Proceedings of the International Conference on Industrial Engineering and Operations Management

Publisher: IEOM Society International, USA DOI: 10.46254/BA08.20250361
Published: December 20, 2025

Development of an Adaptive Speed Control System for Vehicles on Rough Surfaces

Abdullah Al Rakib
Department of Mechatronics Engineering
Khulna University of Engineering and Technology (KUET)
Khulna-9203, Bangladesh
rakib1931014@stud.kuet.ac.bd

Asief Javed
Department of Mechatronics Engineering
Khulna University of Engineering and Technology (KUET)
Khulna-9203, Bangladesh
asiefjaved@mte.kuet.ac.bd

Abstract

Road anomalies such as potholes and speed bumps pose significant challenges to driver safety and passenger comfort. Despite modern vehicles increasingly employing sophisticated sensors to mitigate such hazards, existing frameworks and infrastructural constraints make these solutions exceedingly expensive. This paper suggests a low-cost, small, and modular system that could be deployed in smart cars, uses open-source tools and readily available hardware to detect road irregularities in real-time, and adjust the vehicle's speed appropriately. The core of the system is a custom-trained, lightweight YOLOv8m object detection model to recognize road anomalies with high accuracy and optimized to run inference on limited hardware. The overall framework comprises two intertwined nodes: Raspberry Pi acts as the vehicle's hardware controller, managing GPIO-driven dual motor control, allowing for smooth speed control, and hosting a Flask-based streaming server. Meanwhile, a remote pc connected to the same network runs a detection script that takes the live video stream of the Raspberry Pi, detects objects within every single frame, and sends real-time control commands using the MQTT message protocol. The system GUI also provides a visual interface for the user to manually start or stop the vehicle. To ensure smooth and reliable operation, buffered detection logic, confidence threshold, speed ramping, and falsification mechanism were also added to the speed controller system. Experimental results demonstrate that the system performs well in real-world situations. The performance was evaluated based on detection accuracy, inference time, latency, and responsiveness of the speed control module.

Keywords
Road Anomalies, YOLOv8m, MQTT, Flask Server, Speed Ramping

1. Introduction
Roads are the most important mode of transportation for a country to provide countrywide commuting services. Roads are typically constructed of asphalt pavement and are susceptible to various structural problems over time. Potholes, cracks, and unwanted speed bumps can pose significant safety risks to vehicles and passengers. Roads are damaged by several reasons, such as weather, heavy vehicles, construction techniques, construction materials, and lack of maintenance. Roads with potholes are dangerous for commuters and increase the likelihood of serious traffic accidents. Unwanted and poorly constructed speed bumps also cause a great deal of discomfort for passengers. According to official records, 5024 persons died in road accidents in 2023, averaging 14 deaths a day (Source: Road Safety

Foundation and BRTA). BUET's road accidents research institute says accidents can occur due to frequent ruts and potholes, construction flaws of speed bumps, and the absence of proper signs and signals.

Autonomous and semi-autonomous car technology has improved rapidly, but effective operation on roads with varying, unpredictable surface conditions remains a significant problem since anomalies impair stability, handling, fuel efficiency, and passenger comfort. To perceive and react to such conditions in real time, modern AVs require sophisticated algorithms, sensors, and processors. Advances in machine vision and intelligent control make adaptive speed control—detecting road abnormalities and dynamically adjusting speed—feasible while contributing to safety and efficiency. A variety of techniques (computer vision, GPS, onboard sensors, mapping, and object recognition frameworks) can be combined to detect impediments, analyze conditions, and plan responses (G. Bathla et al., 2022).

Unexpected and unidentified road hazards can cause serious accidents; local media reports that many speed bumps are built arbitrarily (Daily Sun, 2025), and potholes are a major contributor to crashes in Bangladesh (Yousuf et al., 2021). Current detection techniques hardly interact dynamically with vehicle controllers, and conventional speed-control systems lack real-time adaptation to rapid road changes, requiring a trade-off between vehicle integrity and passenger comfort. As a result, autonomous cars require rapid, accurate anomaly detection and response; an adaptive system that detects road anomalies in real time and adjusts vehicle speed is required to decrease impact and increase safety.

1.1 Objectives

The primary aim of this thesis is to design and implement an adaptive speed control system for vehicles on rough surfaces. To achieve this aim, the following objectives are defined: 1) Develop a system that combines real-time motor speed control and road anomaly detection on a vehicle prototype based on computer vision. 2)Implement an alert system for drivers to notify them about the rough terrain.

2. Literature Review

This literature review section includes all pertinent publications from various research studies relevant to this paper that helped to comprehend several distinct aspects of the issues.

Convolutional neural networks have dominated current research because they combine detection accuracy and real-time throughput. For on-vehicle application, single-stage detectors (YOLO family, SSD) are preferred because of their low latency, but two-stage detectors (Faster R-CNN) usually produce higher accuracy at a larger computational cost. Lightweight YOLO variants, particularly Tiny-YOLOv4, provide the best real-time trade-off, according to comparative studies: Asad et al. (2022) found that Tiny-YOLOv4 achieved nearly 96% detection effectiveness with high FPS in comparison to heavier YOLO versions and SSD-Mobilenetv2, and Park et al. (2021) found that YOLOv4-tiny produced the highest mAP@0.5 (78.7%) among YOLOv4, YOLOv5s, and YOLOv4-tiny on a dataset of 665 images. According to Ahmed et al. (2021), two-stage Faster R-CNN with robust backbones (such as ResNet50) can outperform one-stage models in accuracy (reported ~91.9% in comparison testing), but it is less suitable for hardware constraints and needs significantly more training effort.

Earlier approaches for detecting anomalies include thresholding, morphological/connected-component analysis, steerable and Gaussian filters, or SVM variations. These lack robustness and generalization, although they perform well on tiny, controlled datasets. For instance, Nhat-Duc & Hoang (2018) used LS-SVM on 200 pictures to reach ≈89% accuracy and AUC ≈0.96 after substantial pre-processing; nevertheless, practical deployment is limited by the small dataset and computational expense of LS-SVM.

Recent research has combined lightweight detectors with multi-scale feature modules (FPN, SPP) to improve small/irregular defect detection and edge compatibility. In order to improve safety relevance, Dong-Hoe Hao et al. (2023) suggested SPFEN-YOLOv4-tiny (FPN+SPP+YOLOv4-tiny) and included a 2-D risk assessment that predicts severity from tire-contact area vs pothole size. However, this approach is still susceptible to unfavorable lighting and visibility conditions.

Monocular/stereoscopic cameras, ZED stereo rigs, LIDAR, and vehicle IMUs are used in the research. Depending on the sensor, model, and dataset, reported accuracy ranges from approximately 77% to 97%. While LIDAR or sensor-fusion techniques increase robustness at a higher cost, recent image-based CNNs like MobileNet-SSD and Tiny-YOLOv4 report the best practical accuracies and FPS for real-time settings. Many research relies on hand-annotated

datasets of a few hundred photos, which limits diversity and generalization (Park et al., 2021; Peralta Lopez et al., 2023; Asad et al., 2022).

Table 1. Summary of Approaches to Detect Speed Bumps and Potholes

Method	Type of Detection	Sensor	Accuracy	Contributions
Support vector machine	Road anomaly	Accelerometer	78.5%	JY Hsu et al. (2010)
Height difference-based algorithm	Pedestrian crossing and speed bump	Image and LIDAR	85%	Yi. K. et al. (2012)
Color image thresholding	Speed bump	Image	85%	Devapriya et al. (2015)
Connected component analysis	Speed bump	Image	92%	Devapriya et al. (2016)
Mobilenet-SSD CNN model	Speed hump/bump	Image (ZED)	97.4%	Peralta Lopez et al. (2023)
Inception V2	Pothole	Image	77%	Maeda et al. (2018)
Otsu thresholding	Speed hump/bump	Image	90%	Babu et al. (2020)
Tiny-YOLOv4	Pothole	Image	90%	Asad et al. (2022)
YOLO & Potholes and bumps	Potholes and bumps	Image	88.9%	Shah et al. (2019)

Table 1 lists the summary of approaches to detect road anomalies. The majority of current research on road anomaly detection focuses on applying deep learning or image processing approaches to detect potholes or speed bumps. These methods do not, however, extend to real-time integration with vehicle control systems and are frequently restricted to offline analysis or dataset evaluation. Also, the deployment of detection models on resource-constrained embedded hardware has been the subject of very few studies, despite the fact that this is essential for realistic vehicle applications in cost-sensitive settings. Previous studies have also overlooked fail-safe measures and driver awareness.

3. Methodology

This chapter outlines the specific approach of the suggested analysis for autonomous vehicles to detect real-time items such as speed bumps, potholes and reduce speed accordingly.

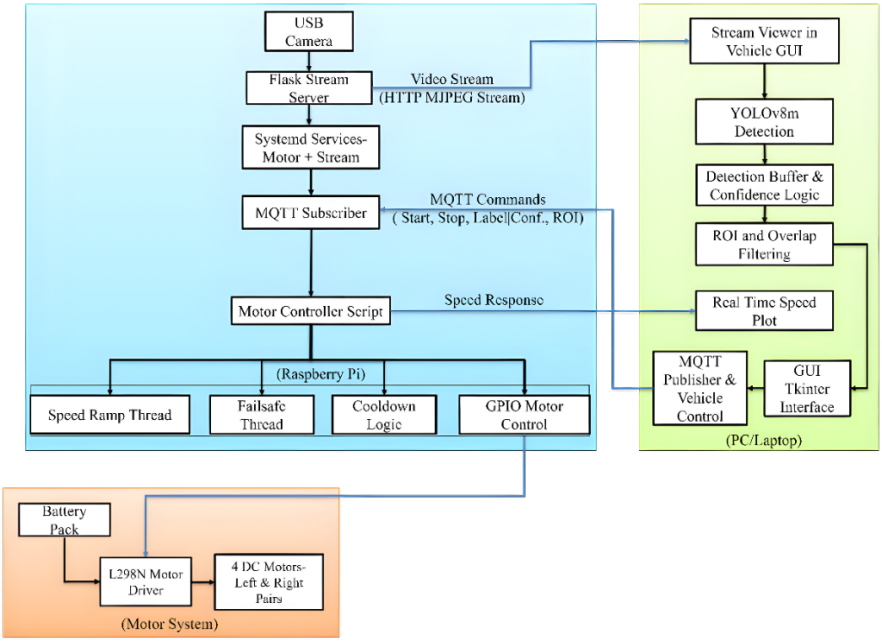

Figure 1. Flowchart of the System Architecture

Figure 1 shows the architecture of the developed system, which comprises two main parts: Remote PC running the detection model in real time on the HTTP stream coming from the webcam in the vehicle, a Raspberry Pi that streams the road in front of the vehicle via a Flask stream server and runs the motor controller script based on the MQTT command received from the PC.

3.1 Deployment Flow

Figure 2 shows the system deployment flowchart of the Flask stream in the Raspberry Pi.

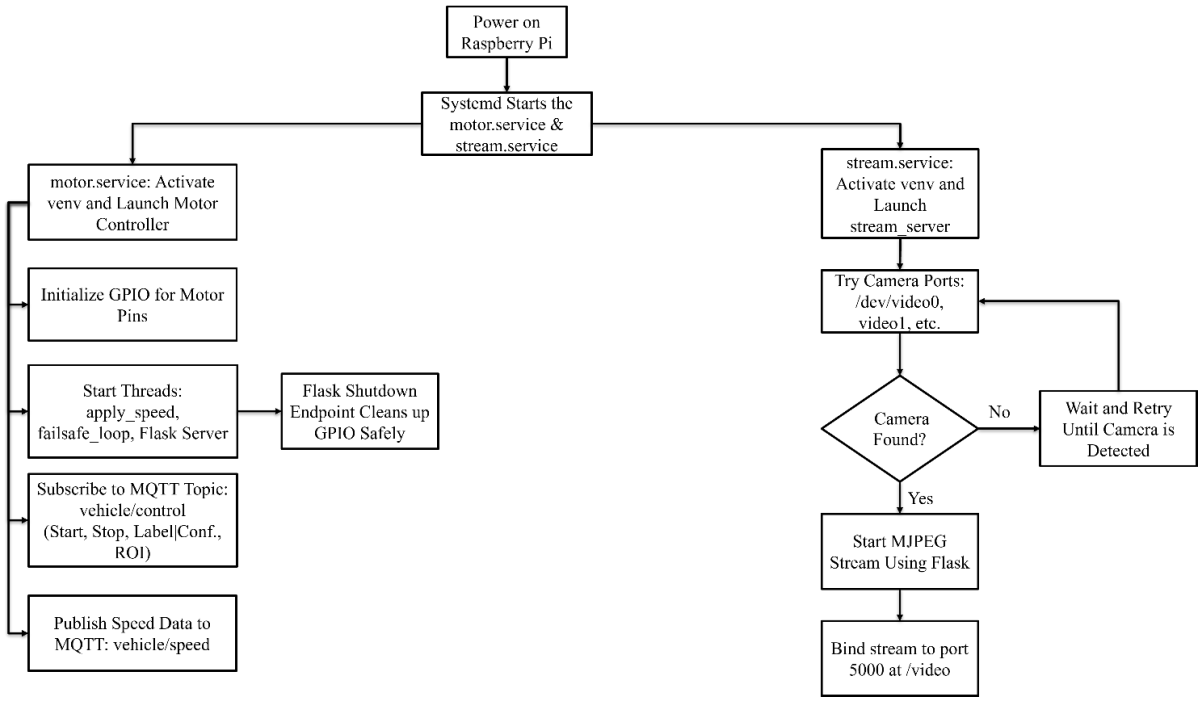

Figure 2. Flowchart of the System Deployment (Flask Stream)

1) System Initialization

a. Raspberry Pi boots and systemd auto-starts: Flask video stream (stream.service), Smart motor controller (motor.service)
b. All GPIO pins are initialized.
c. PWM starts on ENA and ENB at 0\% duty cycle.
d. MQTT client and Flask /shutdown endpoint are initialized.
e. Three background threads are launched: apply speed () – handles smooth speed transitions, failsafe loop () – restores cruise speed if no objects are detected, Flask – for optional remote shutdown.

2) Idle State (Awaiting Command)

a. Motors remain OFF (running = False).
b. Controller waits for an MQTT command.

3) Receiving MQTT Message

a. The controller listens to vehicle/control commands, and incoming messages are parsed.
b. "start" → enables motor (running = True), sets target speed to 75%.
c. "stop" → stops motor immediately, resets running = False.
d. "Pothole confidence " or "speedbump confidence " with ROI → passes to detection buffer logic.

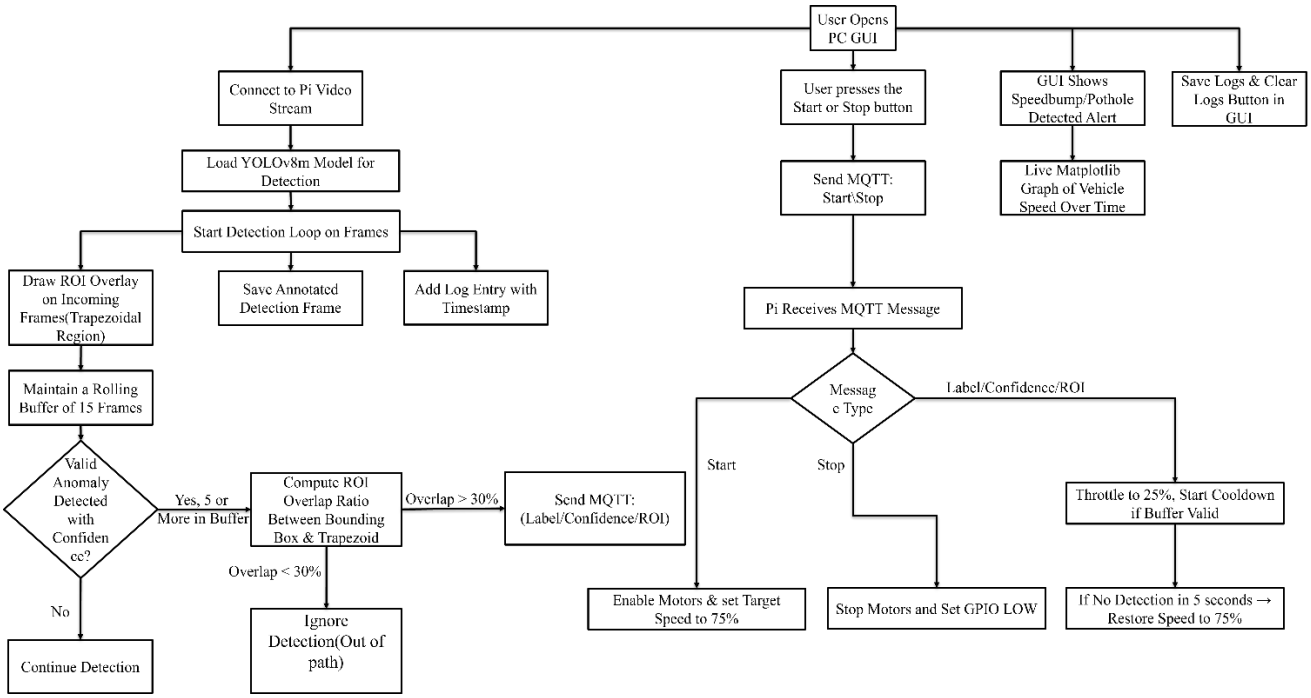

Figure 3. Flowchart of the System Deployment (PC Side)

Figure 3 shows the deployment flow on the PC side, comprising the GUI and road anomaly detection model running in real time.

4) Buffered Detection Handling

a. Every detection is stored in a 15-frame rolling buffer.
b. Only acts if: Label confidence and ROI overlap meet the threshold (pothole = 0.6, speed bump = 0.45, ROI = 40%) or Label appears 5 or more times in buffer or Cooldown period (2 seconds) has passed. “Pothole confidence " or "speedbump confidence " with ROI → passes to detection buffer logic.
c. If valid: Sets target speed = 25 and Cooldown starts to prevent rapid toggling.

5) Smooth Speed Transition

a. apply speed() thread constantly adjusts actual motor speed (current speed).
b. Steps +1 or -1 toward target speed every 100ms.
c. Ensures smooth acceleration or braking and avoids jerks.

6) Failsafe recovery

a. failsafe loop() checks if no detection occurred for more than 5 seconds.
b. If so, restores cruising speed: target speed = 75, only if the motor is still running.

7) Flask Shutdown Endpoint

a. Accessing http://pi ip:8080/shutdown: Calls stop motor(), Performs GPIO.cleanup(), shuts down the motor controller if needed.

3.2 Communication Layer

The communication between the PC (detection unit) and the Raspberry Pi (control unit) was implemented using two separate channels.

1) **Broker Setup:** Eclipse Mosquitto broker was installed on the Raspberry Pi and configured to run on the default port (1883).

2) **Client Roles:**
   a. The PC acts as the publisher, sending control messages after processing video frames through the YOLOv8n detection model.
   b. The Raspberry Pi acts as the subscriber, listening to the topic vehicle/control.

3) **Message Format:**
   a. "start" → sets cruise speed.
   b. "stop" → stops the motor.
   c. "label|confidence|ROI" → anomaly detection message, e.g., "pothole|0.35|50%".

4) **Qos:** Messages are transmitted with QoS 0 to minimize latency.

3.3 Mathematical Formulation of Control Logic

1) Detection Confidence and Buffering

To mitigate transient false positives and stabilize predictions, a decision buffer logic is implemented.

Let:

- c_t = confidence of detection at frame t
- l_t = detected label at frame t, where $l_t \in \{\text{pothole}, \text{speedbump}, \text{none}\}$
- c_{th} = confidence threshold

A decision buffer B_t of length $N = 15$(sliding window of last N frames) is defined as:

$$B_t = \{ b_{t-N+1}, \dots, b_t \}$$

Where each buffer entry is

$$b_k = \begin{cases} l_k, & If\ detection\ at\ frame\ k\ is\ valid \\ none, & otherwise \end{cases}$$

A final decision is triggered only if the buffer contains a minimum number of positive detections:

$$\text{Count}(B_t, \text{pothole}) \geq 5 \quad \text{or} \quad \text{Count}(B_t, \text{speedbump}) \geq 5$$

Where Count(B_t,x) is the number of occurrences of label x inside the ordered buffer B_t.

2) ROI-Based Spatial Filtering

To ensure that only road anomalies located within the vehicle's driving path influence control decisions, a Region of Interest (ROI) filtering mechanism is applied to each detection.

Let:

- R = set of pixels belonging to the trapezoidal ROI in image space
- D_t = bounding box region of the detected object at frame t
- $|A|$ = area of region A in pixels

The overlap ratio between the detected object and the ROI is defined as:

$$O_t = \begin{cases} \dfrac{|R \cap D_t|}{|D_t|}, & if\ |D_t| > 0, \\ 0, & if\ |D_t| = 0 \end{cases}$$

where $R \cap D_t$ represents the intersection area of the detection box and the ROI.
A detection is considered valid (in-path) if:

$$valid_t: (c_t \geq c_{th}) \wedge (O_t \geq \theta)$$

where θ is the overlap threshold (empirically set between 0.3 and 0.5, depending on camera placement and calibration). Only detections ($valid_t$) that both exceed the confidence threshold and lie within the ROI are added to the decision buffer.

3) Cooldown Logic

To prevent excessively frequent actions, a cooldown period $T_c = 2$ seconds is enforced. A new motor action is permitted only if:

$$t - t_{\text{last\_action}} \geq T_c$$

where $t_{\text{last\_action}}$ is the timestamp of the last applied action.

4) Gradual Speed Adaptation

To ensure smooth transitions and prevent abrupt changes in motor speed, a gradual speed adaptation algorithm is implemented. Let:

- $v_t = \text{current speed (duty cycle \%) at time } t$
- $v_{\text{target}} = \text{target speed set by detection logic}$
- $\Delta v = \text{step change per iteration (1\%)}$
- $\Delta t = \text{control update interval (0.1s=100ms)}$

The speed is updated incrementally according to:

$$v_{t+\Delta v} = \begin{cases} v_t + \Delta v, & if\ v_t < v_{\text{target}}, \\ v_t - \Delta v, & if\ v_t > v_{\text{target}}, \\ v_t, & otherwise \end{cases}$$

This ensures smooth interpolation between speed values rather than instantaneous jumps.

5) Target Speed Function

With ROI-based filtering incorporated, the target speed v_{target} is updated conditionally as:

$$v_{target} = \begin{cases} v_{cruise}, & if\ no\ detection\ in\ ROI\ for\ T_f\ seconds \\ v_{slow}, & if\ valid\ anomaly\ detected\ within\ ROI\ and\ cooldown\ expired \end{cases}$$

Where

v_{cruise}=75%, v_{slow}=25%, T_f=5s

6) Fail-safe Recovery Mechanism

To maintain safe operation during periods of uncertain detection, a fail-safe mechanism is implemented. If no valid detection is observed for a duration T_f=5 seconds, the target speed is reset to the default cruising speed:

$$v_{\text{target}} = v_{\text{cruise}}$$

Where v_{cruise} represents the default cruising speed, set at 75% of the maximum duty cycle.

7) Integrated Controller Function

The complete system behavior can be expressed as a function mapping the detection inputs, ROI spatial constraints, and system parameters to motor speed outputs:

$$\boldsymbol{v_{t+1} = f(l_t, c_t, O_t, B_t, T_c, T_f, v_{\text{cruise}}, v_{slow})}$$

Where:

- l_t: detected label at time
- c_t: detection confidence at time
- O_t: ROI overlap ratio of detection at time
- B_t: decision buffer of length N = 15
- T_c: cooldown period (2 seconds)
- T_f: failsafe timeout period (5 seconds)
- v_{cruise}: normal cruising speed (75%)
- v_{slow} : reduced speed during valid detection (25%)

The function *f* integrates spatial and temporal filtering as follows:

$$v_{t+1}\begin{cases} v_{\text{slow}}, & \text{if Count}(B_t.x) \geq 5 \ \wedge \ \left(t - t_{\text{last\_action}} \geq T_c\right) \\ v_{\text{cruise}}, & \text{if no valid entry in } B_t \, for \, T_f \\ v_t, & \text{otherwise} \end{cases}$$

The function ensures that only confident, spatially relevant detections influence the motor controller. The resulting $\boldsymbol{v_{t+1}}$ is then smoothed through the gradual speed adaptation mechanism defined earlier, ensuring continuous and stable control transitions.

3.4 Hardware Setup

Figure 4 shows the layout designed in Tinkercad before the actual hardware setup.

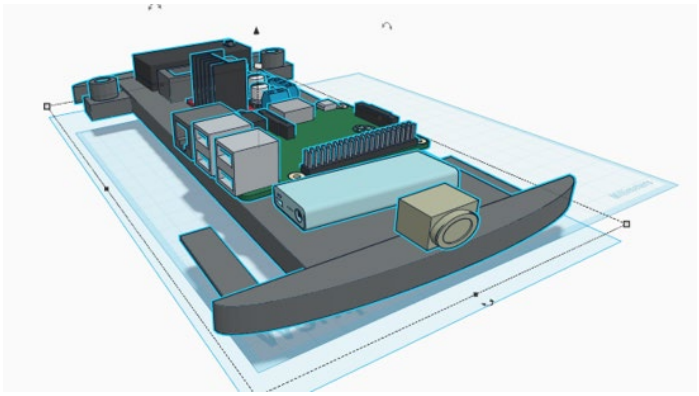

Figure 4. Physical Layout in Tinkercad

The wiring diagram was designed on Fritzing and was used to validate the wiring on actual hardware. The main components of the system are a Raspberry Pi 4B, a L298N motor driver, a webcam, and four DC motors. The wiring Layout of the system is shown in Figure 5.

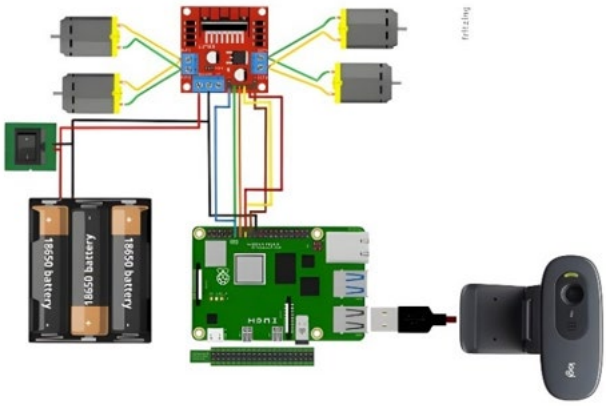

Figure 5. Wiring Layout of the System

4. Data Collection and Processing

This study did not involve primary data collection. The model was trained using publicly available datasets containing annotated images of potholes and speed bumps. The datasets were obtained from open repositories such as Kaggle, Mendeley Data, and Dataset Ninja. These datasets provide labeled road surface conditions suitable for training and evaluating the proposed object detection model.

4.1 Dataset Preprocessing and Augmentation

Preprocessing applied included:

- Auto-orientation
- Resize with aspect-fit padding to 640×640
- Adaptive contrast enhancement using histogram equalization
- Normalization of images and labels

Augmentations applied included:

- Rotation ±5°
- Horizontal and vertical flip
- Brightness and contrast jitter
- Mosaic: 0.5
- Mixup: 0.1
- Blur and noise
- Scale: 0.2
- HSV Augmentation

4.2 Model Training and Implementation

Model training was conducted using Kaggle Notebooks, which provide access to high-performance NVIDIA GPUs. For improved detection accuracy, the YOLOv8m (medium) variant was selected instead of the lighter YOLOv8n or YOLOv8s models. The model was initialized with pretrained COCO weights (yolov8m.pt) and fine-tuned on the combined pothole and speed-bump dataset.

Training Configuration:

- Model: YOLOv8m (medium variant)
- Platform: Kaggle Notebooks
- GPU: NVIDIA Tesla P100 Accelerator
- Epochs: 120
- Batch Size: 16
- Image Size: 640 × 640
- Initial Learning Rate (lr0): 0.001
- Final Learning Rate (lrf): 0.01
- Patience (Early Stopping): 30
- Workers: 6
- Device: GPU (device = 0)

5. Results and Discussion

This section presents the result of the YOLOv8m model evaluation for real-time road anomaly detection and the implementation result on Raspberry Pi 4B. The model performance is evaluated in terms of training and validation losses, precision, recall, mAP, and F1-score over the epochs. The best-performing model is identified based on the highest mAP and F1-score. Visualizations and tables are provided to illustrate the model's learning behavior and detection accuracy.

5.1 Confusion Matrix Analysis

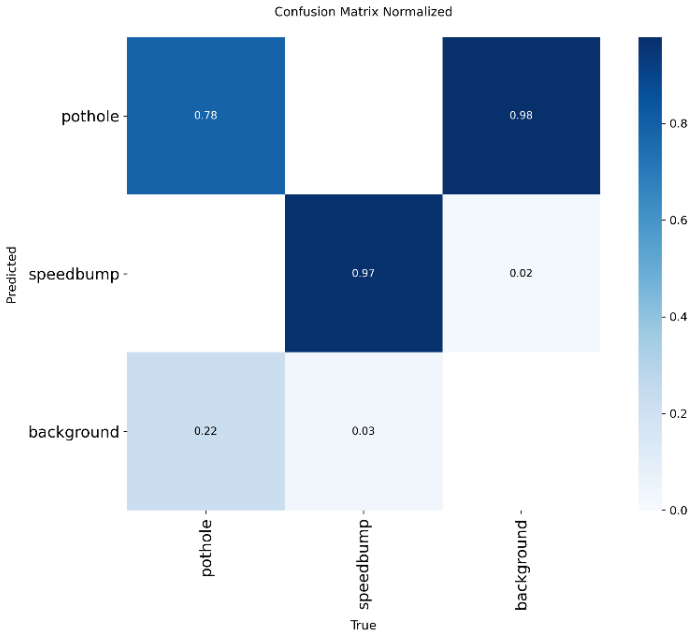

Figure 6. Normalized Confusion Matrix

As seen from the confusion matrix shown in Figure 6, the model shows a strong overall performance, but several areas still need refinement. The speedbump class is identified with high reliability, achieving a true positive rate of 0.97, indicating that the model correctly recognizes most speed bumps with minimal confusion. The background class also performs very well, reaching a perfect or near-perfect true positive rate of 0.98, meaning clean road surfaces are rarely misclassified. However, the pothole class exhibits noticeable weakness. With a true positive rate of 0.78, the model misclassifies a considerable portion of potholes, frequently confusing them with the background (0.22 misclassification rate).

5.2 Precision Recall Curve Analysis

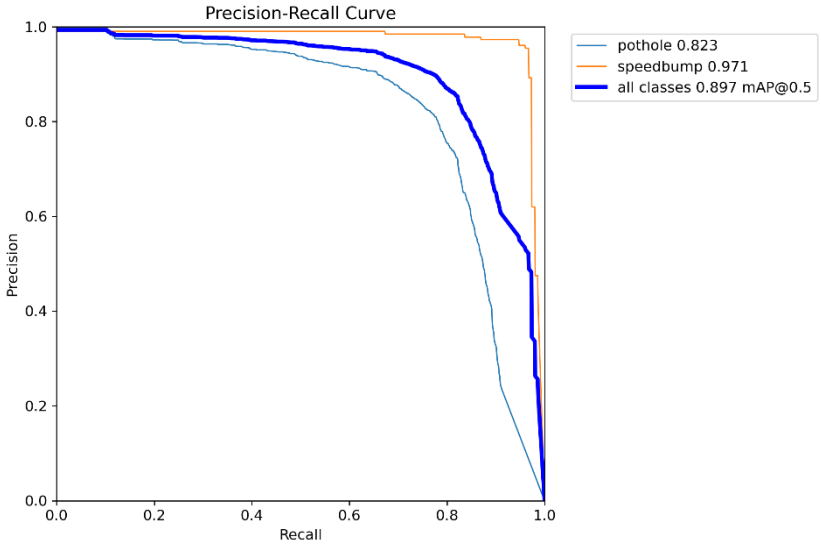

Figure 7. Precision-Recall Curve

The Precision-Recall (PR) curve and generated Average Precision (AP) metrics shown in Figure 7 provide a thorough evaluation of the model's performance across all decision thresholds, which is especially important given the class imbalance found in the confusion matrices. For the ‘speedbump’ class, the model's high AP of 0.971 shows that it can recall the majority of pertinent speedbump events with good overall precision. By comparison, the 'pothole' class's AP is significantly lower at 0.823. Given that a low AP is the consequence of a sharp decline in precision when the model tries to improve its recall for this class, this much lower score reflects the machine's great difficulties in accurately spotting potholes. Although the model's detection accuracy is summarized by the mean Average Precision (mAP@0.5) of 0.897, the key deficit in 'pothole' detection is concealed by the high performance on the ‘speedbump’ and 'background' classes.

5.3 Precision Recall Curve Analysis

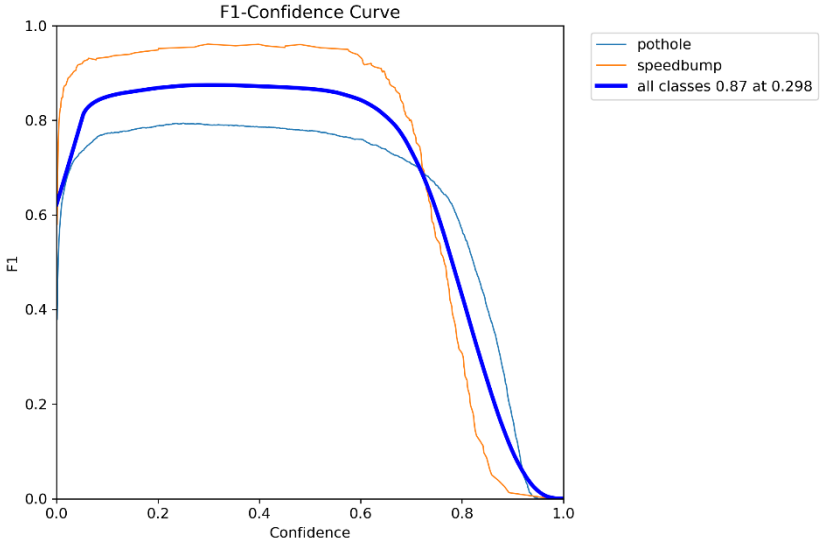

Figure 8. F1-confidence Curve

The F1-Confidence Curve in Figure 8 shows how the F1 score changes with varying confidence thresholds for the classes 'pothole' and 'speedbump'. The speedbump class achieves higher F1 scores across most thresholds, indicating better model performance for this category. The overall model performance, represented by the bold blue curve, peaks at an F1 score of 0.87 at a confidence threshold of 0.298. This threshold provides the best trade-off between precision and recall for all classes combined.

5.4 Raspberry Pi Implementation

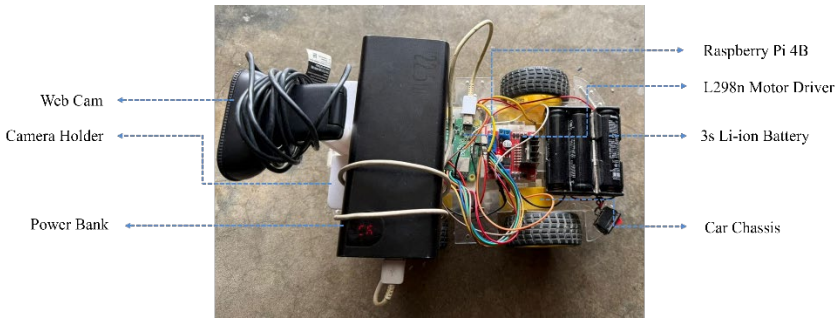

Figure 9. Developed System Hardware

Figure 9 shows the developed system hardware in action.

Table 2. Smart Vehicle GUI Features

Features	Description
Live Stream Viewer	Displays real-time video feed from the Raspberry Pi's Camera
Start Button	Starts the vehicle by sending MQTT "Start" command
Stop Button	Stops the vehicle immediately by sending MQTT "Stop" command
Detection Alert Indicator	Displays on-screen alert when pothole/speedbump is detected
Detection Log Viewer	Real-time display of detections with timestamp

Table 2 lists the smart vehicle GUI features, and Figure 10 shows the smart vehicle dashboard.

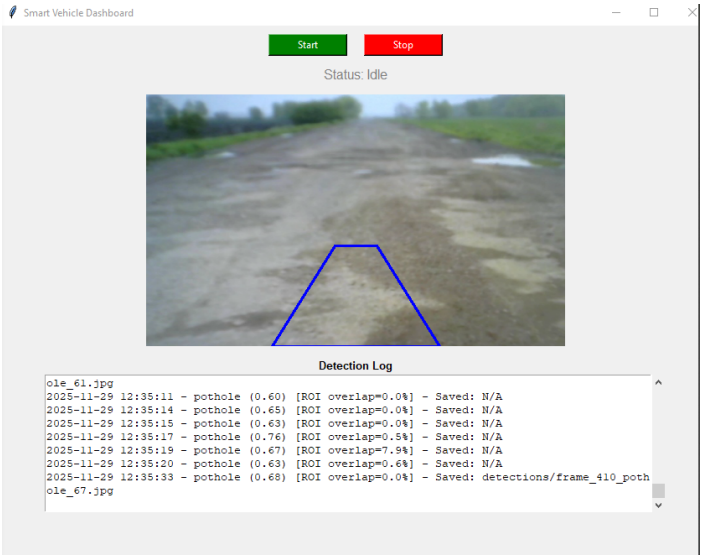

Figure 10. Smart Vehicle Dashboard

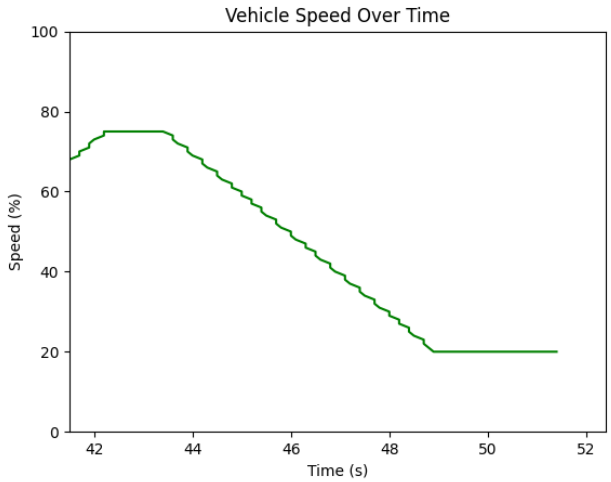

Figure 11. Real-time Vehicle Speed Over Time Graph

Figure 11 shows the vehicle speed change over time graph in real-time to observe the speed adaptation, and every detection frame is auto-saved with Label, confidence, and ROI in the Detections folder, which is shown in Figure 12.

.

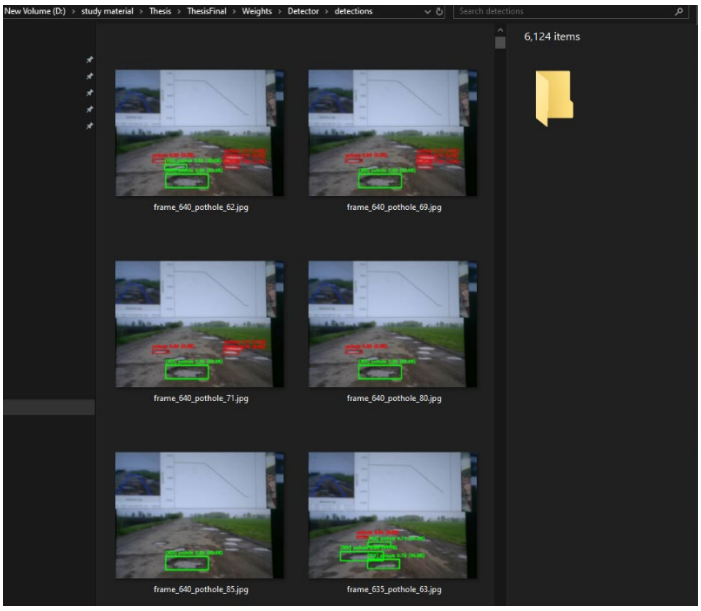

Figure 12. Detection Frame Auto-saved with Label in Detections Folder

5.5 Proposed Improvements

The developed system was trained by a publicly available dataset, which is not nearly enough for accurate detection. To increase detection accuracy and reduce false positives for on-vehicle deployment, the system requires a diversified road-anomaly dataset that includes a variety of appearances, lighting, and, in particular, unmarked or distorted speed bumps commonly found in Bangladesh. Currently, detection is run on a PC, while real-time video streaming is handled on a Raspberry Pi over a local network; to scale this for remote or distributed monitoring, both devices should join as clients to a cloud-hosted MQTT broker (e.g., AWS/HiveMQ/Mosquitto). Implementing reconnect logic, QoS levels, and secure authentication (username/password or TLS) will result in dependable, secure cross-network functioning.

6. Conclusion

Intelligent transportation systems are growing at a rapid rate, which emphasizes the rising need for safer and more flexible driving technology. To show real-time road anomaly detection and speed regulation, a hardware prototype combining an adaptive motor controller on a Raspberry Pi with custom-trained YOLOv8m detection was created in this work. In order to solve a significant problem for autonomous and semi-autonomous cars operating on uneven road surfaces, the system combines road anomaly detection with smooth speed adaptation. Although the technology was tested on a small-scale prototype car, the approach is representative of larger uses. In underdeveloped nations, where bad road conditions continue to be a major obstacle to the adoption of autonomous vehicles, the combination of lightweight neural network models with embedded control units shows potential for an affordable deployment. While the GUI allows for human monitoring and override capabilities, the controller has been reinforced against false detections by the use of a buffered detection technique, cooldown timers, and failsafe recovery. This paper shows a workable framework for expanding detection to other road hazards and implementing more advanced control strategies like PID or fuzzy logic, despite its limitations of focusing only on two anomaly types (potholes and speed bumps) and depending on Wi-Fi-based communication. With additional research, particularly in dataset expansion and model optimization for South Asian road conditions, this line of work has the potential to uncover new opportunities in the autonomous vehicle market and pave the path for smarter, safer transport systems.

References

Ahmed, K. R., *Smart pothole detection using deep learning based on dilated convolution*, *Sensors*, vol. 21, no. 24, p. 8406, 2021

Asad, M. H., Khaliq, S., Yousaf, M. H., Ullah, M. O. and Ahmad, A., *Pothole detection using deep learning: A real-time and AI-on-the-edge perspective*, *Advances in Civil Engineering*, vol. 2022, no. 1, p. 9221211, 2022.

Babu, C. N. K., Priya, W. D., Srihari, T. and Nandakumar, R., *Speed-bump detection using Otsu's algorithm and morphological operation*, *International Journal on Emerging Technologies*, vol. 11, no. 3, pp. 989–994, 2020.

Bathla, G., Bhadane, K., Singh, R. K., Kumar, R., Aluvalu, R., Krishnamurthi, R., Kumar, A., Thakur, R. N. and Basheer, S., *Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities*, *Mobile Information Systems*, vol. 2022, no. 1, p. 7632892, 2022.

Choi, J., Lee, J., Kim, D., Soprani, G., Cerri, P., Broggi, A. and Yi, K., *Environment-detection-and-mapping algorithm for autonomous driving in rural or off-road environment*, *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 974–982, 2012.

Daily Sun, *Bumpy ride, no legal guide*, *Daily Sun*, 17 June 2025.

Devapriya, W., Babu, C. N. K. and Srihari, T., *Advance driver assistance system (ADAS)-speed bump detection*, *Proc. 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pp. 1–6, 2015.

Devapriya, W., Babu, C. N. K. and Srihari, T., *Real time speed bump detection using Gaussian filtering and connected component approach*, *Proc. 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*, pp. 1–5, 2016.

Heo, D.-H., Choi, J.-Y., Kim, S.-B., Tak, T.-O. and Zhang, S.-P., *Image-based pothole detection using multi-scale feature network and risk assessment*, *Electronics*, vol. 12, no. 4, p. 826, 2023.

Hoang, N.-D., *An artificial intelligence method for asphalt pavement pothole detection using least squares support vector machine and neural network with steerable filter-based feature extraction*, *Advances in Civil Engineering*, vol. 2018, no. 1, p. 7419058, 2018.

Islam, M. A. and Dinar, Y., *Evaluation and spatial analysis of road accidents in Bangladesh: an emerging and alarming issue*, *Transportation in Developing Economies*, vol. 7, no. 1, p. 10, 2021.

Park, S.-S., Tran, V.-T. and Lee, D.-E., *Application of various YOLO models for computer vision-based real-time pothole detection*, *Applied Sciences*, vol. 11, no. 23, p. 11229, 2021.

Peralta-López, J.-E., Morales-Viscaya, J.-A., Lázaro-Mata, D., Villaseñor-Aguilar, M.-J., Prado-Olivarez, J., Pérez-Pinal, F.-J., Padilla-Medina, J.-A., Martínez-Nolasco, J.-J. and Barranco-Gutiérrez, A.-I., *Speed bump and pothole detection using deep neural network with images captured through ZED camera*, *Applied Sciences*, vol. 13, no. 14, p. 8349, 2023.

Shah, S. and Deshmukh, C., *Pothole and bump detection using convolution neural networks*, *Proc. 2019 IEEE Transportation Electrification Conference (ITEC-India)*, pp. 1–4, 2019.

Tai, Y.-c., Chan, C.-w. and Hsu, J. Y., *Automatic road anomaly detection using smart mobile device*, *Proc. Conference on Technologies and Applications of Artificial Intelligence*, Hsinchu, Taiwan, 2010.

Biographies

Abdullah Al Rakib recently received his bachelor's degree in mechatronics engineering from Khulna University of Engineering & Technology (KUET). His academic career has been guided by a strong interest in intelligent systems, robotics, and computer vision, particularly their applications in real-world safety and automation. Throughout his undergraduate studies, he emphasized interdisciplinary problem solving by combining mechanical design, embedded systems, machine learning, and control systems.

Dr. Asief Javed is working as an Assistant Professor in the Department of Mechatronics Engineering. He joined Khulna University of Engineering & Technology (KUET) in 2020. He completed his PhD on vibration reduction in magnetic suspension system on 2018 from Saitama University, Japan. He completed his Masters from the same university on 2015. He was awarded MEXT scholarship to pursue both of the postgraduation programs. During his masters and PhD program, he worked at Control Engineering Lab (Seigyo Kenkyushitsu). He finished his B.Sc. in Mechanical Engineering from Rajshahi University of Engineering & Technology (RUET) in 2012. His research interest includes implementation of control in different systems, vibration suppression, robotics, etc.