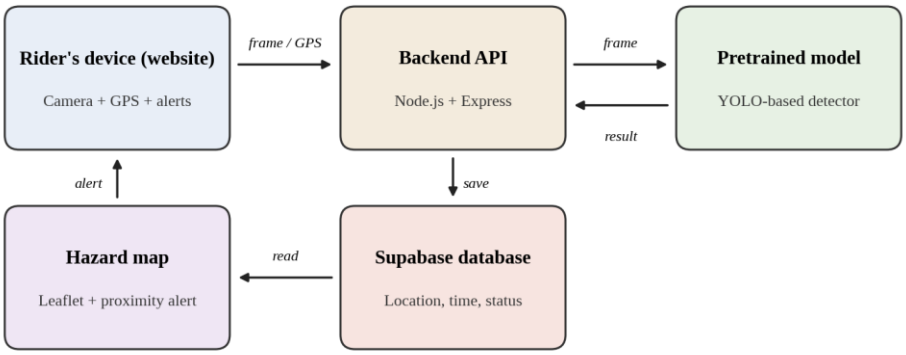


Fig. 1 - Architecture of the Road Watch system.

1.4. *Working methodology*

The system works as a loop that runs for the whole ride:

- The rider logs in and taps Start Ride; the browser asks for permission to use the back camera and the location.
- Every three seconds a frame is drawn from the live video onto an HTML5 canvas and encoded as a JPEG image.
- The frame is sent to the backend, where the pretrained model checks whether it contains a pothole.
- If no pothole is found, the frame is discarded and the loop continues with the next frame.
- If a pothole is found, the GPS coordinates and the time are attached, the rider is alerted by an on-screen message and a sound, and the record is saved in Supabase.
- The saved detection appears on the shared hazard map; when another rider comes within a set distance of it, the same alert is raised in advance.

1.5. *Frame sampling and pretrained model*

Processing continuous video on a phone would drain the battery quickly and demand a large amount of data and computation. Sampling one frame every three seconds greatly reduces this load while still giving the rider a live-checking experience. At 40 km/h a vehicle travels about 33 m in three seconds, so a short pothole may fall between two frames; this trade-off is discussed in Section 6. Training a detector from scratch needs thousands of labelled pothole images and weeks of effort, so Road Watch instead uses a pretrained model published on Roboflow Universe and only connects it to the website.

1.6. *Technology stack*

The technologies used for each part of the system are listed in Table 1.

Table 1 - Technology stack of Road Watch.

Layer	Technology	Purpose
Front end	React with Vite (website)	User interface running in the browser
Camera and frames	getUserMedia API, HTML5 Canvas	Back-camera access, frame capture
Backend	Node.js with Express	Receives frames, calls the model
AI model	Pretrained YOLO-based model (Roboflow)	Pothole detection
Database	Supabase	Stores detections with location and time
Location and map	Geolocation API, Leaflet.js	GPS tagging, hazard map

Implementation status

1.7. *Front end*

The front end is built with React and Vite and uses React Router for navigation. Four screens are implemented: a login screen; a dashboard with summary cards and buttons for starting a ride and opening the hazard map; a Start Ride screen with instructions for mounting the phone; and a Live Detection screen. The Live Detection screen opens the back camera with getUserMedia, shows the live feed, and captures a frame to a hidden canvas every three seconds while displaying a running frame count. Each frame is posted to the backend, and a red alert banner is shown when the response reports a pothole.

1.8. *Backend prototype*

A Node.js and Express server exposes a /api/check-frame endpoint that accepts a base64-encoded frame and returns a JSON object with a pothole flag and a confidence value. At present this endpoint is a mock that returns a random result. It allows the complete flow from camera to alert to be tested before the real model is connected.

1.9. *Module status*

Table 2 summarises the state of each module at the time of writing.

Table 2 - Implementation status of the modules.

Module	Status
Login, dashboard and Start Ride screens	Implemented
Camera access and 3-second frame capture	Implemented
Mock detection endpoint and alert banner	Implemented (mock)
Real pretrained model integration	Pending
GPS tagging of detections	Pending
Hazard map page (Leaflet)	Pending
Supabase integration	Pending
Proximity alerts	Pending

Planned evaluation and expected outcome

No accuracy results are reported yet, because the real detection model has not been integrated. Once it is connected, the system will be tested on real roads with the phone fixed on a handlebar mount. The planned measures are the precision and recall of pothole detection on captured frames, the end-to-end delay from frame capture to alert, the rate of false alerts, the battery consumed over a ride, and the share of known potholes that are correctly warned about when the same route is ridden a second time.

The expected outcome is a working website in which a rider receives a warning before reaching a detected pothole, and a shared map shows where potholes have been found, giving road authorities location-tagged data on where repairs are most needed.

Limitations and future work

Sampling one frame every three seconds may miss short potholes at higher speeds; the interval could be shortened or made dependent on speed. Camera-based detection also depends on lighting, camera angle and motion blur, and water-filled potholes are hard to see. Browsers may restrict camera access when the screen is locked or the website runs in a background tab, so the phone has to stay on during the ride, and alerts should be audible so that the rider need not look at the screen. A single detection may also be a false positive, so a pothole could be marked as confirmed only after several riders report the same location.

Future work includes severity classification (minor and major damage), detection of other road damage such as cracks, combining camera detection with accelerometer data, and integration with road-maintenance systems of the authorities.

Conclusion

This paper presented Road Watch, a website that uses a phone's back camera, a pretrained YOLO-based model, GPS and a Supabase database to detect potholes while riding, warn the rider and build a shared hazard map. The front end and the camera frame-capture pipeline are in place, and a mock backend verifies the flow from camera to alert. The remaining work is to integrate the real model, GPS tagging, the hazard map, Supabase storage and proximity alerts, and then to evaluate the system on real roads.

Acknowledgements

The authors thank the management and faculty of Sri Shakthi Institute of Engineering and Technology, Coimbatore, for the facilities and support provided for this work.

REFERENCES

- Agafonkin, V. (n.d.). *Leaflet: An open-source JavaScript library for mobile-friendly interactive maps*. https://leafletjs.com
- Arya, D., Maeda, H., Ghosh, S. K., Toshniwal, D., & Sekimoto, Y. (2021). Deep learning-based road damage detection and classification for multiple countries. *Automation in Construction, 132*, 103935.
- Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., & Balakrishnan, H. (2008). 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* (pp. 29–39). ACM.
- Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLOv8* [Computer software]. https://github.com/ultralytics/ultralytics
- Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., & Omata, H. (2018). Road damage detection and classification using deep neural networks with smartphone images. *Computer-Aided Civil and Infrastructure Engineering, 33*(12), 1127–1141.
- Mednis, A., Strazdins, G., Zviedris, R., Kanonirs, G., & Selavo, L. (2011). Real time pothole detection using Android smartphones with accelerometers. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)* (pp. 1–6). IEEE.
- Mozilla. (n.d.). *MediaDevices: getUserMedia() method*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779–788).
- Roboflow. (n.d.). *Roboflow Universe*. https://universe.roboflow.com
- Supabase. (n.d.). *Supabase documentation*. https://supabase.com/docs