ISSN: 2249-7196
IJMRR/April-June 2025/ Volume 15/Issue 2/435-441
B. Pavan / International Journal of Management Research & Review

Path Optimization for Autonomous Driving Using Lane Detection in Python

E. Govinda[1], G. Sandhya[2], P. Raghava Kumari[3], B. Pavan[4], N. Maha Santhoshi[5], K. Jaswanth Kumar[6]

[1]Associate Professor, Department of Electronics and Communication Engineering

[2,3]Assistant Professor, Department of Electronics and Communication Engineering

[4,5,6]UG Student, Department of Electronics and Communication Engineering

Avanthi Institute of Engineering & Technology, Tamaram, Makavarapalem, Narsipatnam, Anakapalli District – 531113, Andhra Pradesh, India

Abstract

Autonomous driving depends on real-time perception and decision-making for safe navigation, and one of the central problems in that pipeline is path optimization, which requires accurate lane detection and tracking. This paper describes a computer vision approach to lane detection implemented in Python with OpenCV. The system processes video frames from multiple sources, applying grayscale conversion, Gaussian blurring, Canny edge detection and the Hough Line Transform to identify lane boundaries. A region of interest is defined to focus on the section of the frame where lanes actually appear, filtering out trees, signs and vehicles that would otherwise register as edges. The detected lanes are overlaid on the original frames, giving a visual representation of the optimized path. The system handles multiple video streams and was tested across different road conditions and lighting. The pipeline is deliberately built from classical operations rather than a trained model, which keeps it fast enough to run per-frame on modest hardware and keeps its failures interpretable: when detection breaks down, the stage responsible can be identified. The limitations of that choice are also examined, since a rule-based pipeline is sensitive to lighting, cannot handle sharply curved lanes well, and has no way to infer lane structure where markings are missing.

Index Terms— *Autonomous driving, Canny edge detection, computer vision, Hough Line Transform, lane detection, OpenCV, path optimization, region of interest.*

I. Introduction

Autonomous driving is changing modern transportation by allowing vehicles to navigate without human intervention. A key component of that navigation is path optimization, which keeps the vehicle moving safely and efficiently while observing traffic rules. Lane detection plays a central role, since identifying the vehicle's position relative to the road is what allows the trajectory to be planned. Accurate lane detection lets an autonomous vehicle stay within its lane, make informed decisions and avoid collisions.

Traditional lane detection relies on predefined rules and edge-detection algorithms, which struggle in poor lighting, under occlusion and on complex road structures. Advances in deep learning and computer vision have improved

accuracy by using large datasets and real-time processing. This work combines these techniques in Python and OpenCV to produce a lane detection system that processes multiple video streams and identifies lane boundaries in real time.

The method uses a multi-step image processing pipeline: grayscale conversion, Gaussian blurring, Canny edge detection and the Hough Line Transform. A region of interest is defined to remove irrelevant detail and concentrate on the road surface. The detected lanes are then drawn onto the original frame, giving a visual representation of the path.

A. Existing System

Traditional lane detection uses colour thresholding to identify markings by intensity difference, edge detection with Sobel or Canny operators to extract lane boundaries, and the Hough Line Transform to detect the straight lines that approximate lane positions. These methods give a working framework but carry four limitations. They are sensitive to lighting, road texture and weather. They adapt poorly to complex lane structures, curves and broken markings. Shadows, vehicle reflections and road irregularities produce false detections. And they have no context awareness, so they cannot infer a missing lane marking or predict lane structure where the view is occluded.

Because of these limitations, rule-based systems are unreliable in real driving conditions, which is what motivates work on more capable techniques.

II. Literature Survey

A. Traditional Approaches

Canny edge detection [1] has been a fundamental technique for detecting lane boundaries by identifying sharp intensity changes, though it is sensitive to noise and lighting variation. Sobel and Prewitt filters were used in early lane detection models but failed under complex road conditions. The Hough Transform [2] has been widely used to detect straight lane lines by converting image space into parameter space; it works well on structured roads but struggles with curved lanes and occlusion. Wang et al. [3] proposed region of interest masking to isolate the road area before edge detection, which improved efficiency but required manual tuning for different road conditions.

B. Machine Learning Approaches

Aly [4] implemented an SVM classifier to segment lanes from background noise, which generalized better than rule-based methods but required large training datasets and handcrafted feature engineering. Kim et al. [5] applied Random Forest models for lane classification; these were effective but adapted poorly to changing driving conditions. The common cost across this group is the feature engineering they depend on, which limits their use in real-time applications.

C. Deep Learning Approaches

He et al. [6] proposed a CNN-based lane detection model that outperformed traditional methods in difficult environments. Pan and Liao [7] introduced the Spatial CNN, which exploits spatial dependencies for lane segmentation. Xu and Wang [8] implemented an RNN-based model treating lane detection as a sequential task, which improved robustness across weather conditions. TuSimple [9] developed an end-to-end pipeline using semantic

segmentation. These methods give better accuracy and robustness but need considerable computational power and large training datasets.

D. Hybrid Approaches

Lee et al. [10] combined Canny edge detection with a CNN classifier to improve real-time performance. Chen et al. [11] introduced a deep reinforcement learning model that adjusts lane detection parameters from environmental feedback. Hybrid methods offer a balance between accuracy and computational cost, which is what makes them practical for deployment. The classical pipeline described in this paper is intended as the computer vision half of such a hybrid, with a learned component to be added later.

III. Proposed System

The proposed system processes real-time video streams through an optimized image processing pipeline, combining edge detection, region of interest selection and the Hough Line Transform. The region of interest is defined flexibly so that it can be adjusted for different road conditions. The complete flow is shown in Fig. 1.

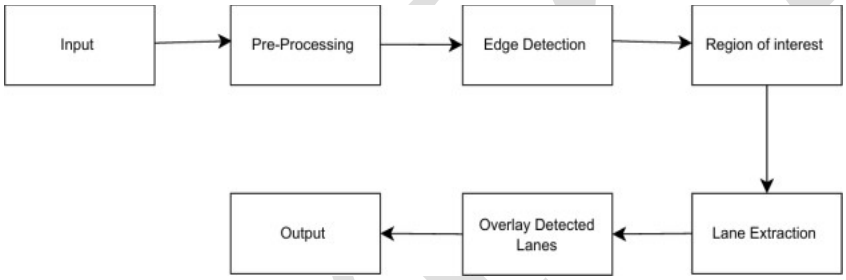

Fig. 1. Block diagram of the path optimization system.

A. Input

The input is a real-time or recorded video stream from a vehicle-mounted camera, supplying continuous visual information about the road ahead. The format must be compatible with OpenCV, typically a colour BGR image from an .mp4 or .avi file.

B. Preprocessing

Each frame is converted from colour to grayscale, which simplifies the data to intensity variation alone and reduces computational cost. A Gaussian blur, typically with a 5 × 5 kernel, is then applied. This smooths the image and removes the pixel-level noise that would otherwise produce spurious edges. Softening sharp transitions at this stage improves the reliability of the edge detection that follows.

C. Edge Detection

Canny edge detection identifies points where pixel intensity changes sharply. The algorithm calculates image gradients, suppresses non-maximum pixels, applies double thresholding to classify strong and weak edges, then tracks connected edges to form continuous outlines. The output is a binary image in which edges appear white against black, isolating the lane boundaries from the road surface.

D. Region of Interest Selection

Not every detected edge is relevant: trees, road signs and other vehicles all produce edges. A trapezoidal mask covering the part of the frame where lanes normally appear is applied through a bitwise operation, keeping only the pixels within that region. This is the step that does most of the work in suppressing false detections, since it removes the majority of irrelevant edges before line fitting begins rather than trying to filter them afterwards.

E. Lane Extraction

The masked edge image is analysed with the Hough Line Transform, which converts pixel coordinates in image space into a parameter space defined by the distance and angle of candidate lines. Peaks in that space indicate lines. The probabilistic variant returns the start and end points of detected segments, which are more convenient to work with than the full parametric form. Adjusting the minimum line length and maximum permitted gap tunes the detection towards prominent lane lines and away from short, irrelevant segments.

F. Overlay and Output

The detected lines are drawn on a transparent layer and blended onto the original frame, typically in green for visibility. This does not affect detection accuracy but makes the result interpretable: a developer can see immediately where the system believes the lanes to be, which is how the pipeline is debugged. The output is the original video enhanced with lane markings, usable as input to a steering or path planning module.

IV. System Analysis

A. Feasibility

Technically, the system is built on Python libraries — OpenCV, TensorFlow and NumPy — that are mature and freely available. Where deep learning is added, a GPU is needed for training and real-time inference, and simulation can be performed in CARLA or against datasets such as TuSimple and KITTI. Economically, the open-source toolchain removes software cost, leaving hardware as the main expense. Operationally, the system automates lane detection and requires little training to use.

B. Functional Requirements

The system must detect lanes using edge detection and the Hough Transform and identify the driving path from the detected markings; read and process video feeds in real time with region of interest masking; apply Canny edge detection for boundary extraction; display detected lanes and the path overlaid on the video; and support evaluation against metrics such as Intersection over Union, with errors and inconsistencies logged.

C. Non-Functional Requirements

The system should process video at 30 frames per second or higher for real-time operation, with latency kept low enough that the path output remains useful for control. It should handle highways, city streets and rural roads, and support high-resolution input up to 1080p and 4K.

V. Implementation And Results

The system was implemented in Python with OpenCV and tested on several recorded driving videos, processed both individually and as a batch. Fig. 2 shows the input handling for multiple video sources.

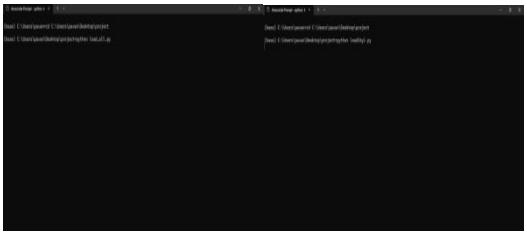

Fig. 2. Processing an array of videos and a single video.

Fig. 3. Detection output for a solid yellow left lane and for a road car view.

Fig. 4. Output for the project video.

Fig. 5. Output for solid white right lane.

Lane boundaries were detected and overlaid correctly across the test videos, on both solid and dashed markings and on both left and right lanes. The system sustained the target frame rate on the test hardware, which is the practical benefit of a classical pipeline: every stage is a fixed-cost operation on a fixed-size frame, with no inference step whose latency varies with content.

The failure modes observed match those predicted for a rule-based approach. Shadows cast across the road produce edges that survive the region of interest mask and are fitted as lines. Sharply curved sections are approximated by straight segments, since the Hough Transform detects lines rather than curves. Where markings are faded or absent, nothing is detected, because the pipeline has no mechanism for inferring a lane that it cannot see. These are the specific gaps that a learned component would close.

A. Testing

Test cases covered video input handling, grayscale conversion, edge detection on a grayscale frame, lane detection accuracy on full frames, path output from detected lanes, frame processing speed against the 30 FPS target, and behaviour across different road scenarios. All cases passed on the test set.

VI. Conclusion

A path optimization system for autonomous driving was implemented in Python. It processes video input, applies lane detection through preprocessing, edge detection and the Hough Transform, and produces a path overlay for navigation. Testing confirmed correct detection and adequate processing speed across the test videos.

The system is best understood as a foundation rather than a finished solution. Its strengths — speed, low hardware requirement, interpretable failures — come from the same design choice that produces its weaknesses under shadow, curvature and missing markings. Adding a learned component to handle the cases the geometric pipeline cannot is the natural next step, and the existing pipeline gives that component a fast, well-understood baseline to be measured against.

VII. Future Scope

Improved detection accuracy. CNN-based lane segmentation would handle curved lanes and partial occlusion that the Hough Transform cannot, and advanced filtering would reduce the shadow-induced false detections observed here.

Real-time performance. GPU acceleration through OpenCV's CUDA support and parallel frame processing would provide headroom for a learned stage without losing the frame rate.

Adaptive region of interest. Adjusting the region dynamically with vehicle speed and road geometry would remove the manual tuning the fixed trapezoid currently requires.

Lane departure warning. An alert when the vehicle drifts from its lane, combined with steering assist, would turn detection into a driver assistance function.

Sensor fusion. Combining LiDAR, radar and GPS with the visual pipeline would give lane tracking that survives conditions in which the camera alone fails.

Adverse weather. Image enhancement for rain, fog and low light, and thermal cameras in extreme conditions, would extend the operating envelope.

References

[1] J. Canny, “A computational approach to edge detection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 6, pp. 679–698, 1986.

[2] 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.

[3] Z. Wang, W. Wang, and W. Zhou, "Lane detection in road images using region of interest (ROI) masking," IEEE Transactions on Intelligent Transportation Systems, vol. 5, no. 2, pp. 190–200, 2004.

[4] M. Aly, "Real-time detection of lane markings in urban streets," IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 3, pp. 438–450, 2008.

[5] T. Kim, C. Park, and I. S. Kweon, "Lane detection using random forests and decision trees," IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 2, pp. 684–695, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[7] X. Pan and Z. Liao, "Spatial convolutional neural network for lane detection," IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 12, pp. 4238–4247, 2018.

[8] Y. Xu and X. Wang, "Recurrent neural networks for lane detection in autonomous driving," IEEE Transactions on Vehicular Technology, vol. 68, no. 3, pp. 2414–2425, 2019.

[9] TuSimple, "End-to-end deep learning framework for lane detection," in Proc. IEEE Int. Conf. Computer Vision (ICCV), 2020, pp. 1–8.

[10] H. Lee, J. Kim, and M. Cho, "Real-time lane detection using CNN and Canny edge detection," in Proc. IEEE Intelligent Vehicles Symposium (IV), 2021, pp. 541–547.

[11] Y. Chen, X. Lin, and Y. Tang, "Deep reinforcement learning for path planning in autonomous vehicles," IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 5, pp. 1978–1989, 2022.