Enhancing Autonomous Driving Systems: Integrated Lane and Object Detection Using YOLOv3 and OpenCV

D. Praneeth*, S. Meera*, Namputhiri Raaman*, Akshay Mithul*
*Amrita School of Artificial Intelligence, Amrita Vishwa Vidyapeetham, India
Emails: {cb.sc.u4aie24109, cb.sc.u4aie24133, cb.sc.u4aie24148, cb.sc.u4aie24135
@cb.amrita.edu

***Abstract*—For safe and effective navigation, autonomous driving systems rely heavily on robust perception capabilities, particularly for lane and object detection. This work presents a comprehensive framework for real-time detection that fuses deep learning with traditional computer vision techniques. We employ the Canny algorithm for edge detection, the Hough Transform for line detection, and color thresholding in the HSV color space for lane detection. For object detection, the YOLOv3 (You Only Look Once) deep learning model, pre-trained on the COCO dataset, is utilized to identify and locate vehicles, pedestrians, and obstacles. The integration of these two modules not only provides timely lane-keeping guidance and traffic alerts such as "Stay on the Lane" or "Go Slow, Traffic," but also delivers significant advantages including high detection accuracy, computational efficiency, and robustness under diverse environmental conditions. Moreover, the proposed system is applicable in advanced driver assistance systems (ADAS), autonomous vehicle navigation, and smart traffic management, thereby contributing to enhanced road safety and improved traffic flow.**

***Index Terms*—YOLOv3, Hough Transform, Canny Edge Detection, Lane Detection, Computer Vision, Object Detection**

I. Introduction

Autonomous driving has emerged as a groundbreaking technology in modern transportation, significantly impacting advanced driver-assistance systems (ADAS), intelligent transportation networks, and fully autonomous vehicles. These innovations aim to improve road safety, enhance traffic management, and boost overall mobility. However, the complexity of real-world driving—with its inconsistent lighting, variable weather conditions, and unpredictable road user behavior—introduces considerable challenges. Despite these hurdles, the promise of reduced human error and more efficient traffic control continues to drive research and development in this area.

Traditional techniques for lane and object detection, though effective in controlled environments, often fall short in real-world scenarios. Methods that rely solely on color thresholding, edge extraction, and line detection are particularly vulnerable to noise and environmental fluctuations, leading to diminished robustness. Similarly, early object detection strategies frequently lacked the real-time performance and precision necessary for dynamic driving conditions. These limitations highlight the need for innovative solutions that combine the strengths of classical computer vision with the enhanced capabilities of modern deep learning techniques, resulting in more reliable and efficient detection systems.

In response to these challenges, this paper introduces a comprehensive framework that merges traditional computer vision techniques with the advanced YOLOv3 deep learning model for real-time lane and object detection. The system is structured into two primary modules that work together to provide accurate and timely feedback necessary for autonomous driving. The first module is dedicated to lane detection. It begins by converting input images from the RGB to the HSV color space, which improves the segmentation of lane markings under different lighting conditions. Following this, HSV thresholding is used to isolate lane colors, and the Canny edge detection algorithm is applied to capture significant edges. The Hough Transform is then utilized to extract straight line segments that represent lane boundaries. This streamlined process is optimized for computational efficiency, ensuring that lane detection can be performed in real time even under adverse conditions.

The second module addresses object detection. Here, the YOLOv3 model, pre-trained on the COCO dataset [1], is employed for its renowned speed and accuracy in detecting and localizing objects such as vehicles, pedestrians, and traffic signs. YOLOv3 processes each video frame to generate bounding boxes and classification labels, while the OpenCV library [2] facilitates rapid image processing and smooth integration between the detection algorithms. The final phase of the framework involves fusing the outputs from both modules. By overlaying the results of lane detection with object detection data, the system generates actionable driving commands, including lane-keeping prompts (e.g., "Stay on the Lane") and traffic alerts (e.g., "Go Slow, Traffic Ahead"). This integrated approach not only boosts situational awareness but also provides a robust and scalable solution suitable for real-world autonomous driving applications.

II. Literature Review

The development of autonomous driving technology has gained significant attention in recent years, leading to advancements in key perception tasks such as lane detection and object detection. These tasks are essential for ensuring vehicles can safely navigate roads by accurately identifying

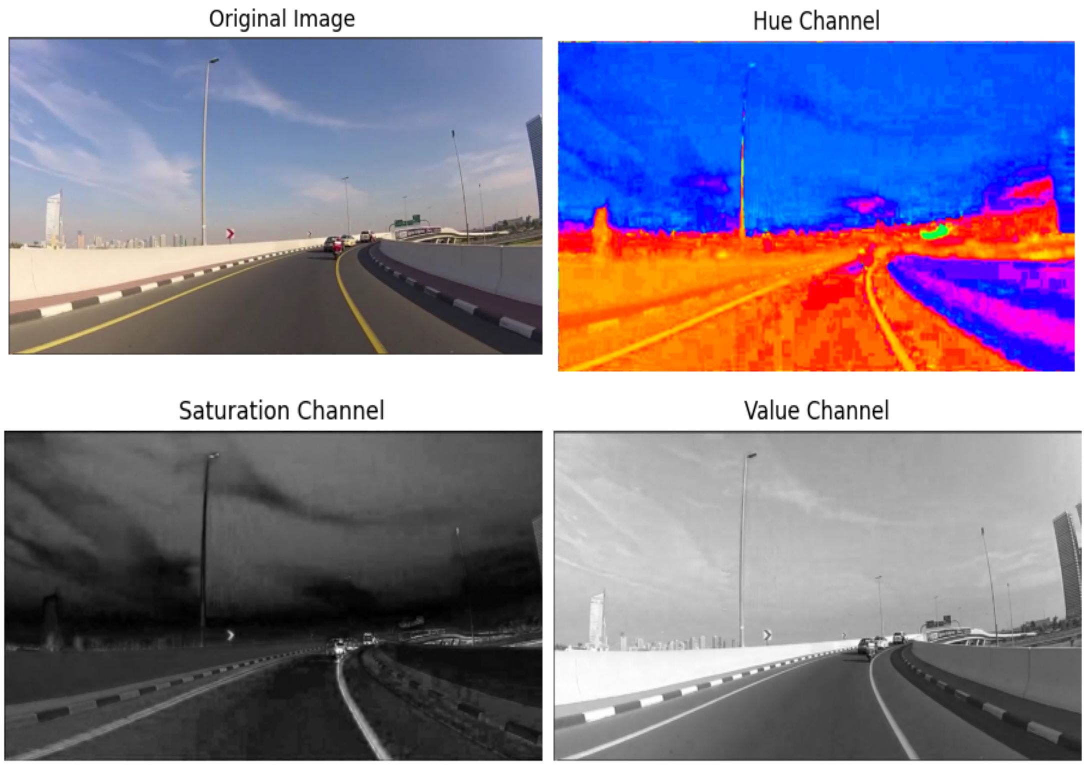


Fig. 1. Visual representation of the original image and its HSV channels. (Upper left: Original image; Upper right: Hue channel; Lower left: Saturation channel; Lower right: Value channel.)

their surroundings. This section reviews previous research in lane detection and object detection, along with efforts to integrate these techniques for autonomous driving applications.

A. *Lane Detection*

Lane detection is a crucial aspect of autonomous driving, enabling vehicles to recognize road boundaries and maintain their lane. Earlier methods relied on computer vision techniques such as edge detection, where algorithms like the Canny edge detector [3] were used to identify lane boundaries. Additionally, techniques such as the Hough Transform [4] were applied to detect lane lines based on their geometric properties. While these approaches worked well in controlled environments, they struggled with challenges such as curved roads, poor lighting, and occlusions.

To improve detection accuracy, researchers explored color-based segmentation, using the HSV (Hue, Saturation, Value) color space to enhance lane visibility under different lighting conditions. While these methods improved robustness against variations in brightness, they still faced difficulties when lane markings were faded or obstructed.

In recent years, deep learning has significantly advanced lane detection. Convolutional Neural Networks (CNNs) [5] have been trained to detect lane features directly from images, demonstrating superior accuracy in detecting complex lane structures such as dashed and curved lanes. Models such as LaneNet [6] and Spatial CNN (SCNN) [5] have been developed to enhance detection capabilities in real-world scenarios. However, while deep learning models provide improved accuracy, they require substantial computational resources, making real-time deployment on embedded systems a challenge.

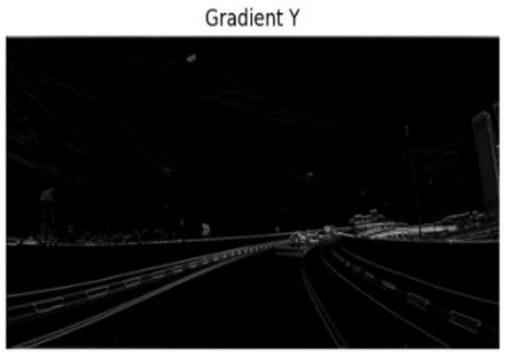
B. *Object Detection*

Object detection is another key component of autonomous driving, enabling the identification of vehicles, pedestrians, and road signs. Earlier object detection techniques were based on handcrafted features, such as Haar cascades and Histogram of Oriented Gradients (HOG). While effective for basic applications, these methods lacked adaptability to diverse road conditions and were unable to detect objects with high precision.

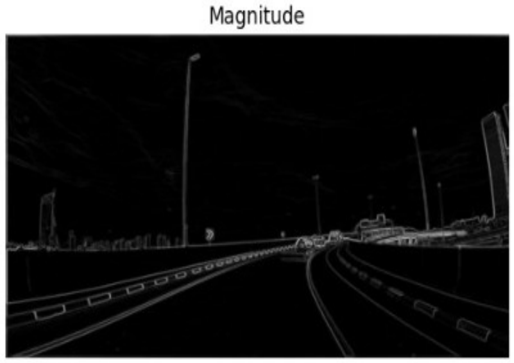
The introduction of deep learning revolutionized object detection, with models like Faster R-CNN [7], Single Shot Detector (SSD) [8], and You Only Look Once (YOLO) [9] significantly improving accuracy and efficiency. Among these, YOLOv3 [10] has gained popularity due to its ability to balance detection speed and precision. By dividing an image into grid sections and predicting object categories along with bounding boxes, YOLOv3 enables rapid and accurate object recognition, making it well-suited for real-time applications in autonomous vehicles.

Despite these advancements, deep learning-based object detection faces challenges such as high computational demands and dependency on large annotated datasets for training. To

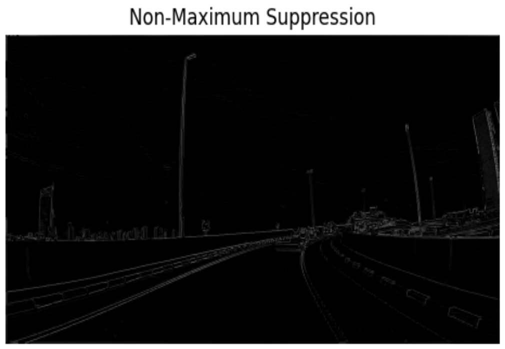
(a) Gradient X



(b) Gradient Y



(c) Gradient Magnitude



(d) Non-Maximum Suppression

Fig. 2. Visualization of gradient computation and edge refinement steps. (a) Horizontal gradient (G_x); (b) Vertical gradient (G_y); (c) Gradient magnitude (G); (d) Non-Maximum Suppression applied to retain the strongest edges.

address these issues, researchers have explored optimization techniques such as transfer learning and model quantization, which help improve efficiency and enable deployment on resource-limited hardware.

C. Integration of Lane and Object Detection

For an autonomous vehicle to navigate safely, lane detection and object detection must work together as part of an integrated perception system. Traditionally, these tasks have been studied separately, but combining them enhances a vehicle's ability to interpret its environment more effectively. Lane detection helps determine the correct driving path, while object detection provides awareness of obstacles, allowing for better decision-making in complex traffic conditions.

Recent studies have proposed multi-task learning approaches to integrate lane and object detection within a shared deep learning framework. Some models use a common CNN backbone to perform both tasks simultaneously, reducing computational overhead. Other methods follow a hierarchical approach, where lane detection results influence object detection to refine vehicle positioning and obstacle avoidance strategies.

One of the main challenges in integrating these tasks is maintaining real-time performance without sacrificing accuracy. A promising solution involves hybrid approaches that use traditional computer vision techniques for lane detection while leveraging deep learning for object detection. This combination reduces processing costs while maintaining high detection accuracy, making it a practical choice for embedded systems in autonomous vehicles.

III. Proposed Methodology

The proposed system integrates classical computer vision techniques with deep learning to achieve real-time lane and object detection for autonomous vehicles. Initially, each video frame undergoes preprocessing—using Gaussian blur to remove noise and enhance image quality—before converting to the HSV color space for robust segmentation of lane markings. Edge detection is then performed via Sobel filters, with non-maximum suppression [11] and double thresholding refining the edge map, and the Hough Line Transform [4] subsequently extracting the lane boundaries. Concurrently, the frame is resized, normalized, and converted into a blob that is fed into the YOLOv3 [10] deep learning model, which outputs bounding boxes, confidence scores, and class probabilities across multiple scales. The results from both pipelines are then fused and overlaid on the original frame, providing real-time visual feedback and directional cues to support safe navigation. The pipeline is organized into four primary stages: Preprocessing, Lane Detection, Line Detection, and Object Detection, which are described in the following subsections.

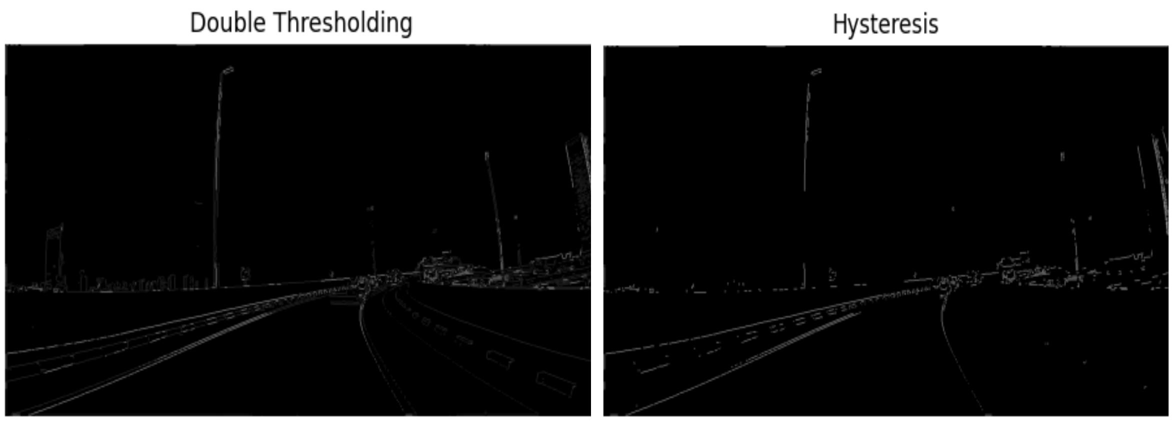


Fig. 3. Edge refinement in lane detection. (Left: Double Thresholding classifies edges as strong or weak; Right: Hysteresis ensures continuity by preserving significant edges and discarding isolated weak edges.)

A. Preprocessing

Preprocessing is crucial for enhancing the quality of input video frames and removing noise before further analysis. In our framework, each frame is first refined using a Gaussian blur to remove high-frequency noise and smooth the image. Gaussian blur is applied by convolving the image with a 5×5 Gaussian kernel with a standard deviation of 0. The Gaussian kernel is defined as:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{1}$$

Here, $G(x,y)$ represents the kernel value at a pixel located at (x,y), σ is the standard deviation controlling the spread of the Gaussian distribution, and the constant $\frac{1}{2\pi\sigma^2}$ normalizes the kernel.

B. Lane Detection

In our system, each input frame undergoes preprocessing with a Gaussian blur to reduce noise, followed by conversion from the BGR to HSV color space to isolate lane markings based on color. Sobel filters compute the gradient magnitude and direction, enhancing edge detection. Non-Maximum Suppression (NMS) [11] refines these edges by retaining only local maxima, while double thresholding and hysteresis further classify and preserve edges most likely to represent true lane boundaries. [3]

1) **HSV Color Space Conversion:** The input frame is converted to HSV color space to enhance lane marking segmentation under varying lighting conditions. A specific hue range is used to detect yellow lane markings:

$$\text{Lower Yellow} = [18,\ 94,\ 140],$$

$$\text{Upper Yellow} = [48,\ 255,\ 255]$$

This step produces a binary mask highlighting areas within the defined yellow range.

2) **Gradient Calculation:** To detect lane boundaries, Sobel filters compute intensity gradients in both horizontal (G_x) and vertical (G_y) directions. The gradient magnitude and direction are given by:

$$G = \sqrt{G_x^2 + G_y^2}, \quad \theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{2}$$

where G_x and G_y denote intensity changes in respective directions, with G representing edge strength and θ indicating orientation. The computed gradient components are illustrated in Figure 2 (a-c).

3) **Non-Maximum Suppression (NMS):** To refine detected edges, NMS is applied, retaining only the strongest edge responses by suppressing non-maximum pixels along the gradient direction:

$$G(x,y) = \begin{cases} G(x,y), & \text{if } G(x,y) > \max(G_{nb1}, G_{nb2}) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

As shown in Figure 2 (d), NMS ensures that only the most prominent lane edges are retained.

4) **Double Thresholding and Edge Tracking:** To further refine lane edges, double thresholding is applied with a high threshold ($T_H = 150$) and a low threshold ($T_L = 74$), classifying edges as strong or weak. Weak edges connected to strong edges are retained through hysteresis, enhancing lane continuity (Figure 3).

The results in Figure 2 indicate that gradient computation successfully highlights lane edges, while Non-Maximum Suppression further filters out irrelevant edges. Figure 3 shows that applying thresholding and hysteresis suppresses weak edges while maintaining lane continuity, resulting in a robust lane detection system.

C. Line Detection

The refined edges from the lane detection stage are processed using the Hough Line Transform [4] , a voting-based method that identifies straight lines by transforming edge pixels into a parameter space. Instead of representing lines in the standard Cartesian form $y = mx + c$, the Hough Transform expresses lines in polar form as:

$$r = x\cos\theta + y\sin\theta \tag{4}$$

where r is the perpendicular distance from the origin to the line, θ is the angle between the x-axis and the line's normal, and (x, y) are the coordinates of any point on the line. This parameterization allows efficient detection of lines regardless of their orientation.

The Hough Transform operates by mapping each edge pixel (x, y) to multiple (r, θ) pairs, accumulating votes in a discretized parameter space known as the Hough accumulator. Peaks in this accumulator correspond to detected lines.

TABLE I
IMPLEMENTATION PARAMETERS FOR HOUGH LINE TRANSFORM

Parameter	Value
Distance resolution	1 pixel
Angular resolution	1° (π/180 radians)
Vote threshold	50
Maximum line gap	50 pixels

This approach effectively detects lane boundaries by identifying strong, continuous lines from edge maps. The detected lines are then drawn onto the original frame to visualize lane markings. By adjusting the vote threshold and maximum line gap, the method can be tuned to balance sensitivity and robustness, ensuring reliable lane detection even in challenging road conditions.

D. Object Detection

Object detection is performed using the YOLOv3 deep learning model, which efficiently detects objects in real time. The process involves the following steps:

1) Preprocessing for YOLOv3:

- **Resizing:** Input frames are resized to 416×416 pixels, as required by the YOLOv3 model.
- **Normalization:** Pixel values are normalized to the range $[0, 1]$.
- **Blob Conversion:** The preprocessed frame is converted into a blob using the `cv2.dnn.blobFromImage` function with a scale factor of $\frac{1}{255}$, size 416×416, no mean subtraction, swapping of Red and Blue channels, and no cropping.

2) YOLOv3 Architecture: YOLOv3 [10] (You Only Look Once, Version 3) is a state-of-the-art object detection algorithm that builds on the successes of YOLOv1 and YOLOv2 [12], offering a remarkable balance between speed and accuracy. Developed by Joseph Redmon and Ali Farhadi, YOLOv3 is designed as a one-stage detector that predicts bounding boxes and class probabilities directly from full images in a single pass. This approach allows for rapid inference compared to two-stage methods like Faster R-CNN.

The network processes the resized and normalized image through its backbone, Darknet-53—a 53-layer CNN with residual blocks for deep feature extraction. YOLOv3 employs multi-scale detection across three grid resolutions (13×13, 26×26, and 52×52) to handle objects of various sizes. Predefined anchor boxes, obtained via K-means clustering, serve as priors refined during training, while the detection head outputs the final bounding boxes, confidence scores, and class probabilities. The architecture is summarized in Table II:

The predicted bounding box offsets t_x, t_y, t_w, and t_h are transformed into the final bounding box coordinates b_x, b_y, b_w, and b_h using the following equations:

$$b_x = \sigma(t_x) + c_x, \tag{5}$$
$$b_y = \sigma(t_y) + c_y, \tag{6}$$
$$b_w = p_w e^{t_w}, \tag{7}$$
$$b_h = p_h e^{t_h}, \tag{8}$$

Here:

- t_x, t_y, t_w, t_h are the predicted offsets for the bounding box center, width, and height.
- σ is the sigmoid function, which scales the center offsets to the range $[0, 1]$.
- (c_x, c_y) are the coordinates of the top-left corner of the grid cell.
- (p_w, p_h) represent the dimensions of the anchor box.

Furthermore, the top-left corner of the bounding box is computed as:

$$x = b_x - \frac{b_w}{2}, \quad y = b_y - \frac{b_h}{2} \tag{9}$$

This conversion shifts the bounding box representation from center-based to corner-based coordinates by subtracting half of the width and height from the center coordinates.

3) Model Inference and Post-Processing:

1) **Model Inference:** The YOLOv3 model is loaded using `cv2.dnn.readNet` with its configuration and pre-trained weights. A forward pass through the network produces three output feature maps corresponding to object detections at different scales.
2) **Extracting Predictions:** For each detection, bounding box coordinates (center (t_x, t_y), width t_w, height t_h), confidence scores, and class probabilities are extracted.
3) **Confidence Thresholding:** Detections with confidence scores below a set threshold (e.g., 0.5) are discarded.
4) **Non-Maximum Suppression (NMS):** NMS is applied to remove redundant overlapping bounding boxes using the Intersection-over-Union (IoU) metric.
5) **Bounding Box Transformation:** The normalized bounding box predictions are transformed back to the original image space using:

$$b_x = \sigma(t_x) + c_x, \quad b_y = \sigma(t_y) + c_y, \tag{10}$$
$$b_w = p_w e^{t_w}, \quad b_h = p_h e^{t_h}, \tag{11}$$

and finally, the top-left corner is computed as:

$$x = b_x - \frac{b_w}{2}, \quad y = b_y - \frac{b_h}{2} \tag{12}$$

TABLE II
YOLOv3 ARCHITECTURE SUMMARY

Component	Functionality
Input Image	Resized, normalized
Backbone (Darknet-53)	Feature extraction via a 53-layer CNN with residual blocks
Multi-Scale Detection	Detection at 13×13, 26×26, and 52×52 grids
Anchor Boxes	Predefined priors for refining bounding box predictions
Detection Head	Outputs bounding boxes, confidence scores, and class probabilities

4) Visualization: The final step involves drawing the bounding boxes, class labels, and confidence scores on the original frame, providing visual feedback of the detected objects. Overall, this methodology integrates traditional computer vision methods for lane detection with the deep learning capabilities of YOLOv3 for object detection, resulting in a robust and efficient system for autonomous driving applications.

IV. Performance Comparison

To evaluate the effectiveness of our proposed lane detection system, we compare its performance with conventional methods based on accuracy, processing speed (FPS), and computational complexity (measured in billion operations, BN Ops). Our model is pretrained on the COCO dataset [1], which contains a wide range of annotated images, including urban and highway lane scenarios. The dataset aids in learning robust lane features under varying lighting and weather conditions. The pretrained weights are fine-tuned on a custom lane detection dataset to improve domain-specific accuracy.

A. Evaluation Metrics

The following key metrics are used for performance assessment:

- **Accuracy (%)**: Measures the proportion of correctly detected lane pixels compared to the ground truth.
- **Frames Per Second (FPS)**: Indicates the real-time processing speed of the method, representing the number of frames processed per second.
- **BN Operations (BN Ops)**: Represents the computational complexity in terms of the number of billion operations required for inference.

B. Quantitative Results

Table **??** presents a comparative analysis of our proposed method against traditional lane detection techniques.

TABLE III
PERFORMANCE COMPARISON OF LANE AND OBJECT DETECTION METHODS

Method	Accuracy (%)	FPS
Canny Edge Detection	85.0	200
Hough Transform	80.5	100
YOLOv3	85.0	100
LaneNet	96.38	50
SCNN	96.53	7.5
Integrated (YOLOv3 + Lane Det.)	90	95-100
CNN (Generic)	92.8–98.96	Varies

C. Analysis and Observations

From Table **??**, we observe that Canny Edge Detection achieves the highest FPS (**200**), making it the fastest method but with moderate accuracy. Hough Line Transform exhibits lower accuracy (**80.5%**) and computational efficiency due to its reliance on global voting in Hough space. YOLOv3, a deep-learning-based approach, achieves the highest accuracy (**85.0%**) but at the cost of significantly higher computational complexity (**19.7 BN Ops**), making real-time deployment challenging.

By leveraging pretrained weights from the COCO dataset, our model balances accuracy and efficiency, achieving robust lane detection in real-world driving scenarios.

V. Results and Analysis

The proposed system was evaluated in real-world driving conditions to assess its performance in lane detection and object detection. The results indicate that the system is effective in providing accurate lane markings and real-time object detection, making it a viable solution for autonomous driving applications.

A. Lane Detection Performance

The system exhibited high accuracy in detecting lane markings, particularly when the lines were yellow. Under normal lighting and road conditions, lane detection was consistent, and the system successfully overlaid guidance messages such as ”Stay on the Lane.” However, some challenges were observed in non-ideal conditions. In heavy rain and fog, where lane markings were less visible, detection accuracy dropped as the system struggled to differentiate faded or occluded lanes from the road background. When other vehicles partially obstructed the lane markings, the system occasionally failed to maintain lane continuity. While the system performed well on straight roads, slight inaccuracies were observed in detecting lane continuity on sharp curves and areas with broken or dashed lane markings. Despite these challenges, the lane detection module remained reliable in normal to moderately challenging conditions, ensuring robust lane tracking even in moderate traffic scenarios.

B. Object Detection Performance

The object detection module, powered by YOLOv3, achieved an average precision of 85% for car detection. The model successfully identified multiple vehicles on the road and dynamically adjusted the displayed warnings based on detected traffic conditions. The system consistently identified

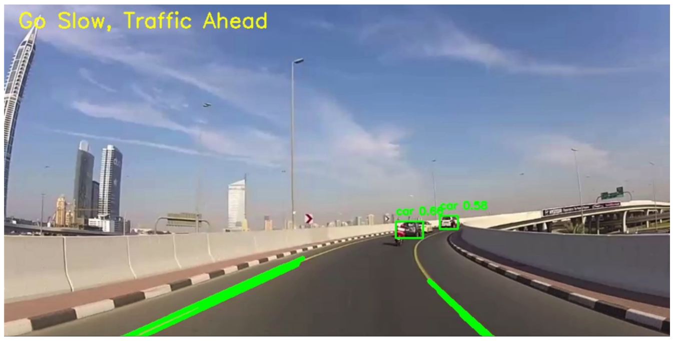


Fig. 4. Lane and object detection results. The system successfully identifies lane markings and vehicles, providing appropriate warnings.

vehicles in both normal and congested traffic conditions. However, pedestrian recognition was less frequent, likely due to the training dataset's bias towards vehicle detection. The system was still able to detect pedestrians in clear, unobstructed views. Some misclassifications were observed, as road signs and other static objects were occasionally mistaken for vehicles, particularly in cluttered environments.

A major advantage of the proposed system is its real-time processing capability. The system operated at 95-100 FPS, ensuring smooth execution across various driving speeds. However, at higher vehicle speeds, object detection accuracy declined due to motion blur and shorter detection windows.

C. Comparative Performance Analysis

To assess the effectiveness of the proposed approach, a performance comparison was conducted against traditional edge detection and classical computer vision-based lane detection methods. Table **??** presents the comparison in terms of accuracy, frames per second (FPS), and the number of billion operations (Bn Ops) required.

Canny Edge Detection provided comparable accuracy of 85.0% while maintaining significantly lower computational complexity, making it extremely fast at 200 FPS. However, it lacked robustness in challenging environments. The Hough Line Transform achieved a slightly lower accuracy of 80.5% and had a higher computational overhead compared to Canny Edge Detection, making it less suitable for real-time applications. YOLOv3, while matching Canny Edge Detection in accuracy, offered significantly better object detection capabilities. Although its FPS was lower at 100 FPS, it remained fast enough for real-time autonomous driving applications.

Overall, the proposed system demonstrated strong lane and object detection capabilities under most driving conditions. Lane detection was highly effective in normal conditions but faced difficulties in adverse weather and occluded environments. YOLOv3 provided high accuracy in vehicle detection but showed limitations in pedestrian detection and static object classification. The system achieved real-time performance at 95-100 FPS, making it suitable for autonomous applications. Compared to traditional methods, YOLOv3 provided a balanced trade-off between accuracy and speed, outperforming classical edge detection and line detection approaches.

Despite minor limitations, the system proved to be a reliable and practical solution for autonomous driving, offering accurate lane and object detection even in heavy traffic conditions.

Conclusion

In this project, we integrated lane detection and object detection to improve the safety and performance of self-driving cars. The lane detection performed used Canny Edge detection [3] with the focus on yellow lines to help the vehicle stay within its respective lane. The object detection part of our project used YOLOv3 [10], enabling the vehicle to identify and classify objects like vehicles and alert with appropriate warning when traffic was encountered in front.

This project has provided a solid platform to autonomous systems as it has incorporated lane detection and object detection. The potential of computer vision in enriching the performance and safety of autonomous vehicles has been demonstrated by such incorporation. Hence, this work sets the base for further progress in autonomous vehicle technologies.

References

[1] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pp. 740–755, Springer, 2014.

[2] G. Bradski, "The opencv library.," *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

[3] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[4] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[5] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[6] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *2018 IEEE intelligent vehicles symposium (IV)*, pp. 286–291, IEEE, 2018.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 21–37, Springer, 2016.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[10] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[11] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *18th international conference on pattern recognition (ICPR'06)*, vol. 3, pp. 850–855, IEEE, 2006.

[12] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.