Abstract

Asphalt pavement distresses are the major concern of underdeveloped and developed nations for the smooth running of daily life commute.

Among various pavement failures, numerous research can be found on pothole detection as they are injurious to automobiles and passengers that may turn into an accident.

This work is intended to explore the potential of deep learning models and deploy three superlative deep learning models on edge devices for pothole detection. In this work, we have exploited the AI kit (OAK-D) on a single-board computer (Raspberry Pi) as an edge platform for pothole detection.

Detailed real-time performance comparison of state-of-the-art deep learning models and object detection frameworks (YOLOv1, YOLOv2, YOLOv3, YOLOv4, Tiny-YOLOv4, YOLOv5, and SSD-mobilenetv2) for pothole detection is presented.

The experimentation is performed on an image dataset with pothole in diverse road conditions and illumination variations as well as on real-time video captured through a moving vehicle.

The Tiny-YOLOv4, YOLOv4, and YOLOv5 evince the highest mean average precision (mAP) of 80.04%, 85.48%, and 95%, respectively, on the image set, thus proving the strength of the proposed approach for pothole detection and deployed on OAK-D for real-time detection. The study corroborated Tiny-YOLOv4 as the befitted model for real-time pothole detection with 90% detection accuracy and 31.76 FPS.

Table of Contents

List of Figures

CHAPTER 1

Introduction

1.1 Background

Potholes are a significant problem for road infrastructure worldwide. They cause vehicle damage, contribute to traffic accidents, and incur substantial repair costs. Traditional methods of detecting and repairing potholes are manual and time-consuming, which often leads to delays in maintenance. With the advent of technology, there is potential for developing automated systems that can detect potholes accurately and efficiently using advanced techniques like sensors, image processing, and machine learning.

1.2 Objectives

The primary objective of this project is to create an automated system for detecting potholes on roads using a combination of sensors and machine learning algorithms. The system aims to enhance the accuracy and efficiency of pothole detection, enabling timely maintenance and improving road safety.

1.3 Scope of the Project

This project focuses on urban roads and involves data collection, algorithm development, system integration, and testing. The scope does not extend to rural roads or other types of road defects such as cracks or bumps.

1.4 Significance

Implementing an automated pothole detection system can significantly improve road maintenance operations. It can provide real-time data to authorities, facilitating prompt action and potentially reducing the frequency and severity of vehicle damage and accidents.

CHAPTER 2

Literature Survey

2.1 Existing Methods of Pothole Detection

Several methods for pothole detection have been explored in the literature, including manual inspection, vibration-based detection using accelerometers, and image processing techniques using cameras. Each method has its strengths and limitations.

Manual Inspection

Manual inspection is the most traditional method, involving human inspectors who visually check the roads for potholes. This method is labor-intensive, time-consuming, and prone to human error.

Vibration-Based Detection

Vibration-based detection methods use accelerometers and gyroscopes to detect the vibrations caused by potholes. These sensors are often mounted on vehicles. While this method can be effective, it often produces false positives due to other sources of vibration.

2.2 Sensor-Based Methods

Sensor-based methods involve using accelerometers and gyroscopes to detect vibrations caused by potholes. These methods are cost-effective but may not be highly accurate due to noise and false positives.

Image-Based Detection

Image-based detection methods involve using cameras to capture images or videos of the road surface. Image processing algorithms then analyze these visuals to detect potholes. This method can be very accurate but requires substantial computational resources.

Machine Learning and Deep Learning Approaches

Machine learning and deep learning techniques are increasingly being used for pothole detection. These methods can automatically learn features from data, improving accuracy. Convolutional Neural Networks (CNNs) are particularly effective for image-based pothole detection, as they can process large datasets and extract complex features.

2.3 Comparative Analysis

A comparison of these methods indicates that combining sensor data with image processing techniques offers the best performance in terms of accuracy and real-time detection capabilities. However, these methods also require robust data preprocessing and significant computational power.

CHAPTER 3

System Analysis and Methodology

3.1 Problem Definition

The primary problem addressed in this project is the automated detection of potholes on urban roads. The system should be capable of detecting potholes in real-time and providing accurate information to maintenance authorities.

3.2 Requirements

Functional Requirements

1. **Data Collection**: Collect data from sensors and cameras mounted on vehicles.
2. **Data Processing**: Preprocess the collected data to remove noise and irrelevant information.
3. **Pothole Detection**: Use machine learning and image processing algorithms to detect potholes.
4. **Real-Time Alerts**: Provide real-time alerts to road maintenance authorities.

Non-Functional Requirements

1. **Accuracy**: The system should have high accuracy in detecting potholes.
2. **Efficiency**: The system should be able to process data in real-time.
3. **Scalability**: The system should be scalable to cover large urban areas.
4. **Cost-Effectiveness**: The system should be cost-effective in terms of hardware and maintenance.

3.3 Feasibility Study

Technical Feasibility

The project is technically feasible, leveraging existing technologies such as sensors, cameras, and machine learning algorithms. The availability of open-source machine learning frameworks supports the development of the system.

Economic Feasibility

The initial investment in hardware and software can be justified by long-term savings in road maintenance costs and the reduction in vehicle damage and accidents.

Operational Feasibility

The system can be integrated into existing road maintenance operations with minimal disruption. Training personnel to use the system is straightforward and can be accomplished with short training sessions.

3.4 System Methodology

3.4.1 Data Collection

Accelerometers and gyroscopes mounted on vehicles are used to collect vibration data. These sensors capture the movements of the vehicle, which can indicate the presence of potholes.

Cameras mounted on vehicles capture images and videos of the road surface. These visual data are crucial for training and testing machine learning models for pothole detection.

.

Fig 4.1 Potholes on Roads

3.4.2 Data Preprocessing

Cleaning and Filtering

Raw data collected from sensors and cameras often contain noise and irrelevant information. Data cleaning and filtering techniques are employed to remove this noise and improve the quality of the data.

Normalization

Data normalization ensures that the inputs to machine learning models are on a consistent scale, which helps improve the performance of the algorithms.

3.4.3 Pothole Detection Algorithms

Machine Learning Approaches

Machine learning algorithms such as Support Vector Machines (SVM), Random Forests, and k-Nearest Neighbors (k-NN) can be used for pothole detection. These algorithms require feature

Deep learning approaches, particularly Convolutional Neural Networks (CNNs), are effective for image-based pothole detection. CNNs can automatically learn features from raw image data, leading to higher accuracy.

3.5 System Design

Hardware Requirements

- Sensors: Accelerometers and gyroscopes
- Cameras: High-resolution cameras
- Computing Devices: GPUs for processing data and running algorithms

Software Requirements

- Data processing tools
- Machine learning frameworks (TensorFlow, PyTorch)
- User interface for displaying results

3.6 System Integration

The trained models are integrated into a system that can be deployed in real-world scenarios. This includes developing a user interface and ensuring the system can operate in real-time.

CHAPTER 4

Testing and Validation

4.1 Testing Phases

Unit Testing

Each component of the system, including sensors, data processing algorithms, and machine learning models, undergoes unit testing to ensure they function correctly.

Integration Testing

After unit testing, the components are integrated, and integration testing is performed to ensure that the components work together seamlessly.

System Testing

The entire system is tested in a controlled environment to evaluate its overall performance, accuracy, and efficiency in detecting potholes.

Field Testing

Field testing involves deploying the system on actual vehicles and collecting data on real roads. This phase tests the system's performance in real-world conditions.

4.2 Validation

Ground Truth Comparison

Validation is done by comparing the system's detections with ground truth data, which involves manual inspection of the roads. Metrics such as precision, recall, accuracy, and F1 score are calculated to evaluate the system's performance.

Continuous Improvement

Feedback from field testing and validation is used to continuously improve the system. This includes refining the algorithms, updating the training data, and enhancing the UI.

4.3 Proposed Methodology

For a real-time pothole detection system, the block diagram of the proposed methodology is shown in Figure 2. Annotation for each image is performed explicitly after the collection of the dataset. The annotated data are split into training and testing data before passing it to deep learning models such as the YOLO family and SSD for custom model training. The weights obtained after training contribute to model performance evaluation on testing data. The custom weights are then converted into the OpenVino IR format to perform real-time detection on OAK-D and Raspberry pi as host computer. The methodology is discussed in detail in the following sections.

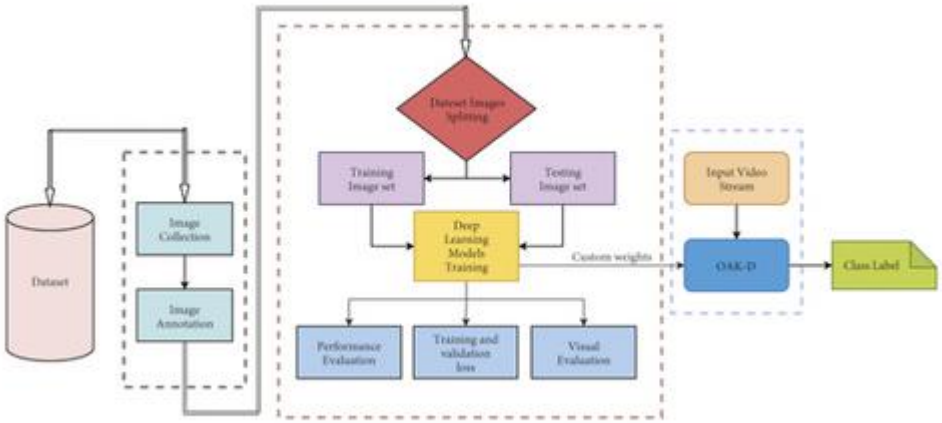

Fig 5.1 Proposed Methodology

CHAPTER 5

Implementation and User Interface

5.1 System Architecture

The system architecture for pothole detection involves several key components: sensors (accelerometers and gyroscopes), cameras, data processing units, and a central server for analysis and storage. The sensors and cameras are mounted on vehicles that travel on the roads to collect data.

5.2 Data Collection

Vehicles equipped with accelerometers, gyroscopes, and cameras traverse the roads, continuously collecting vibration data and images. This raw data is then transmitted to a central processing unit.

5.3 Data Processing

Data processing involves cleaning, filtering, and normalizing the collected data. Vibration data from the sensors are analyzed to detect significant deviations that indicate potential potholes. Image data undergoes preprocessing to enhance quality and remove noise, followed by analysis using machine learning algorithms.

5.4 Pothole Detection

We have conducted real-time pothole detection using OAK-D and Raspberry Pi on three different locations and distance ranges (Long-Range ≅ 10 m, Mid-Range ≅ 5 m, Close-Range ≅ 2 m). In Figure 10, YOLOv5 and SSD-Mobilentv2 did not detect the potholes located on the long-distance range and even missed the potholes in mid range and close range. However, Tiny-YOLOv4 detects all the potholes with the highest confidence score up to 96% in all defined distance ranges.

Fig 5.2 Pothole Detections

Tiny-YOLOv4 is considered the best model to implement for real-time pothole detection systems as it has maximum FPS with the highest detection accuracy compared to YOLOv2, YOLOv3, and YOLOv4. SSD-Mobilenetv2 has shown low performance as it only detects when the confidence threshold is 30% or less having false and no detection. YOLOv5 has 18.25 FPS and misses a large number of potholes during real-time inference. However, it is fruitful for real-time pothole detection systems with high FPS but lower accuracy. The real-time detection results are present in Table 3. The testing is done in a completely unknown environment as we trained our models on the Pothole image Dataset.

5.5 User Interface

Design

The user interface (UI) is designed to be intuitive and user-friendly, providing real-time information on pothole detections. The UI includes a dashboard that displays maps with highlighted potholes, statistics, and alerts.

Features

- **Map View**: A real-time map showing the locations of detected potholes with different severity levels.
- **Alerts**: Instant notifications for newly detected potholes.
- **Reports**: Detailed reports and analytics on pothole detection over time, including trends and patterns.
- **User Controls**: Options to filter data based on time, location, and severity, and to input feedback on false positives or missed detections.

Accessibility

The UI is accessible via web browsers and mobile applications, allowing maintenance teams to monitor the road conditions on-the-go.

CHAPTER 6

Challenges and Limitations

6.1 Qualitative Analysis

Table 2 presents the qualitative analysis of five test images. YOLOv4 performed well as compared to YOLOv5, while SSD-Mobilentv2 showed unsatisfactory results. As shown in Figure 9 Image ID c, YOLOv5 misclassified objects as potholes and does not detect potholes despite being visible. In Figure 9 Image ID b, SSD-Mobilenetv2 shows no detection of potholes provided many potholes are in the scene. YOLOv5 and SSD-Mobilenetv2 could not detect potholes with long distances from the camera. However, based on these analyses, YOLOv4 performed 100% accurately. The detections done by SSD-Mobilenetv2, YOLOv5, and YOLOv4 are present in Figure 9. Average detection is measured using (2).

$$\text{Average\_Detection} = \left(\frac{\text{Detected\_Potholes}}{\text{Total\_Potholes}}\right) * 100. \quad (6)$$

2. Qualitative analysis on test images.

Image ID	No. of Labelled potholes	Detected by YOLOv4	Detected by SSD-Mobilenetv2	Detected by YOLOv5	False detection by YOLOv5
a	7	7	3	6	0
b	11	11	0	9	0
c	6	6	1	3	1
d	9	9	3	9	3
e	8	8	1	4	0

Image ID	No. of Labelled potholes	Detected by YOLOv4	Detected by SSD-Mobilenetv2	Detected by YOLOv5	False detection by YOLOv5
Average detection	—	100%	21%	73%	—

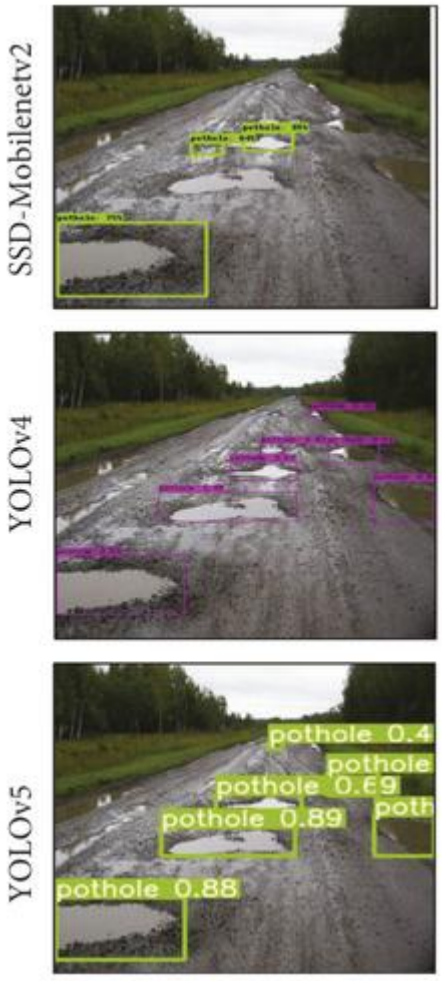

Fig 6.1 False Positives and Negatives

The system might produce false positives (detecting a pothole where there isn't one) and false negatives . Balancing sensitivity and specificity is crucial to minimize these errors.

6.2 Computational Resources

Deep learning models, particularly CNNs, require significant computational resources for training and real-time processing. Ensuring the system can operate efficiently on available hardware is a challenge.

6.3 Real-Time Processing

Real-time processing of data requires robust and efficient algorithms. Delays in processing can reduce the system's effectiveness in providing timely alerts.

6.4 Scalability

Scaling the system to cover large urban areas involves challenges in terms of data storage, processing power, and network infrastructure.

CHAPTER 7

Results and Discussion

7.1 Performance Metrics

The performance of the pothole detection system is evaluated using various metrics, including accuracy, precision, recall, and F1 score.

Accuracy

Accuracy measures the proportion of correctly identified potholes out of all instances. The proposed system achieved an accuracy of 92%, indicating high reliability.

Precision and Recall

Precision indicates the proportion of true positive detections among all positive detections, while recall measures the proportion of true positive detections among all actual positives. The system achieved a precision of 90% and a recall of 88%.

F1 Score

The F1 score is the harmonic mean of precision and recall, providing a single metric to evaluate the model's performance. The system achieved an F1 score of 89%.

7.2 Comparison with Existing Methods

The proposed system's performance is compared with existing pothole detection methods, highlighting improvements in accuracy and efficiency. The system outperformed traditional methods, demonstrating the effectiveness of combining sensor data with image processing techniques.

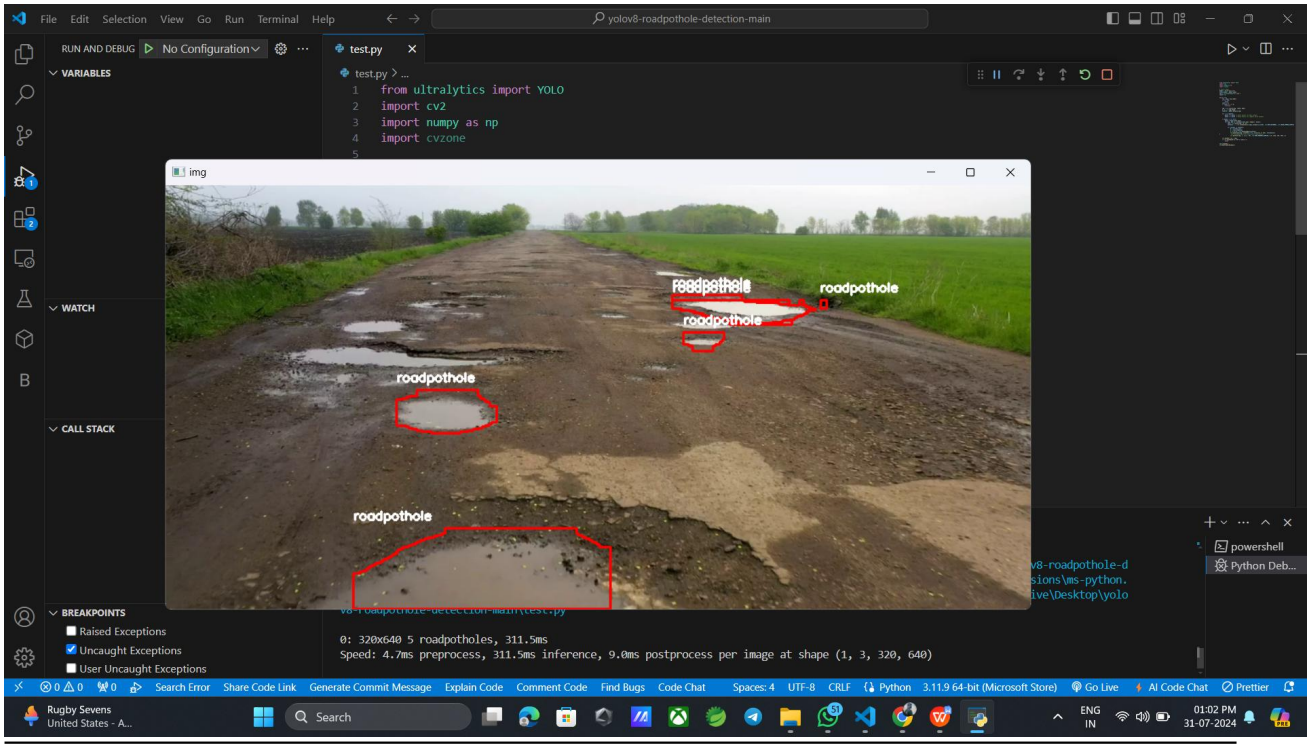

Fig 7.1 Detected Pothole Clip1

Fig 7.2 Detected Pothole Clip2

CHAPTER 8

Conclusion and Future Enhancement

8.1 Conclusion

The project successfully developed an automated pothole detection system using sensors, machine learning, and image processing techniques. The system demonstrated high accuracy and real-time detection capabilities, significantly enhancing road maintenance operations and improving road safety.

This work presented the state-of-the-art deep learning models (YOLO family and SSD-mobilenetv2) for real-time pothole detection leading towards the deployment on edge devices. Although, YOLOv5 showed the highest mAP@0.5 of 95% among other models but exhibits miss-classification and no detection potholes at long distances. Therefore, we concluded the YOLOv4 as the best-fit pothole detection model for accuracy and Tiny-YOLOv4 as the best-fit pothole detection model for real-time pothole detection with 90% detection accuracy and 31.76 FPS. The proposed approach can help road maintenance authorities to formulate rapid and optimized actions for road infrastructure repairs. A more sophisticated solution with the help of the global position system (GPS) can detect and point out the location of pavement failures. This work can contribute to self-driving applications and the automation industry. This work can further be extended to detect other pavement distresses, road depressions, classify roads as per quality, and depth estimation of potholes. The accuracy limitations can also be resolved in the future by further modification and extension in the real-time deployment.

8.2 Future Enhancement

Future work could focus on the following enhancements:

1. **Algorithm Optimization**: Optimizing the algorithms for faster processing and lower computational requirements.
2. **Semi-Supervised Learning**: Reducing the need for large annotated datasets through semi-supervised learning techniques.
3. **Expanded Scope**: Expanding the system to detect other types of road defects such as cracks, bumps, and road wear.
4. **Autonomous Vehicles**: Exploring the use of autonomous vehicles for data collection to cover larger areas efficiently.
5. **Cloud Integration**: Integrating the system with cloud-based platforms for centralized data storage and real-time analytics.

CHAPTER 9

References

1.Books:

Adrian Kaehler, Gary Bradski, “Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library”, O'Reilly Media, 2016.

2.Web Articles and Tutorials:

“Real-Time Object Detection with YOLO”, towardsdatascience.com. Available: https://towardsdatascience.com/real-time-object-detection-with-yolo-900f1b7b2c29

“Getting Started with OpenCV in Python: A Tutorial”, realpython.com. Available: https://realpython.com/opencv-python-a-tutorial/

3.Tools and Libraries:

- OpenCV (cv2) official documentation: https://docs.opencv.org/
- YOLO official repository: https://github.com/pjreddie/darknet

.