PROJECT REPORT (CSN304)

A report submitted in partial fulfilment of the requirement for the course
ARTIFICIAL INTELLIGENCE

Part of the degree of

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING

Project Title:
LANE LINE DETECTION USING COMPUTER VISION (OpenCV)

Submitted to: Himani Sharma
Assistant Professor
School of Computing
DIT University, Dehradun

Submitted by:
Name: Rajeev Singh
Roll No. : 23A03D0305
SAP ID: 1000020122

SCHOOL OF COMPUTING DIT

UNIVERSITY, DEHRADUN

(State Private University through State Legislature Act No. 10 of 2013 of Uttarakhand and approved by UGC)

Mussoorie Diversion Road, Dehradun, Uttarakhand - 248009, India.

August-December 2025

INDEX

CERTIFICATION

I hereby certify that the work presented in this report entitled "Lane Line Detection Using Computer Vision (OpenCV)" is an authentic record of my own work carried out during the course of study under the supervision of

Himani Sharma(**Assistant Professor**).

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our respected guide, Ms. Himani Sharma Ma'am , for her constant guidance, valuable feedback, and continuous encouragement throughout the development of this project. Her expert supervision, insightful suggestions, and patient support have been instrumental in helping us understand the Artificial Intelligence, as well as in successfully implementing the features of our project.

We are also deeply thankful to our college administration and the Department of Computer Science for providing us with the resources, facilities, and a supportive learning environment to carry out this project. Their encouragement and trust in our abilities have motivated us to put forth our best efforts.

Finally, we extend our sincere appreciation to all those who directly or indirectly contributed to the successful completion of our project titled **"LANE LINE DETECTION USING COMPUTER VISION"** .

INTRODUCTION

Lane Line Detection is a critical component in modern driver assistance and autonomous vehicle systems. By identifying the boundaries of road lanes, the system helps vehicles stay within lanes, alert drivers to deviations, and ensure safer navigation. The project implements lane detection using computer vision techniques with OpenCV in Python.

The concept relies on image processing, edge detection, and geometric analysis. These algorithms process frames from road videos to detect and highlight lane lines. The use of Canny edge detection and Hough line transform ensures accurate detection even under varied lighting and environmental conditions.

This project demonstrates the practical implementation of computer vision algorithms and their real-world impact in the field of autonomous driving and intelligent transportation systems.

LITERATURE SURVEY/RELATED WORK

Several lane detection methods have been developed in the past decade, ranging from traditional image processing techniques to advanced deep learning-based approaches. Early research utilized edge detection and color thresholding, while recent systems employ convolutional neural networks (CNNs) and segmentation algorithms.

Traditional approaches, such as Canny edge detection and Hough line transform, are computationally efficient and effective for straight lane lines under clear visibility. However, they face challenges in curved lanes or poor lighting conditions. Deep learning-based systems, on the other hand, can learn features automatically and adapt to complex road geometries, but they require significant computational power and large datasets.

This project focuses on the classical vision-based approach using OpenCV, which is lightweight, efficient, and easy to understand for educational and research purposes.

REQUIREMENT ANALYSIS & SPECIFICATIONS

The system requires both hardware and software components. The hardware setup includes a computer with a camera or road image source. The software environment is built using Python with OpenCV and NumPy libraries.

The image is captured, processed, and analyzed in real time to detect lanes. The design is modular, allowing each step (edge detection, masking, line detection) to be individually tuned for better accuracy.

This system can run on standard consumer laptops, making it suitable for university-level research and experimentation.

Hardware Requirements

- Processor: Intel Core i3 or higher
- RAM: Minimum 4 GB
- Storage: 500 MB
- Camera (optional) for live video input

Software Requirements

- Programming Language: Python 3.x
- Libraries: OpenCV, NumPy
- IDE: VS Code / Jupyter Notebook

SYSTEM DESIGN

The architecture of the lane detection system consists of several sequential stages: image capture, preprocessing, feature extraction, and line detection. Each stage contributes to the overall robustness of the detection pipeline.

Below is a conceptual diagram illustrating the design flow of the system.

Figure 1: System Architecture Diagram (Input → Preprocessing → Edge Detection → Hough Transform → Lane Display)

CODE

```
import cv2

import numpy as np

import os


# Step 1: Create input and output folders if missing

os.makedirs("input", exist_ok=True)

os.makedirs("output", exist_ok=True)


# Step 2: Define region of interest

def region_of_interest(image):

    height = image.shape[0]

    polygons = np.array([

        [(100, height), (image.shape[1]-100, height), (image.shape[1]//2,
int(height/2) + 50)]

    ])

    mask = np.zeros_like(image)

    cv2.fillPoly(mask, polygons, 255)

    masked = cv2.bitwise_and(image, mask)

    return masked


# Step 3: Detect edges using Canny

def detect_edges(image):

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    blur = cv2.GaussianBlur(gray, (5, 5), 0)
```

```
    edges = cv2.Canny(blur, 50, 150)
    return edges

# Step 4: Detect and draw lines
def detect_lines(image, original):
    lines = cv2.HoughLinesP(image, 1, np.pi/180, 50, minLineLength=100, maxLineGap=50)
    line_image = np.zeros_like(original)
    if lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line[0]
            cv2.line(line_image, (x1, y1), (x2, y2), (0, 255, 0), 5)
    combined = cv2.addWeighted(original, 0.8, line_image, 1, 1)
    return combined

# Step 5: Process image
def process_image(inputroadjpeg):
    image = cv2.imread("input/Thrissur.jpeg")
    if image is None:
        print("❌ Error: Could not read image. Make sure road.jpg is in the 'input' folder.")
        return
    edges = detect_edges(image)
    cropped = region_of_interest(edges)
    result = detect_lines(cropped, image)
    output_path = os.path.join("output", "lane_output.jpg")
```

```
    cv2.imwrite(output_path, result)
    print(f"✅ Lane-detected image saved at: {output_path}")
    cv2.imshow("Lane Detection Result", result)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Step 6: Run the program
input_image_path = os.path.join("input", "Thrissu.jpg")
process_image(input_image_path)
```

IMPLEMENTATION MODULES

The implementation uses Python and OpenCV to perform lane detection through multiple processing steps. The process starts with reading an input image, converting it to grayscale, applying Gaussian blur to remove noise, and performing edge detection using the Canny method.

A region of interest is then defined to focus only on the part of the image that contains the road. The Hough Line Transform is applied to identify straight lines that correspond to lane boundaries. Finally, these detected lines are superimposed on the original image using green color to visualize the lanes.

The following figures show the lane detection outputs from two different road environments, demonstrating how the algorithm performs under natural and urban conditions.

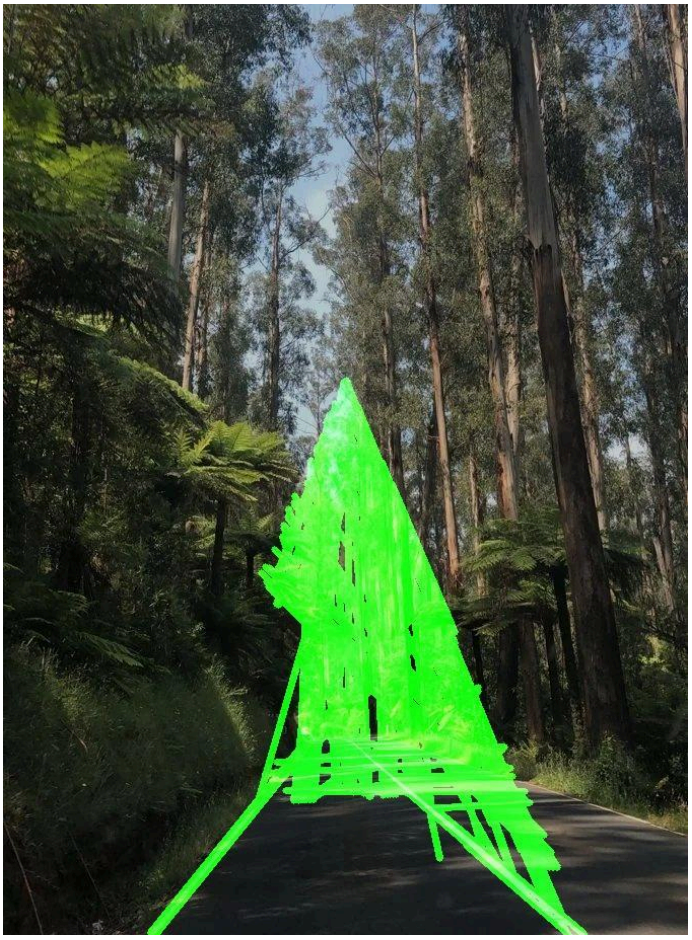

Figure 2: Lane Detection on Forest Road Environment

Figure 3: Lane Detection on Urban Indian Road

RESULTS AND DISCUSSION

The system successfully detects lane lines in varied environments, such as highways, city roads, and rural paths. The results demonstrate the effectiveness of classical computer vision techniques when applied correctly.

In favorable conditions, the system accurately detects lane boundaries using Canny edge detection and Hough Transform. However, challenges arise under poor lighting, faded lane markings, or heavy shadows. These can be mitigated with adaptive thresholding or machine learning-based enhancements.

Below is an additional conceptual image illustrating the output overlay on a sample input frame.

Figure 4: Conceptual Output of Lane Detection (Green Lines Indicating Lane Boundaries)

CONCLUSION AND FUTURE SCOPE

The project successfully demonstrates the use of OpenCV in detecting lane lines from road images. It highlights the importance of preprocessing techniques and robust line detection algorithms. The achieved results show the potential for implementing these systems in real-time driving scenarios.

This work can be expanded by integrating AI-based algorithms like deep learning segmentation models to handle complex environments, curved roads, and multiple lane markings. Future systems could incorporate sensor fusion, real-time tracking, and weather adaptability to improve robustness.

Thus, the project serves as a foundational step towards the development of autonomous driving and advanced driver-assistance systems (ADAS).

REFERENCES

[1] OpenCV Documentation: https://docs.opencv.org
[2] Python Official Docs: https://docs.python.org
[3] NumPy Documentation: https://numpy.org
[4] Research papers on lane detection and self-driving systems.
[5] Lecture notes and resources from AI course material.

[6]Githublink to run program properly:https://github.com/ray33664/Lane-Line-Detection.git