

Autonomous Vehicle Object Detection

Rita Patil¹, Shraddha Chavan², Kartik More³

¹ Project Guide, Department of Information Technology

² Department of Information Technology

³ Department of Information Technology

Sandip Institute of Technology and Research Center (SITRC), Nashik, India

Abstract: *Autonomous vehicles require a reliable perception system to understand objects present in their surrounding environment. Object detection is therefore an important component of autonomous driving and Advanced Driver Assistance Systems (ADAS). This paper presents a software-based real-time road-object detection system using Python, OpenCV, and YOLOv8.*

A webcam provides live video input, OpenCV performs frame capture and basic preprocessing, and YOLOv8 detects and classifies objects such as cars, buses, trucks, motorcycles, bicycles, and pedestrians. The system displays bounding boxes, class labels, and confidence scores for the detected objects. The proposed work focuses on the visual-perception stage of autonomous vehicles and is designed as a practical prototype that can run on a conventional laptop or desktop computer.

The methodology covers problem identification, literature survey, system analysis, system design, implementation, testing, and result analysis. Standard metrics including precision, recall, F1-score, mean Average Precision (mAP), and frames per second (FPS) are defined for evaluation. The paper also discusses the limitations of camera-only detection and identifies future extensions such as lane detection, object tracking, distance estimation, traffic sign recognition, collision warning, adverse-weather handling, and edge deployment.

Keywords: Autonomous Vehicles, Object Detection, YOLOv8, Computer Vision, Deep Learning, OpenCV, Artificial Intelligence, Real-Time Detection, ADAS

I. INTRODUCTION

Autonomous vehicles are an important application of Artificial Intelligence (AI), Machine Learning (ML), computer vision, and deep learning. Unlike conventional vehicles, autonomous-driving systems use sensors and intelligent algorithms to perceive the environment, identify relevant objects, and support driving decisions. Among the different perception tasks, object detection is fundamental because it provides information about what objects are present and where they are located in an image or video frame [1,4].

Road environments contain different types of dynamic and static objects, including cars, buses, trucks, motorcycles, bicycles, pedestrians, traffic signs, traffic lights, and other obstacles. Detecting these objects in real time can provide useful information for higher-level functions such as tracking, collision warning, navigation, and path planning. Research on autonomous-vehicle perception has shown that detection systems must consider not only accuracy but also inference speed, computational resources, object size, occlusion, lighting, weather, and sensor limitations [7].

Deep learning has substantially improved object-detection performance. The YOLO (You Only Look Once) family introduced a unified approach that performs object localization and classification using a single network, making it suitable for applications where processing speed is important [3]. Later YOLO-based approaches have been studied for road-object detection and autonomous driving [5,6].

The present project develops a practical software prototype using YOLOv8, Python, and OpenCV. The system receives live video from a webcam and processes each frame to detect objects. The detected objects are displayed using bounding boxes, class labels, and confidence scores. The project follows the scope defined in the project synopsis and focuses on perception rather than complete autonomous vehicle control [1].



The main objectives of the proposed work are:

- To develop a real-time object detection prototype using Python, OpenCV, and YOLOv8;
- To capture and process live video using a webcam;
- To detect and classify common road objects;
- To display bounding boxes, labels, and confidence scores;
- To study the real-time behavior of the detector under different visual conditions; and
- To provide a foundation for future autonomous-driving and ADAS perception modules.

III. BACKGROUND AND THEORETICAL FOUNDATION

3.1 Object Detection

Object detection combines two main tasks: classification and localization. Classification identifies the category of an object, while localization determines its position using a bounding box.

3.2 Deep Learning in Autonomous Driving

Deep learning models are widely used in autonomous-driving perception because they can learn visual features directly from data. CNN-based models are commonly applied to image classification, object detection, and scene perception. Reviews of autonomous-driving research identify perception, sensor fusion, decision-making, and real-time computation as important areas of development [1,4].

3.3 YOLO Object Detection

YOLO was introduced as a unified, real-time object detector that predicts object locations and classes using a single neural network [3]. The single-stage design makes YOLO attractive for real-time applications because the image can be processed in one forward inference process.

Subsequent YOLO versions improved feature extraction, localization, training strategies, and inference efficiency. A study comparing YOLOv3, YOLOv4, and YOLOv5 for autonomous-vehicle road-object detection demonstrated the continuing use of YOLO-based models for rapid object recognition [5]. Improved YOLO-based detectors have also been investigated specifically for autonomous driving and evaluated on driving datasets [6].

3.4 YOLOv8

YOLOv8 is selected for the proposed prototype because it provides a practical interface for Python-based object detection applications. The model can receive an image or video frame and produce detected object classes, bounding-box coordinates, and confidence values. In the proposed system, YOLOv8 is used for inference rather than being presented as a complete autonomous-driving solution.

3.5 Intersection over Union

Intersection over Union (IoU) measures the overlap between a predicted bounding box and its corresponding ground-truth bounding box:

IV. RELATED WORK

4.1 AI and Machine Learning for Autonomous Driving

Ofoegbu reviewed the role of AI and ML in autonomous driving and discussed perception, object detection, sensor fusion, decisionmaking, control, edge computing, and explainable AI [1]. The review also identifies robustness, computational requirements, data scarcity, and challenging environmental conditions as continuing research issues.

4.2 Edge Intelligence for Object Detection

Ahmed et al. presented an AI-powered edge-intelligence framework for object detection in self-driving cars using YOLOv8. Their work investigated local processing as a means of reducing latency and dependence on cloud processing [2]. The proposed student prototype is smaller in scope and uses a webcam and local computer, but the edge-computing concept provides a possible future direction.



4.3 YOLO as a Real-Time Detector

Redmon et al. introduced YOLO as a unified real-time object-detection framework that performs localization and classification in a single network [3]. This work established the foundation for later YOLO architectures and helped make single-stage detection important for real-time computer-vision applications.

4.4 Deep Learning for Scene Perception

Gupta et al. reviewed deep-learning methods for object detection and scene perception in self-driving cars and discussed challenges including data diversity, perception robustness, and the integration of multiple sensing and perception tasks [4]. Their work shows that object detection is one component of a larger autonomous driving perception system.

4.5 Road-Object Detection Using YOLO

Kolukula et al. investigated autonomous-vehicle road-object detection using YOLOv3, YOLOv4, and YOLOv5 [5]. Their published study reported accuracy values ranging from 76.5% to 93.3%, an mAP of 0.895, and an FPS value of 43.95 in their experimental setup. These values are cited only as published literature results and are not claimed as results of the present project.

4.6 Improved YOLO for Autonomous Driving

Jia et al. proposed an improved YOLOv5-based detector for autonomous driving and evaluated it using the KITTI dataset [6]. The work demonstrates how modifications to detector architecture can be used to address the accuracy and speed requirements of autonomous driving.

4.7 Open Challenges in Autonomous-Vehicle Detection

Balasubramaniam and Pasricha reviewed object-detection approaches for autonomous vehicles and discussed two-stage and single-stage detectors, 2D and 3D detection, resource constraints, and deployment challenges [7]. Their analysis indicates that a practical detector must consider the trade-off between detection quality and computational cost.

V. PROBLEM STATEMENT

Road environments contain multiple moving and stationary objects. Manual observation is affected by distraction, fatigue, delayed reaction, changing visibility, and rapidly changing traffic conditions. A computer-vision system can continuously process camera frames and provide information about objects present in the scene.

The problem addressed in this research is the development of a software-based real-time object detection system that can identify common road objects from live video. The system should be simple enough to run on a conventional computer while providing clear visualization of the detected objects. The proposed system aims to:

- Capture live video using a webcam;
- Process video frames using OpenCV;
- Detect and classify road objects using YOLOv8;
- Determine object locations using bounding boxes;
- Display class labels and confidence scores;
- Evaluate detection behavior under different conditions; and
- Provide a foundation for future intelligent transportation applications.

VI. RESEARCH GAP AND CONTRIBUTION

Existing research has demonstrated high-performance object detectors for autonomous vehicles, but many systems depend on specialized hardware, large driving datasets, multiple sensors, or complex architectures. Research also identifies difficulties related to small objects, occlusion, adverse weather, computational cost, and real-time processing. The proposed work addresses this gap at an academic prototype level by combining a webcam, OpenCV, Python, and YOLOv8 into a simple real-time detection pipeline. The contribution of this work is the design of an accessible visual-perception prototype that can be implemented and tested on a conventional laptop or desktop computer.



The system is intentionally limited to object detection and does not claim to provide complete autonomous driving. This distinction is important because autonomous driving additionally requires localization, mapping, prediction, planning, control, and safety validation.

VII. BLOCK DIAGRAM

The block diagram of the proposed Autonomous Vehicle Object Detection System is shown in Fig. 1. It illustrates the complete flow from camera input to the final display of detected objects.

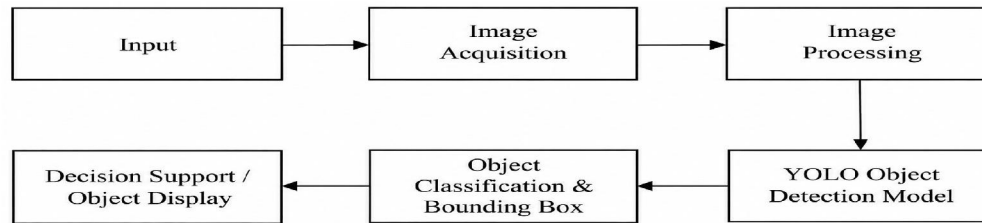


Figure 1 Block Diagram of the Proposed Autonomous Vehicle Object Detection System

7.1 System Flow

The proposed Autonomous Vehicle Object Detection System consists of several sequential stages that work together to detect and classify objects from live video. The complete process starts with camera input and ends with the display of detected objects using bounding boxes, labels, and confidence scores. The major components of the system are explained below.

7.2 Input

The camera acts as the input device of the system. It continuously captures live video from the surrounding environment, including vehicles, pedestrians, traffic signs, and other road objects. The captured video serves as the input for the object detection system.

7.3 Video Frame Capture

The live video is divided into a sequence of individual frames. Each frame is processed separately, enabling the system to analyze the video continuously and perform real-time object detection.

7.4 OpenCV (Image Processing)

OpenCV is an open-source computer vision library used to process the captured video frames. It acquires frames from the camera, performs image preprocessing, and prepares the input for the object detection model. OpenCV also helps in displaying the final output with detected objects.

7.5 YOLO Object Detection Model

YOLO (You Only Look Once) is a deep learning-based object detection algorithm. It analyzes each video frame and detects multiple road objects such as cars, buses, trucks, motorcycles, bicycles, pedestrians, and traffic signs simultaneously. The model identifies both the object class and its location within the image.

7.6 Object Detection and Classification

After processing the input frame, the system classifies each detected object and determines its exact position. It assigns a class label, such as Car, Person, Bus, or Motorcycle, along with a confidence score that indicates the accuracy of the detection.

7.7 Bounding Boxes and Labels

The detected objects are highlighted by drawing rectangular bounding boxes around them. Each bounding box is labeled with the corresponding object name and confidence score, allowing users to easily identify the detected objects in the video.



7.8 Display Output

The final processed video is displayed on the screen with all detected objects, bounding boxes, class labels, and confidence scores. This output provides a real-time visualization of the surrounding environment and demonstrates the working of the AI-based object detection system.

VIII.OBJECT DETECTION ALGORITHM

The real-time detection process is summarized in Table 1.

Table 1 Real-Time Road Object Detection Algorithm

Step	Algorithm Procedure
1	Initialize YOLOv8 model.
2	Initialize webcam.
3	Capture the current video frame.
4	Preprocess the frame using OpenCV.
5	Run YOLOv8 inference on the processed frame.
6	Obtain object classes, bounding boxes, and confidence scores.
7	Filter detections using the selected confidence threshold.
8	Draw bounding boxes and object labels on the detected objects.
9	Display the processed video frame.
10	Repeat Steps 3–9 while the webcam is active.
11	Release the webcam and close the display window.

VIII. METHODOLOGY

The methodology follows the seven phases defined in the project synopsis.

8.1 Phase 1: Problem Identification

The first phase identifies the requirement for accurate and efficient real-time road-object detection. The system focuses on the visual perception problem rather than complete autonomous vehicle control.

8.2 Phase 2: Literature Survey

Research papers related to autonomous vehicles, deep learning, computer vision, object detection, YOLO, and real-time perception were studied. Particular attention was given to YOLO-based detection, autonomous-driving perception, and deployment constraints.

8.3 Phase 3: System Analysis

The required input, processing environment, detection model, and output are identified. A webcam is selected as the input source, OpenCV as the computer-vision layer, and YOLOv8 as the primary detection model.

8.4 Phase 4: System Design

The system workflow is designed so that the webcam, OpenCV, YOLOv8, and output visualization work as separate functional stages. This modular design also makes future integration of tracking, lane detection, and distance estimation easier.

8.5 Phase 5: System Deployment

The prototype is implemented using Python, OpenCV, the Ultralytics YOLO framework, and NumPy. The webcam captures live frames, OpenCV handles the frames, and YOLOv8 performs object detection. The final results are displayed to the user.



8.6 Phase 6: Testing and Validation

Testing includes common road objects such as cars, buses, trucks, motorcycles, bicycles, and pedestrians. The system should also be observed under different lighting conditions, object sizes, distances, partial occlusion, and changing backgrounds.

8.7 Phase 7: Result Analysis

The output is analyzed using detection and real-time performance measures. Incorrect detections, missed detections, confidence scores, processing time, and FPS are recorded. Numerical values are reported only from actual experiments performed using the implemented system.

IX. IMPLEMENTATION REQUIREMENTS

Table 2 Software Requirements

Technology	Purpose
Python	Main programming language.
OpenCV	Webcam capture, frame processing, and visualization.
YOLOv8	Real-time object detection and classification.
Ultralytics	YOLOv8 model framework and utilities.
NumPy	Numerical and image-data operations.
Visual Studio Code	Development and debugging environment.
Windows/Linux	Operating-system platform.

Table 3 Hardware Requirements

Hardware	Purpose
Laptop/Desktop	Development and execution platform.
Webcam/Camera	Captures live video input.
Intel Core i5 or equivalent	Recommended processor.
Minimum 8 GB RAM	Supports Python, OpenCV, and YOLOv8.
Minimum 10 GB free storage	Model, libraries, project files, and outputs.
GPU (optional)	Can accelerate model inference.

X. EXPERIMENTAL TESTING AND DISCUSSION

The proposed system is intended to be evaluated using live webcam video. Testing should include multiple objects in a single frame, different object sizes and distances, partial occlusion, indoor and outdoor scenes, daylight, low-light conditions, and changing backgrounds.

- The principal observations to record are:
- Whether the system detects the intended object classes;
- Whether bounding boxes correctly surround the objects;
- The confidence values produced by YOLOv8;
- The number of missed and incorrect detections;
- The processing time per frame; and
- The average FPS obtained on the available computer.

Published results should not be copied into the experimental results of this project. For example, Kolukula et al. reported



76.5–93.3% accuracy, 0.895 mAP, and 43.95 FPS in their own study [5]. Ahmed et al. reported average precision of 0.79, recall of 0.62, and an F1-score of 0.69 in their YOLOv8 edge-intelligence study [2]. These values provide literature context only. The performance of the present prototype must be measured from the actual implementation.

XI. ADVANTAGES

The proposed software prototype provides the following advantages:

- Real-time object detection using a webcam;
- Detection of multiple objects in a single frame;
- Bounding boxes and labels for clear visualization;
- Confidence scores for interpreting model predictions;
- Practical implementation using Python and OpenCV;
- Relatively low hardware requirements during initial development;
- Suitability for academic study of autonomous-vehicle perception; and
- Scope for integration with additional perception modules.

XII. LIMITATIONS

The current prototype has several limitations:

- Camera-only input does not provide direct depth information;
- Detection performance may decrease in fog, rain, darkness, glare, or heavy occlusion;
- Processing speed depends on the available CPU/GPU and model configuration;
- A webcam prototype cannot represent all conditions of a real autonomous vehicle;
- Object detection alone does not perform path planning, decision-making, or vehicle control; and
- Safety-critical vehicle deployment requires extensive validation, redundancy, and safety engineering.

XIII. FUTURE WORK

13.1 Lane Detection

Lane and road-marking detection can be integrated to provide information about the drivable path.

13.2 Traffic Sign and Traffic Light Recognition

Specialized recognition models can be added to identify and interpret traffic signs and traffic signals.

13.3 Object Tracking

Detected vehicles and pedestrians can be tracked across consecutive frames to understand their movement.

13.4 Distance and Speed Estimation

Depth estimation, stereo cameras, LiDAR, or other sensing methods can be added to estimate distance and relative speed.

13.5 Collision Warning

A warning module can be developed to alert the user when an obstacle is detected within a potentially unsafe distance.

13.6 Edge Deployment

The model can be optimized and deployed on embedded AI devices. Edge-based processing is particularly relevant to real-time autonomous-driving applications.

13.7 Adverse Weather Robustness

Additional training data and preprocessing methods can be investigated for rain, fog, low light, glare, and other challenging conditions.

13.8 Integration with Autonomous Driving

Object detection can eventually be combined with localization, mapping, sensor fusion, prediction, path planning, and vehicle control to form a more complete autonomous-driving system.



XV. CONCLUSION

This paper presented a software-based real-time object detection system for autonomous-vehicle perception using YOLOv8, Python, and OpenCV. The proposed workflow captures live video using a webcam, processes individual frames, performs object detection, and displays the detected objects using bounding boxes, labels, and confidence scores.

The literature indicates that deep-learning-based object detection has become an important part of autonomous-vehicle perception, while real-time processing, computational cost, data diversity, sensor limitations, and robustness under changing environmental conditions remain important challenges [1,7].

The proposed system provides a practical academic implementation of the perception stage and can be developed on a conventional computer. It does not represent a complete autonomous-driving system, but it provides a foundation for future modules such as lane detection, object tracking, traffic-sign recognition, distance estimation, collision warning, and edge deployment.

Actual numerical performance of the proposed system should be reported after controlled experiments using the implemented prototype. Published values from previous studies are useful for comparison but should not be substituted for the project's own measurements.

REFERENCES

1. E. Ofoegbu, "Artificial intelligence and machine learning in autonomous driving: current trends and future directions: a review," *Journal of Engineering and Applied Science*, vol. 73, no. 251, 2026, doi: 10.1186/s44147-026-01112-5.
2. I. Ahmed, M. Ahmad, M. U. R. Siddiqi, A. Chehri, and G. Jeon, "Towards AI-Powered Edge Intelligence for Object Detection in Self-Driving Cars: Enhancing IoV Efficiency and Safety," accepted for publication in *IEEE Internet of Things Journal*, 2025.
3. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
4. A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja, "Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues," *Array*, vol. 10, Art. no. 100057, 2021, doi: 10.1016/j.array. 2021.100057.
5. N. R. Kolukula, R. P. Kalapala, S. S. R. Ivaturi, R. K. Tammineni, M. Annavarapu, and U. Pyla, "An efficient object detection by autonomous vehicle using deep learning," *International Journal of Electrical and Computer Engineering*, vol. 14, no. 4, pp. 4287–4295, Aug. 2024, doi: 10.11591/ijece.v14i4.pp4287-4295.
6. X. Jia, Y. Tong, H. Qiao, M. Li, J. Tong, and B. Liang, "Fast and accurate object detector for autonomous driving based on improved YOLOv5," *Scientific Reports*, vol. 13, Art. no. 9711, 2023, doi: 10.1038/s41598-023-36868-w.
7. A. Balasubramaniam and S. Pasricha, "Object Detection in Autonomous Vehicles: Status and Open Challenges," arXiv preprint arXiv:2201.07706, 2022.
8. Ultralytics, "YOLOv8: A State-of-the-Art Object Detection Model for Real-Time Applications," Ultralytics Documentation, 2023.
9. H. Zhu, L. Yang, X. Chen, et al., "Deep learning-based perception systems for autonomous driving: A comprehensive survey," *Neurocomputing*, vol. 489, pp. 255–270, 2022, doi: 10.1016/j.neucom. 2021.08.155.
10. L. G. Galvao, M. Abbod, T. Kalganova, V. Palade, and M. N. Huda, "Pedestrian and Vehicle Detection in Autonomous Vehicle Perception Systems—A Review," *Sensors*, vol. 21, no. 21, Art. no. 7267, 2021, doi: 10.3390/s21217267.
11. A. Balasubramaniam and S. Pasricha, "Object Detection in Autonomous Vehicles: Status and Open Challenges," arXiv preprint arXiv:2201.07706, 2022, doi: 10.48550/arXiv. 2201.07706.



12. M. Masmoudi, H. Ghazzai, M. Frikha, and Y. Massoud, "Object Detection Learning Techniques for Autonomous Vehicle Applications," in *Proc. IEEE Int. Conf. Vehicular Electronics and Safety (ICVES)*, Cairo, Egypt, 2019, pp. 1–5, doi: 10.1109 /ICVES. 2019.8906437.
13. T. Karasawa, K. Watanabe, Q. Ha, A. Tejero-De-Pablos, Y. Ushiku, and T. Harada, "Multispectral Object Detection for Autonomous Vehicles," in *Proc. Thematic Workshops of ACM Multimedia 2017*, 2017, pp. 35–43, doi: 10.1145/3126686.3126727.
14. R. Ravindran, M. Santora, and M. Jamali, "Multi-Object Detection and Tracking, Based on DNN, for Autonomous Vehicles: A Review," *IEEE Sensors Journal*, vol. 21, no. 5, pp. 5668–5677, 2021, doi: 10.1109 /JSEN. 2020.3036285.
15. "Object Detection Under Rainy Conditions for Autonomous Vehicles: A Review of State-of-the-Art and Emerging Techniques," IEEE Xplore, 2020, doi: 10.1109/9307324.

