A Real Time Research Projects Report

on

ROAD LANE DETECTION

Submitted by,

KANDULA VARSHA	22J41A0527
GADE USHA	22J41A0517
PULIMANTI SANKEERTHANA	22J41A0546
ANDE AKANKSHA	23J45A0503

in partial fulfilment of the academic requirements II BTech.
of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Under the Guidance of

Mr. K. SRIKANTH

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MALLA REDDY ENGINEERING COLLEGE(A)
Maisammaguda, Secunderabad, Telangana, India 500100
July -2024

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

<u>BONAFIDE CERTIFICATE</u>

This is to certify that this **Real Time Research Project** work entitled "**ROAD LANE DETECTION**", submitted by **KANDULA VARSHA (22J41A0527), GADE USHA (22J41A0517), PULIMANTI SANKEERTHANA (22J41A0546) ,ANDE AKANKSHA(23J45A0503)** to Malla Reddy Engineering College affiliated to **JNTUH, Hyderabad** in academic requirements II BTech for the award of **Bachelor of Technology** in **Computer Science and Engineering** is a bonafide record of project work carried out under my supervision during the academic year 2023– 2024 and that this work has not been submitted elsewhere for a degree.

SIGNATURE	**SIGNATURE**
Mr. K.SRIKANTH	**Dr. P. SRIRAMA CHANDRAMURTHY**
Assistant Professor	**Head of Department**
Department of CSE	Department of CSE

Malla Reddy Engineering College Malla Reddy Engineering College
Secunderabad, 500100

Submitted for Real Time Research Project viva-voce examination held on ______

INTERNAL EXAMINAR **EXTERNAL EXAMINAR**

ABSTRACT

Road lane line detection is essential for autonomous vehicles to navigate safely and avoidcollisions. In this project, we propose a simple approach to road lane detection using OpenCV, specifically designed to process pre-recorded videos rather than live video inputs. Our method involves several key steps, including colour selection, region of interest (ROI) selection, Gaussian blur, Canny edge detection, and Hough line transformation. By combining these techniques, we can effectively detect and visualize road lanes in videos. The detected lane boundaries are overlaid on the original image or displayed as visual cues to the driver. Overall, the real-time road lane detection project using OpenCV offers a practical solution for enhancing road safety and enabling intelligent transportation systems through reliable and efficient lane detection capabilities. Our approach provides a foundational understanding of lane detection concepts and can serve as a starting point for more advanced implementations.

TABLE OF CONTENTS

CHAPTER 1 :INTRODUCTION

Lane line detection is critical for self-driving cars, ensuring they maintain their designated lanes and avoid accidents. The machine learning project will leverage the OpenCVlibrary to achieve real-time lane line detection. Detection will involve identifying white lane markings on both sides of the lane, employing computer vision algorithms. Techniques such as image preprocessing, edge detection, and line detection algorithms will be utilized for accurate lane detection. Real-time processing capabilities will enable continuous monitoring and adjustment to changing road conditions, ensuring reliable performance in diverse environments.

OBJECTIVES :

The objectives of this project are:

- To gain a comprehensive understanding of OpenCV and its capabilities in image processing.
- To develop proficiency in image processing techniques, focusing on their application in road lane detection.
- To design and implement a robust algorithm using OpenCV that accurately detects lane lines under various conditions.
- To enhance the system's robustness, ensuring reliable performance in the presence of shadows, obstacles, and changes in the road environment.
- To implement classification of different lane types (e.g., left, right, dashed, solid) to aid in more nuanced lane detection.
- To integrate the lane detection system with vehicle systems and provide detailed, user friendly documentation to facilitate implementation and usage.

CHAPTER 2 :LITERATURE SURVEY

The literature on lane detection systems has evolved significantly over the years, driven by the growing demand for Advanced Driver Assistance Systems (ADAS) and autonomous vehicles. This survey reviews notable studies, highlighting key methodologies, advancements,and gaps that our project aims to address.

Traditional Lane Detection Techniques:

Early lane detection methods primarily focused on basic image processing techniques. For instance, Canny edge detection and Hough Transform were widely employed to detect straight lane lines. These methods were computationally efficient and performed well in ideal conditions but struggled with curved lanes and variable lighting conditions.

1. Canny, J. (1986). A Computational Approach to Edge Detection. This seminal work laid the foundation for edge detection techniques in computer vision, introducing the widely used Canny edge detector.
2. Duda, R. O., & Hart, P. E. (1972). Use of the Hough Transformation to Detect Lines and Curves in Pictures. This study introduced the Hough Transform, a crucial technique for line detection in images, including lane detection.

3. Kim, Z. (2008). Robust Lane Detection and Tracking in Challenging Scenarios. This research demonstrated the use of Support Vector Machines (SVM) for lane detection, showcasing improved performance in diverse road conditions.
4. Aly, M. (2008). Real-Time Detection of Lane Markers in Urban Streets. This study utilized machine learning techniques to detect lane markers in urban settings, addressing challenges like occlusions and road markings.

5. Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., & Murphy, K. (2015). An Empirical Evaluation of Deep Learning on Highway Driving. This paper presented a deep learning model for lane detection, achieving state-of-the-art performance on highway driving datasets.
6. Neven, D., De Brabandere, B., Proesmans, M., & Van Gool, L. (2018). Towards End-to-End Lane Detection: An Instance Segmentation Approach. This study proposed an end-to-end lane detection model using instance segmentation, highlighting the potential of deep learning in this domain.

7. Hillel, A. B., Lerner, R., Levi, D., & Raz, G. (2014). Recent Progress in Road and Lane Detection: A Survey. This survey reviewed traditional and machine learning- based lane detection techniques, providing a detailed comparison of their performance.
8. Yenikaya, D., Yildirim, M., & Genc, V. (2013). Keeping the Vehicle on the Road: A Survey on On-Road Lane Detection Systems. This comprehensive review focused on on-road lane detection systems, summarizing various approaches and their applications.

CHAPTER 3:SYSTEM ANALYSIS

Lane detection is a crucial aspect of autonomous vehicles, driver assistance systems, and traffic monitoring systems. The ability to detect and track road lanes accurately is essential forsafe and efficient navigation. In recent years, there has been significant progress in road and lane detection, with various techniques being proposed and implemented. In this project, we focus on developing a simple road lane detection system using OpenCV, a popular computer vision library. The system is designed to detect road lanes in pre-recorded videos or images, and does not require real-time processing. The system uses a combination of image processing techniques, such as filtering, colour masking, edge detection, and Hough transforms, to detect road lanes.

EXISTING SYSTEM

In recent years, the development of various lane detection algorithms has significantly advanced the capabilities of Advanced Driver Assistance Systems (ADAS), which are crucial for improving driving safety and driver convenience. These algorithms primarily focus on detecting straight lanes, employing methods like the Hough Transform, Canny Edge Detection, and machine learning techniques to achieve high accuracy in well-structured road environments. These straight lane detection methods are relatively straightforward and computationally efficient, making them suitable for real-time applications in ADAS. However, the detection of curved lanes presents a more complex challenge. Curved lanes can vary significantly in shape, and their appearance can be influenced by various factors such as road geometry, vehicle speed, and environmental conditions like lighting and weather. Traditional lane detection algorithms often struggle with these complexities, leading to reduced accuracy and reliability in real-world driving scenarios where roads are rarely perfectly straight.

DISADVANTAGES OF EXISTING SYSTEM

- ➢ Focus on Straight Lanes: The majority of existing techniques are designed for straight lane detection, often ignoring curved lanes, which results in poor performance on roads with curves.
- ➢ Lack of Improvements in Hough Transform: Improvements and optimizations in the Hough Transform method have not been thoroughly explored, which could enhance lane detection accuracy.

PROPOSED SYSTEM

In the field of Advanced Driver Assistance Systems (ADAS), lane detection is crucial for safe navigation. However, current models primarily focus on straight lanes, neglecting the detection of curved lanes. Additionally, improvements in the Hough Transform for enhanced lane detection and the impact of unfavourable weather conditions like fog have been overlooked. To address these gaps, we propose a novel model using OpenCV for lane detection. The proposed model aims to bridge the gaps in current lane detection systems by effectively handling both straight and curved lanes, improving accuracy with advanced HoughTransform techniques, and enhancing robustness against environmental conditions, all while maintaining real-time performance and scalability.

PROPOSED SYSTEM ADVANTAGES

- Effective Detection of Both Straight and Curved Lanes: The proposed model uses advanced algorithms to accurately detect both straight and curved lanes, improving adaptability to various road types.
- Enhanced Hough Transform Techniques: By incorporating improved Hough Transform methods, the model increases the precision of lane detection, ensuring more reliable performance.

CHAPTER 4:SYSTEM DESIGN

The purpose of this system design is to outline the architecture and functionality of a simple road lane detection system using OpenCV. This system aims to detect and represent road lanes from input images or video streams, serving as a fundamental component for various applications such as autonomous driving, traffic monitoring, and lane departure warning systems. The road lane detection system comprises several interconnected modules designed to process input data, detect road lanes, and generate appropriate output representations. The key components include:

- Input Module: Responsible for acquiring input data, which can be images or video frames captured by a camera mounted on a vehicle or a stationary camera.

- Preprocessing Module: Handles preprocessing tasks such as image resizing, colour space conversion, noise reduction, and edge detection to prepare the input data for lane detection.

- Lane Detection Module: Implements algorithms for detecting road lanes frompre processed images using techniques like Hough transform or polynomial curve fitting.

- Output Module: Generates output representations of detected road lanes, which may include annotated images, video streams with overlayed lanes, or data files containing lane parameters.

The designed road lane detection system using OpenCV offers a comprehensive solution for accurately detecting road lanes from input images or video streams. By integrating various modules for input processing, lane detection, and output representation, the system can be applied to different scenarios with flexibility and reliability. Further optimizations and enhancements can be explored to improve system performance and robustness in real-world applications.

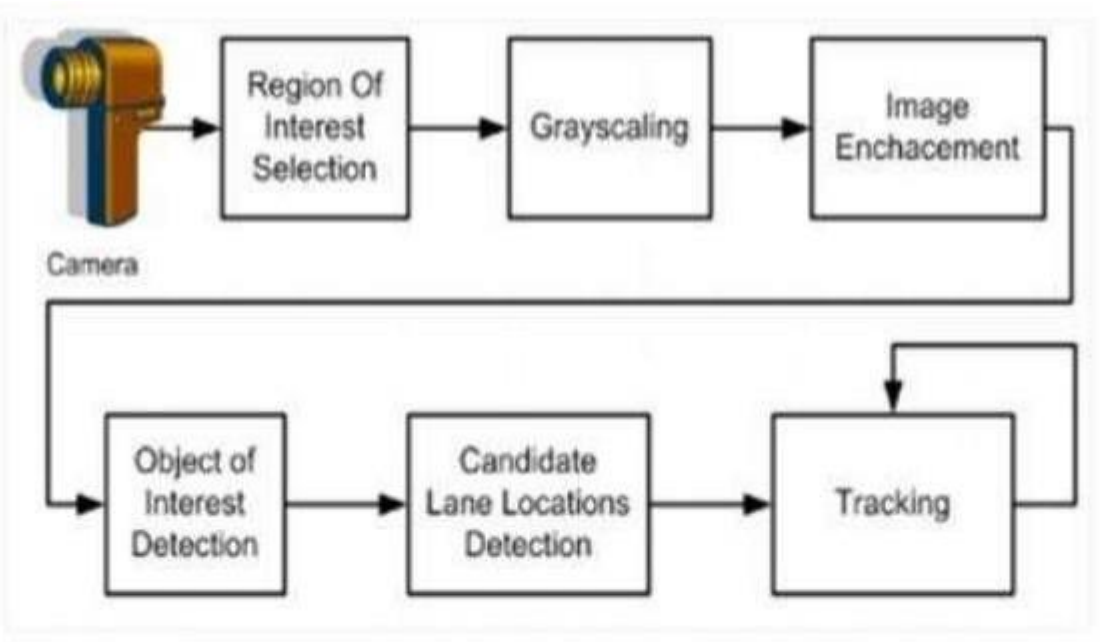
Detailed System Design:

- Input Module:
  - Utilizes OpenCV's Video Capture module to read video frames from a camera or input video file.
  - Converts input frames to the appropriate colour space (e.g., RGB to grayscale) for further processing.
- Preprocessing Module:

- Applies Gaussian blurring to reduce noise and smoothen images.
- Utilizes Canny edge detection to extract edges from pre-processed images.

- Lane Detection Module:
  - Implements Hough transform algorithm to detect straight lines representing road lanes in the edge-detected image.
  - Utilizes region of interest (ROI) masking to focus lane detection on relevant areas of the image.
- Output Module:
  - Draws detected lane lines onto original input frames to create annotated images.
  - Generates video streams with overlayed lane lines for visual representation.

System Architechure – Data Base Design


Camera
Region Of Interest Selection
Grayscaling
Image Enchacement
Object of Interest Detection
Candidate Lane Locations Detection
Tracking

ARCHITECTURE FOR ROAD LANE DETECTION

The project flow will have the following activities:

- Input Image: The process begins with an input image.
- Check if Image is Coloured: Determine whether the input image is coloured.
- Convert to Gray: If the image is coloured, convert it to grayscale.
- Apply Filtering: Apply filtering techniques to the grayscale image to remove noise and smoothen the image.
- Apply Canny Edge Detector: Use the Canny edge detection algorithm to detect edges in the filtered grayscale image.
- Apply Hough Transform: Apply the Hough Transform technique to detect lines in the image from the edges identified.
- Detect and Colour Lanes: Detect the lanes based on the lines identified by the Hough Transform and colour them for visual representation.

UNIFIED MODELING LANGUAGE (UML):

UML (Unified Modelling Language) diagrams are a standardized way to visualize and describe the design of a system. They provide a graphical representation of the system's components, relationships, and behaviour. In this section, we will elaborate on the UML diagrams used to design the road lane detection system.

GOALS OF UML:

The Primary goals in the design of the UML are as follows:

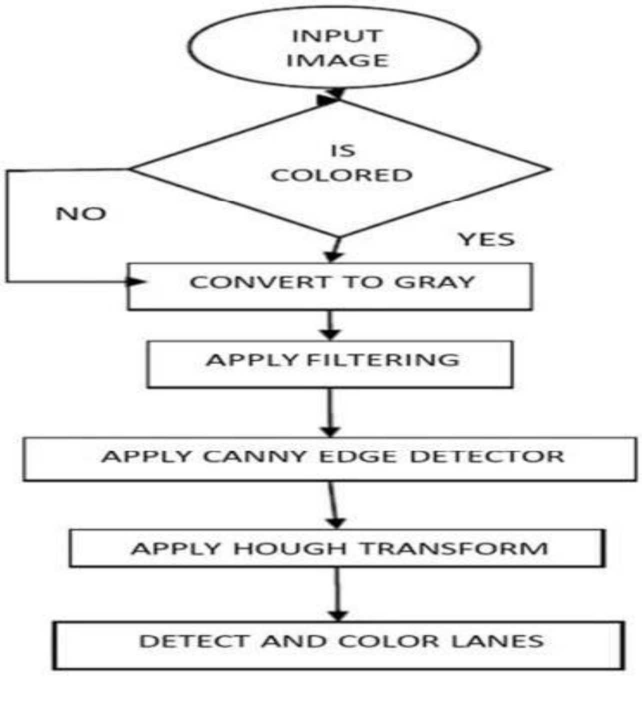
1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.

2. Provide extensibility and specialization mechanisms to extend the core concepts.

3. Be independent of particular programming languages and development process.

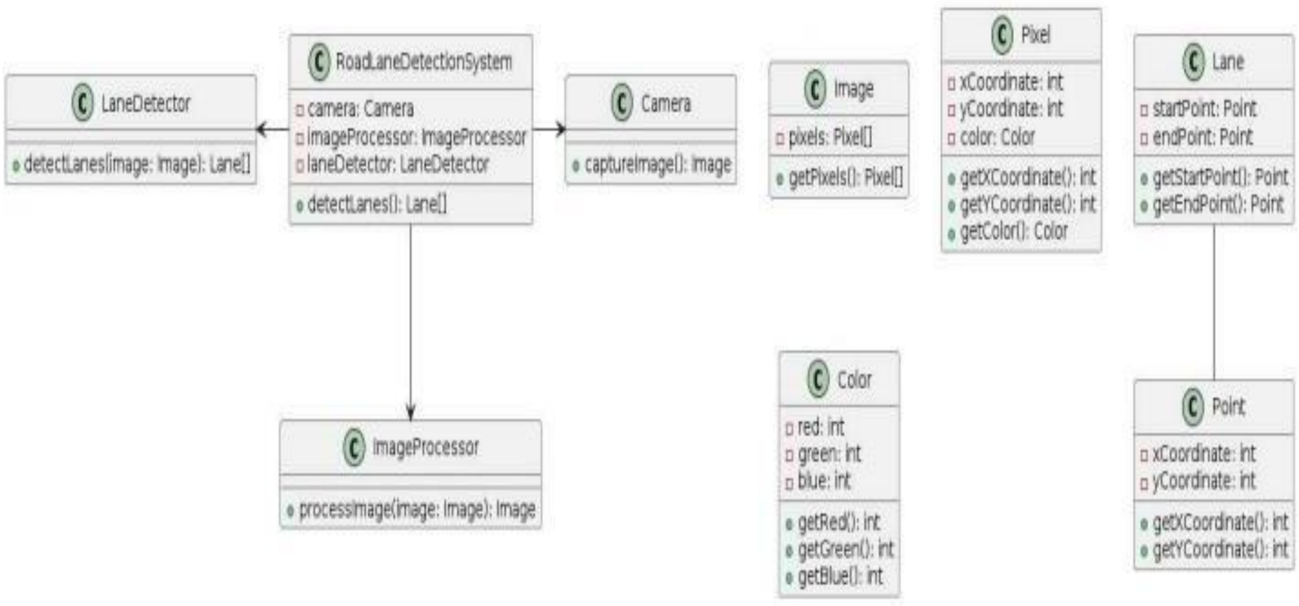
4. Provide a formal basis for understanding the modelling language.

USE CASE DIAGRAM:



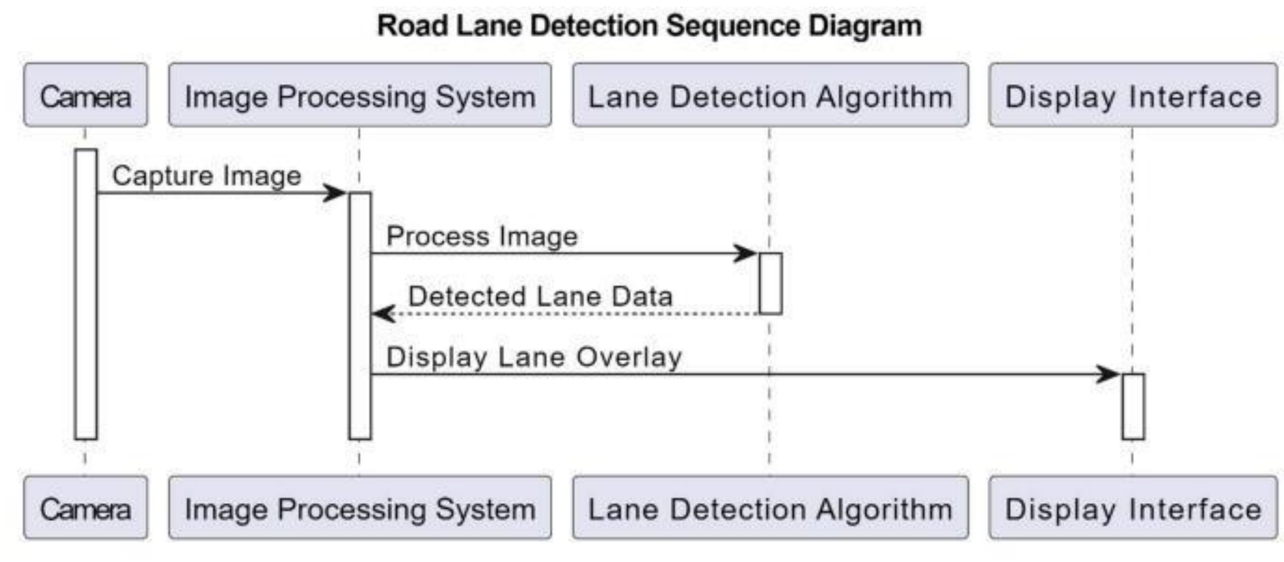
The provided use case diagram outlines a systematic approach for a road lane detection system. It begins with an input image, which is then assessed to determine if it is coloured or grayscale. If coloured, the image undergoes conversion to grayscale for further processing. Next, the grayscale image is subjected to filtering techniques to eliminate noise, followed by the application of the Canny edge detection algorithm to identify edges. Subsequently, the Hough Transform technique is employed to detect lines from the edges detected earlier. Finally, based on these identified lines, the system detects and colours the lanes for visual representation. This sequential flow ensures that each step contributes to the accurate detection and representation of road lanes, thereby enhancing the system's effectiveness in assisting drivers and analysing road conditions.

CLASS DIAGRAM:



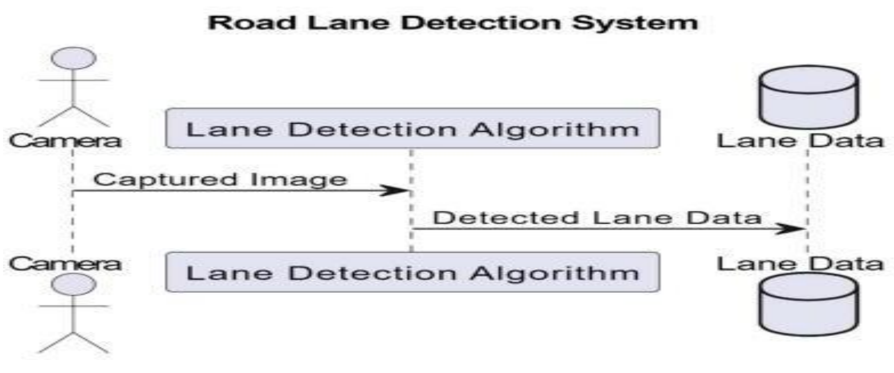
The class diagram illustrates a simplified model of a road lane detection system, a crucial component of autonomous vehicles and driver assistance systems. At the heart of the system lies the `RoadLaneDetectionSystem`, coordinating three main components: the `Camera`, responsible for capturing images of the road environment; the `ImageProcessor`, which analyses captured images to enhance quality and extract relevant features; and the `LaneDetector`, tasked with identifying lane markings from processed images. The `Image` class encapsulates pixel data, while `Pixel` represents individual image pixels with their coordinates and colour information. `Colour` further specifies the RGB components of each pixel. Detected lanes are represented by the `Lane` class, defined by their start and end points, which are instances of the `Point` class containing x and y coordinates. This diagram demonstrates the flow of data and processing steps involved in identifying road lanes, facilitating safer and more accurate navigation for vehicles.

SEQUENCE DIAGRAM:



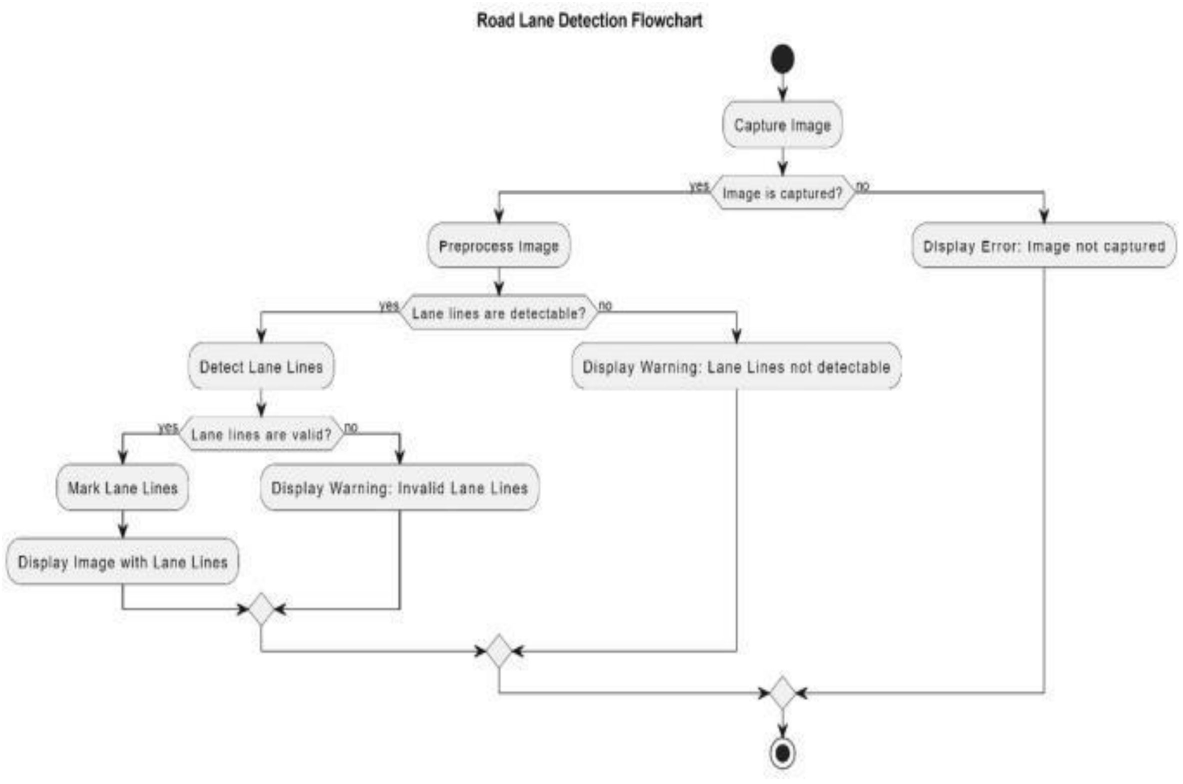
The sequence diagram illustrates the workflow of a road lane detection system. It begins with the camera capturing an image of the road, which is then processed by the image processing system. The processed image is forwarded to the lane detection algorithm, where the algorithm identifies the lane boundaries. Once the lanes are detected, the information is sent back to the image processing system, which overlays the detected lanes onto the original image. Finally, the processed image, now with lane overlays, is displayed on the interface for visualization. This sequence demonstrates the step-by-step process of how an image captured by a camera undergoes analysis and enhancement to highlight road lanes for better visibility and navigation assistance.

DATA FLOW DIAGRAM:



The data flow diagram illustrates a basic system for road lane detection. In this setup, a camera serves as the input device, capturing images of the road. These images are then processed by a lane detection algorithm, depicted as the intermediary component. Upon analysis, the algorithm extracts lane data from the captured images, identifying lane boundaries and markings. Finally, the detected lane data is stored in a database for further utilization or analysis. This system aims to facilitate the automatic identification and tracking of lanes on roads, a critical component in various applications such as autonomous driving systems, traffic monitoring, and road safety management.

FLOW CHART DIAGRAM:


Road Lane Detection Flowchart
Capture Image
Image is captured?
yes
no
Preprocess Image
Display Error: Image not captured
Lane lines are detectable?
yes
no
Detect Lane Lines
Display Warning: Lane Lines not detectable
Lane lines are valid?
yes
no
Mark Lane Lines
Display Warning: Invalid Lane Lines
Display Image with Lane Lines

The provided flowchart illustrates the sequential steps involved in a simple road lane detection process. It begins by capturing an image of the road environment. Once the image iscaptured, it undergoes preprocessing to enhance its quality and prepare it for analysis. The system then attempts to detect lane lines within the pre-processed image. If lane lines are successfully detected, the system proceeds to verify their validity, ensuring they accurately represent the lanes on the road. Valid lane lines are then marked on the image, and the final result, displaying the marked lane lines, is presented. However, if lane lines cannot be detected or are deemed invalid, appropriate warnings are displayed to alert the user. This flowchart serves as a visual guide for understanding the logical flow of operations within the road lane detection algorithm.

CHAPTER 5:IMPLEMENTATION

The goal of implementation is to make a code which is easy to read and understand. This is most vital stage in acquiring fruitful or a framework and giving the client certainly that the new programming or the framework is functionalist venues. The source code must be clear such that the debugging testing, modification can easily be done.

MODULES:

We can consider each of these functions as modules or components:

- canny: Performs Canny edge detection on an input image.
- region_of_interest: Masks out a region of interest in the image.
- houghLines: Detects lines in the input image using the Hough transform.
- addWeighted: Combines the input image with the detected lines.
- display_lines: Draws detected lines on a black image.
- make_points: Calculates endpoints of lines based on slope and intercept.
- average_slope_intercept: Averages the slope and intercept of detected lines to get a single left and right line.

CHAPTER 6:SYSTEM REQUIREMENTS

Hardware Requirements:

- Processor - i5 or above
- RAM - 4 GB (min)
- Hard Disk - 512 GB
- Key - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse

Software Requirements:

- Operating System: Windows 11.
- Coding Language: Python.
- Text editors: vscode/pycharm/eclipse

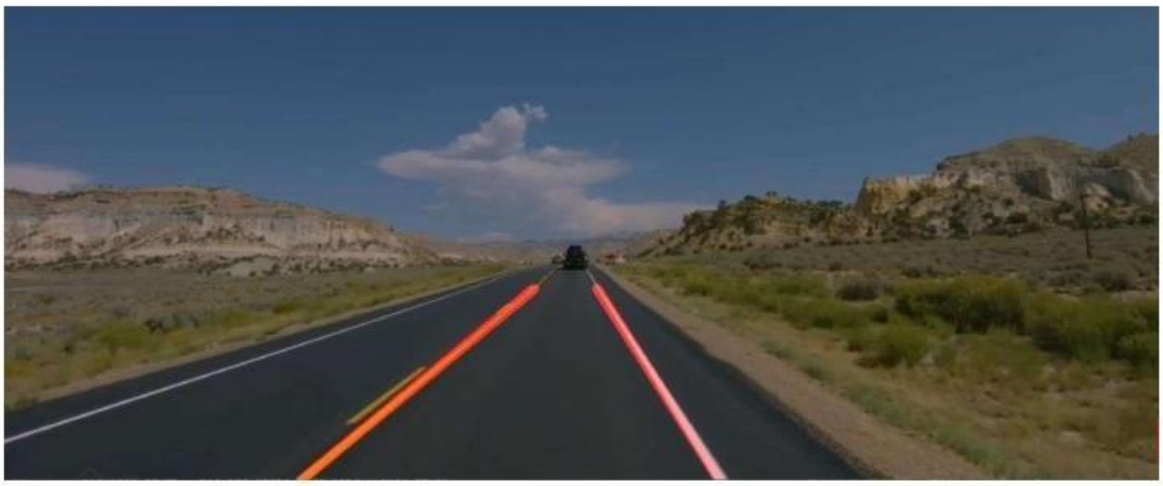
SOURCE CODE :

```
 import cv2

 import numpy as np

 def canny(img):     if

 img is None:

 cap.release()

         cv2.destroyAllWindows()

         exit()

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

     kernel = 5

     blur = cv2.GaussianBlur(gray, (kernel, kernel), 0)

     canny = cv2.Canny(blur, 50, 150)

     return canny

 def region_of_interest(canny):     height =

 canny.shape[0]

     width = canny.shape[1]

     mask = np.zeros_like(canny)

     triangle = np.array([[         (200,

 height),         (800, 350),

         (1200, height),]], np.int32)

     cv2.fillPoly(mask, [triangle], 255)

     masked_image = cv2.bitwise_and(canny, mask)

     return masked_image

 def houghLines(cropped_canny):

     return cv2.HoughLinesP(cropped_canny, 2, np.pi / 180, 100,
np.array([]), minLineLength=40, maxLineGap=5)
def addWeighted(frame, line_image):
    return cv2.addWeighted(frame, 0.8, line_image, 1, 1)
def display_lines(img, lines):     line_image =

np.zeros_like(img)

    if lines is not None:  for line in lines:

                    for x1, y1, x2, y2 in line:

                cv2.line(line_image, (x1, y1), (x2, y2), (0, 0, 255), 10)
    return line_image
def make_points(image, line):
    slope, intercept = line
    y1 = int(image.shape[0])
    y2 = int(y1 * 3.0 / 5)
    x1 = int((y1 - intercept) / slope)
    x2 = int((y2 - intercept) / slope)
    return [[x1, y1, x2, y2]]
def average_slope_intercept(image, lines):
    left_fit = []
    right_fit = []
    if lines is None:        return

None

    for line in lines:
        for x1, y1, x2, y2 in line:
            fit = np.polyfit((x1, x2), (y1, y2), 1)
            slope = fit[0]
            intercept = fit[1]
            if slope < 0:
```

```
        left_fit.append((slope, intercept))
            else:
                right_fit.append((slope, intercept))     left_fit_average = np.average(left_fit, axis=0) if
left_fit else None
    right_fit_average = np.average(right_fit, axis=0) if right_fit else None
    left_line = make_points(image, left_fit_average) if left_fit_average is not None else None

    right_line = make_points(image, right_fit_average) if right_fit_average is not None else None

    averaged_lines = []
    if left_line is not None:
        averaged_lines.append(left_line)
    if right_line is not None:         averaged_lines.append(right_line)

    return averaged_lines
cap = cv2.VideoCapture("straight.mp4")
while(cap.isOpened()):     _,

frame = cap.read()

    canny_image = canny(frame)
    cropped_canny = region_of_interest(canny_image)
    lines = houghLines(cropped_canny)
    averaged_lines = average_slope_intercept(frame, lines)
    line_image = display_lines(frame, averaged_lines)
    combo_image = addWeighted(frame, line_image)
    cv2.imshow("result", combo_image)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

CHAPTER 7 :EXPECTED OUTPUT

Screenshot 7.3.1: Input Image



Screenshot 7.3.2: Output Image

CHAPTER 8:CONCLUSION

The Road Lane Detection project using OpenCV has been an extensive and enlightening endeavour, encompassing various stages of development and testing to ensure the creation of a robust and reliable system. Throughout the project, we have leveraged the powerful image processing capabilities of OpenCV to develop a system capable of accurately detecting road lanes in real-time, a fundamental component of advanced driver-assistance systems (ADAS). The initial stages of the project involved understanding the problem domain and defining the system requirements. We explored the principles of image processing, edge detection, and Hough Transform, which are essential for effective lane detection. The implementation phase focused on developing core modules, including image preprocessing, edge detection, region of interest selection, and line detection. Each module was meticulously tested through unit testing to ensure its functionality and correctness.

In conclusion, the Road Lane Detection system developed using OpenCV is a testament to the effectiveness of combining theoretical knowledge with practical application. The project not only demonstrates the feasibility of implementing lane detection systems using open-source tools but also provides a solid foundation for future enhancements and integration with more advanced ADAS technologies. This project has equipped us with valuable insights and skills in image processing, system testing, and software development, laying the groundwork for further exploration and innovation in the field of autonomous driving and intelligent transportation systems.

CHAPTER 9:REFERENCES

1. Shopa, P., N. Sumetha and P.S.K Pathra. “Traffic sign detection and recognition using OpenCV", International Conference on Information Communication and Embedded Systems (ICICES2014), 2014.

2. “Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixelwise labelling,” arXiv preprint arXiv:1505.07293, 2015.

3. J. Long, E. Shelhamer, and T. Darrell, “LANE DETECTION TECHNIQUES” – A Review.” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

4. M.Cordts,M. Omran,S.Ramos, T.Rehfeld,M. Enzweiler, R.Benenson,U. Franke,S.Roth, and B.Schiele, “The cityscapes dataset for semantic urban scene understanding,” in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

5. S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, And P.H Torr,

“A Layered Approach To Robust Lane Detection At Night.” 2015, pp. 1529–1537.