**REAL TIME ROAD LANE DETECTION**

A Mini Project Report

Submitted in partial fulfillment of the requirements for the award of the degree of

## Bachelor of Engineering

in

## Computer Science and Engineering

By

## PATHLAVATH DIVYA 2456-21-733-046

**GUNDUPALLI THANUSHA 2456-21-733-020**

**GADIRAJU SRAVYA SANTHOSHI 2456-21-733-016**

## C:\Users\Divya P\Downloads\WhatsApp Image 2024-07-12 at 13.40.39_f5ebd91a.jpgWhatsApp Image 2024-07-12 at 13.40.39_f5ebd91aUnder the guidance of Mrs. N. MEGHANA

## Assistant Professor

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## Gokaraju Lailavathi Women’s Engineering College

**(Affiliated to Osmania University)**

## GRIET Campus , Bachupally , Kukatpally , Hyderabad-500090 (2023-2024)

**Department of Computer Science and Engineering**

**Gokaraju Lailavathi Women’s Engineering College**

(Affiliated to Osmania University)

GRIET Campus, Bachupally, Kukatpally, Hyderabad-500090(2023-2024)



CERTIFICATE

This is to Certify that A Mini Project report entitled  **“ Real Time Road Lane Detection ”** is being submitted by Pathlavath Divya **(**2456-21-733-046**),** Gundupalli Thanusha **(**2456-21-733-020**),** Gadiraju Sravya Santhoshi (2456-21-733-016) in partial fulfillment of the requirement of the award for the degree of Bachelor of Engineering in “Computer Science and Engineering” O.U., Hyderabad during the year 2023-2024 is a record of bonafide work carried out by them under my guidance. The results presented in this project have been verified and are found to be satisfactory.

## Project Guide H.O.D

**Mrs. N. Meghana Dr. Padmalaya Nayak**

Assistant Professor, Professor & Head, Dept of CSE Dept. of CSE

## External Examiner(s)



**Department of Computer Science and Engineering**

**Gokaraju Lailavathi Women’s Engineering College**

(Affiliated to Osmania University)

GRIET Campus, Bachupally,

Kukatpally, Hyderabad-

500090(2023-2024)

**DECLARATION**

We, Pathlavath Divya bearing Ht.No.2456-21-733-046, Gundupalli Thanusha bearing Ht.No.2456-21-733-020 and Gadiraju Sravya Santhoshi bearing Ht.No.2456-21-733-016 hereby certify that the minor project entitled “**REAL TIME ROAD LANE DETECTION** ” is submitted in the partial fulfilment of the required for the award of the degree of Bachelor of Engineering in Computer Science and Engineering.

This is a record work carried out by us under the guidance of Mrs. N. Meghana, Assistant Professor, CSE, Gokaraju Lailavathi Women’s Engineering College, Hyderabad. The results embodied in this report have not been reproduced/copied from any source. The results embodied in this report have not been submitted to any other university or institute for the award of any other degree or diploma.

**ACKNOWLEDGEMENT**

It is our privilege and pleasure to express my profound sense of respect, gratitude and indebtedness to our guide Mrs. N Meghana, Assistant Professor, Department of Computer Science and Engineering, Gokaraju Lailavathi Women’s Engineering College, for her inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout this dissertation work.

We express our sincere thanks to Project Coordinator Mrs.K.Pranathi, Assistant Professor and Mrs.M.Archana, Assistant Professor, Department of Computer Science and Engineering, Gokaraju Lailavathi Women’s Engineering College, for their valuable suggestions and constant helping in completing the work.

We express our sincere gratitude to Dr. Padmalaya Nayak, Head of the Department of Computer Science and Engineering, Gokaraju Lailavathi Women’s Engineering College, for her precious suggestions, motivation and co-operation.

We express our sincere thanks to Dr. Akund Sai Hanuman, Principal, Gokaraju Lailavathi Women’s Engineering College, Hyderabad for his encouragement and constant help.

We extend our sincere thanks to all the teaching staff of Computer Science and Engineering Department for their support and encouragement.

Last but not least, we wish to acknowledge my friends and family members for giving moral strength moral strength and helping us to complete this dissertation.

# ABSTRACT

The Lane driving assistance is the need of the hour as the number of people using vehicles has increased. This may increase the chances of accidents. If we consider driving at night time, smart assistance for vehicles is even more important. We have worked on the lane detection, where we have implemented our algorithm for four different scenarios. We have used Adaptive Gamma Correction on dark frames and Inverse Perspective Mapping (IPM) is used with camera calibration and Kalman Filter (KF) plays an important role in predicting the lanes in the dark environment. The experiment conducted shows that our algorithm is robust to use in real-time night conditions.

This project aims to develop a system for real-time lane detection on roads during nighttime conditions. The system will utilize advanced image processing and computer vision techniques to identify and track lane markings in low-light environments. By leveraging algorithms such as edge detection, Hough transforms, and machine learning models, the system will accurately detect lane boundaries and provide real-time feedback to drivers. The project will focus on enhancing road safety by improving lane detection performance in challenging lighting conditions. Through this research, we aim to contribute to the development of intelligent transportation systems that can operate effectively during both day and night.

In conclusion, our project on real time lane detection on roads at night is promising endeavor that aims to

enhance road safety by utilizing image processing and computer vision techniques.

# CONTENTS

Chapter-1 INTRODUCTION 1

* 1. [MOTIVATION… 1](#_TOC_250022)
  2. [PROBLEM STATEMENT… 1](#_TOC_250021)
  3. [PROJECT OBJECTIVE… 2](#_TOC_250020)

[Chapter-2 LITERATURE SURVEY 3](#_TOC_250019)

* 1. [EXISTING SYSTEM… 4](#_TOC_250018)
  2. LIMITATIONS OF EXISITNG SYSTEM… 4

Chapter-3 SOFTWARE REQUIREMENT SPECIFICATION 5

* 1. SOFTWARE REQUIREMENTS… 5
  2. HARDWARE REQUIREMENTS………………………………………………………..5
  3. [FUNCTIONAL REQUIREMENTS 5](#_TOC_250017)
  4. [NON-FUNCTIONAL REQUIREMENTS…......................................................................6](#_TOC_250016)

[Chapter-4 SYSTEM DESIGN 7](#_TOC_250015)

* 1. SYSTEM DESIGN… 7
  2. SYSTEM ARCHITECTURE………………………………………………………….…8
  3. [UML DESIGN… 8](#_TOC_250013)
     1. [ACTIVITY DIAGRAM… 11](#_TOC_250012)
     2. [USECASE DIAGRAM… 1](#_TOC_250011)2
     3. [COMPONENT DIAGRAM… 13](#_TOC_250010)
     4. [SEQUENCE DIAGRAM… 14](#_TOC_250009)
     5. [STATE CHART DIAGRAM… 15](#_TOC_250008)
  4. [TECHNOLOGY DESCRIPTION… 16](#_TOC_250007)

Chapter-5 IMPLEMENTATION 18

* 1. OBJECTIVES AND SCOPE… 18
  2. [MODULES… 2](#_TOC_250006)0
  3. EXECUTABLE CODE… 21

[Chapter-6 TESTING 25](#_TOC_250005)

* 1. TESTING DEFINATION… 25
  2. [UNIT TESTING…..........................................................................................................25](#_TOC_250004)
  3. FUNCTIONAL TESTING……………………………………………………………...26

[Chapter-7 RESULTS 27](#_TOC_250003)

Chapter-8 CONCLUSION 28

* 1. [CONCLUSION… 28](#_TOC_250002)
  2. [FUTURE SCOPE… 2](#_TOC_250001)8

[REFERENCES 2](#_TOC_250000)8

|  |  |  |
| --- | --- | --- |
| **SNO.** | **LIST OF FIGURES** | **PAGE NO.** |
| 4.2 | System Architecture | 8 |
| 4.3.1 | UML Hierarchy Diagrams | 10 |
| 4.3.1.1 | Activity Diagram | 11 |
| 4.3.2.1 | Use Case Diagram | 12 |
| 4.3.3.1 | Component Diagram | 13 |
| 4.3.4.1 | Sequence Diagram | 14 |
| 4.3.5.1 | State Chart Diagram | 15 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table. No** | **List of Tables** | **Page. No** |
| 2.1 | Literature Survey | 5 |
| 7.1 | Upload input | 25 |
| 7.2 | Upload outut | 25 |
|  |  |  |
|  |  |  |

# CHAPTER-1

**INTRODUCTION**

## MOTIVATION

Night driving situation is generally unsafe due to poor visibility and fatigue, in addition to the rising expectations on vehicle safety, the proposed real-time road lane detection system is developed to meet the needs of the specific night driving environment. This solution employs the latest trends in AI, CV and online processing to deliver a simple yet strong platform of improving the performance of LKA and mitigating LDA. Thus, the high performance in various conditions at night allows enhancing the functions of self-driving cars and ADAS systems as well as improving safety and efficiency on the roads.

## PROBLEM STATEMENT

Real-time lane detection on roads at night is a raised task and this is due to poor lighting conditions and high risk of driver fatigue common on such roads, therefore the chances of a lane departure accident is more likely. Today’s lane detection systems are not so perfect and are unable to sustain the degree of reliability and preciseness in these circumstances, which affects safety and operating performance. The requirement for a solution that can respond well and recognises and draws lanes in real-time, especially during the night, is therefore crucial for the improvement of driver safety, self-driving cars and vehicle automation, and improvement of ADAS.

## 

## 1.3 PROJECT OBJECTIVE

The goal of the Late Night Road Detection System project is to create an effective, accurate and fast working system, which would be able to detect and recognize the state of roads at night. This system is designed to improve road safety, minimize probable accidents and optimize flow on roads by means of giving real-time information with reference to existing and possible dangerous conditions on roads. It will give the system’s various algorithms and sensors the ability to precisely detect different types of road conditions including wet, slippery, bumpy and especially with poor or with inadequate lighting during the night or with harsh, unfavourable and inclement weather. It will detect features that cause an obstacle on the road, animals and other items not supposed to be in that area, the results are timely notifications to drivers and other stakeholders concerned to avoid the occurrence of an accident.

Furthermore, the plan intends to gather and estimate the traffic circulation, pavement state, and weather or light conditions during nighttime. The findings from this data will assist in determining areas and times which are most prone to these issues and thus, remedies can be set out and effected. That foilage will be interoperable with current traffic control and survelliance systems in order to help interaction of traffic management and monitoring offices, crisis organizations, and state/local governments. An easily navigable format will be created to give drivers and authorities easy-to-understand and functional information on current road conditions and PRA messages.

The equipment will be designed in order to follow the safety standards and the number of false alarms will be minimized on the maximum of the specificity of the system and the accuracy will be validated by the tests. Another goal is also being able to scale the system to different road networks and settings to modify it as new technologies and methods enter the picture are introduced. In realizing these objectives above, the Late Night Road Detection System has the prospects of increasing road safety especially at night, minimizing the occurrence of accidents, and thereby facilitate better traffic control.

# CHAPTER 2 LITERATURE SURVEY

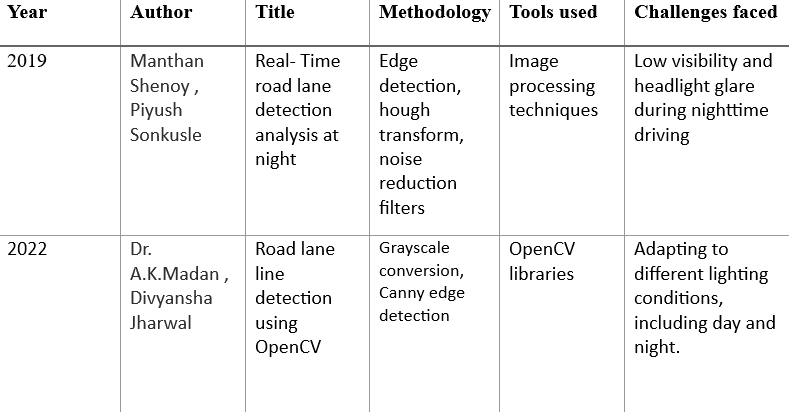
## Real Time Road Lane Detection At Night: This study is a comparative

## analysis done on the subject.

This planned project is generally focused on designing and testing of a real-time road lane detection system which has particularly been designed to work at night. Using enhanced artificial intelligence and especially computer vision algorithms the system is intended to maintain proper detection of the lane markings in poor visibility conditions as well as at night. The work to be done in this study is to evaluate the performance of the propose system with the available lane detection systems and quantify the performance measures such as accuracy, reliability and speed up. The aim is to help increase safety for drivers, facilitate the abilities of self-driving vehicles, and improve ADAS for more safety during night-time driving.

## Cloud-Based Real-Time Road Lane Detection At Night

Real-time road lane detection on clouds at night offers a stylish prospect to the problem of detecting road lanes as a result of low lighting during the night. Expanding on this matter, this approach make use of cloud computing which aids in improving the accuracy of the lanes as well as its processing time because of the availability of powerful AI and computer vision algorithms on remote servers. Real time alerts are produced by the system, greatly increasing the chances of driver’s safety and minimizing the chances of the accident caused by the improper lane departure.



## Existing System

Table- 2.1 Literature Survey

The current solutions of real-time detecting the road lanes at night mainly utilizes the integration of conventional image processing and deep learning models. For instance, edge detection and a method by the name Hough Transform can falter in success in low light or in different lighting. Subsequent strategies employ deep learning, especially CNNs, to enhance the effectiveness and endurance of lane boundaries detection in night conditions.

## Limitations of Existing System

## Reduced accuracy in low-light conditions

## Dependence on well-maintained lane markings

## Adverse weather conditions

## Limited sensor fusion

## Cost and Accessibility

## Complex load scenarios

# CHAPTER-3

## SOFTWARE REQUIREMENT SPECIFICATION

This chapter gives an overview of the software and hardware components required for our project.

* 1. **SOFTWARE REQUIREMENTS** Operating System : Windows 11 Coding Language : Python 3.11
  2. **HARDWARE REQUIREMENTS** System : intel i5

Storage : Sufficient storage

## FUNCTIONAL REQUIREMENTS

Here are some functional requirements for the real-time road lane detection system, specifically for nighttime conditions.

* + - Real-time processing
    - Night time visionPhoto
    - Lane Detection accuracy
    - Robustness to environmental conditions
    - Adaptive to different road types
    - Alerts and feedback
    - Integration with vehicle systems
    - Performance metrics
    - User Interface
    - Documentation and support
    - Scalability

## NON-FUNCTIONAL REQUIREMENTS

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

They basically deal with issues like:

* + - Portability
    - Security
    - Maintainability
    - Reliability
    - Scalability
    - Performance
    - Reusability
    - Integration
    - Compliance
  1. **System Design**

# CHAPTER-4 SYSTEM DESIGN

In this phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

There are two kinds of design documents developed in this phase:

## High-Level Design (HLD)

* + - Brief description and name of each module
    - An outline about the functionality of every module
    - Interface relationship and dependencies between modules
    - Database tables identified along with their key elements
    - Complete architecture diagrams along with technology details

## Low-Level Design(LLD)

* + - Functional logic of the modules
    - Database tables, which include type and size
    - Complete detail of the interface
    - Addresses all types of dependency issues
    - Listing of error messages

## SYSTEM ARCHITECTURE:

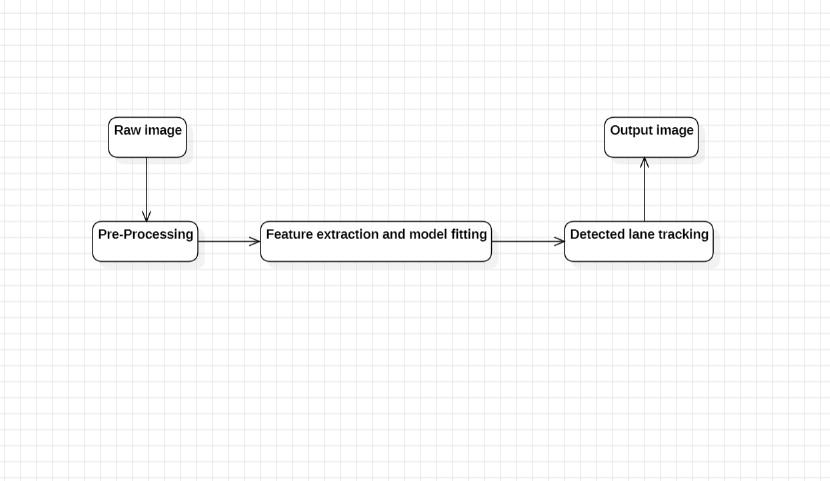
****

Figure-4.2.1 System Architecture

## UML Design:

Unified Modeling Language (UML) is a general purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language; it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system, UML helps software engineers, businessmen and system architects with modeling, design and analysis. The Object Management Group (OMG) adopted Unified Modeling Language as a standard in 1997. It’s been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

**Do we really need UML?**

* Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
* Businessmen do not understand code. So UML becomes essential to communicate with non programmer’s essential requirements, functionalities and processes of the system.
* A lot of time is saved down the line when teams are able to visualize processes, user interactions and static structure of the system.
* UML is linked with object oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

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

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

## Types of UML Diagrams:

**Structural Diagrams:**

Capture static aspects or structure of a system. Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.

## Behavior Diagrams:

Capture dynamic aspects or behavior of the system. Behavior diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

### The image below shows the hierarchy of diagrams according to UML

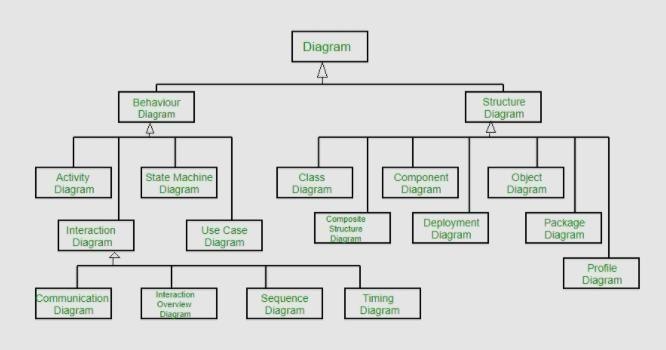


Figure-4.2.1 UML Hierarchy diagrams

## ACTIVITY DIAGRAM:

In software engineering, a Activity diagram in the Unified Modeling Language (UML) is a flow chart to represent the flow from one activity to another activity. The activity can be described as a operation of a system. The control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

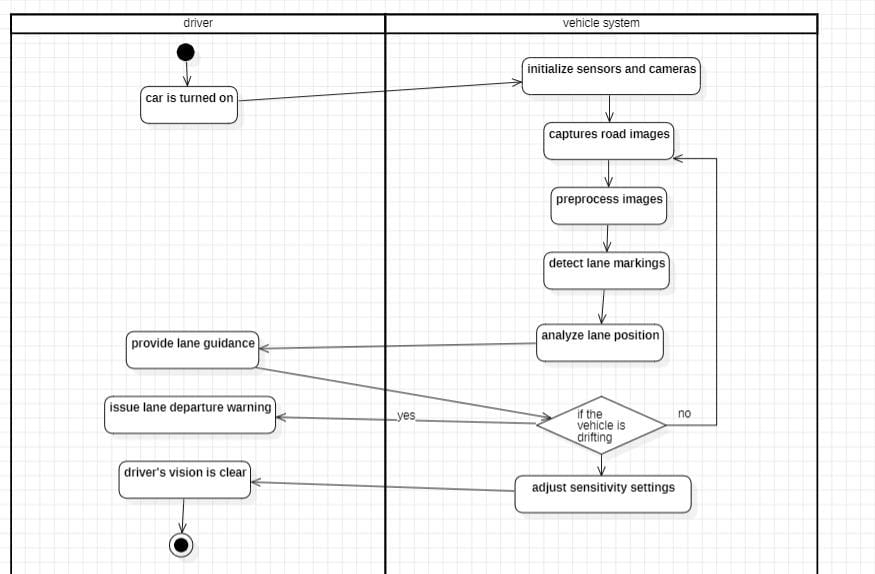


Figure-4.3.1.1 Activity Diagram

## USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral

diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted

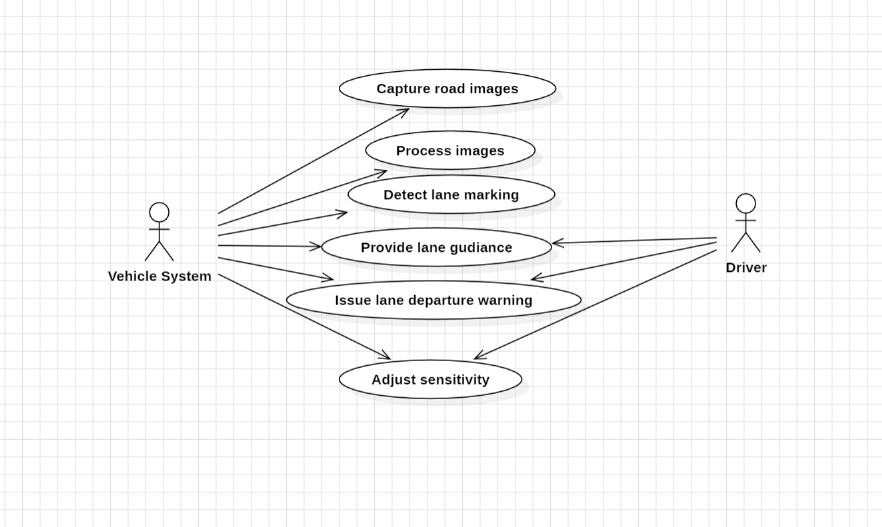


Figure-4.3.2.1 Use Case Diagram

## COMPONENT DIAGRAM:

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.

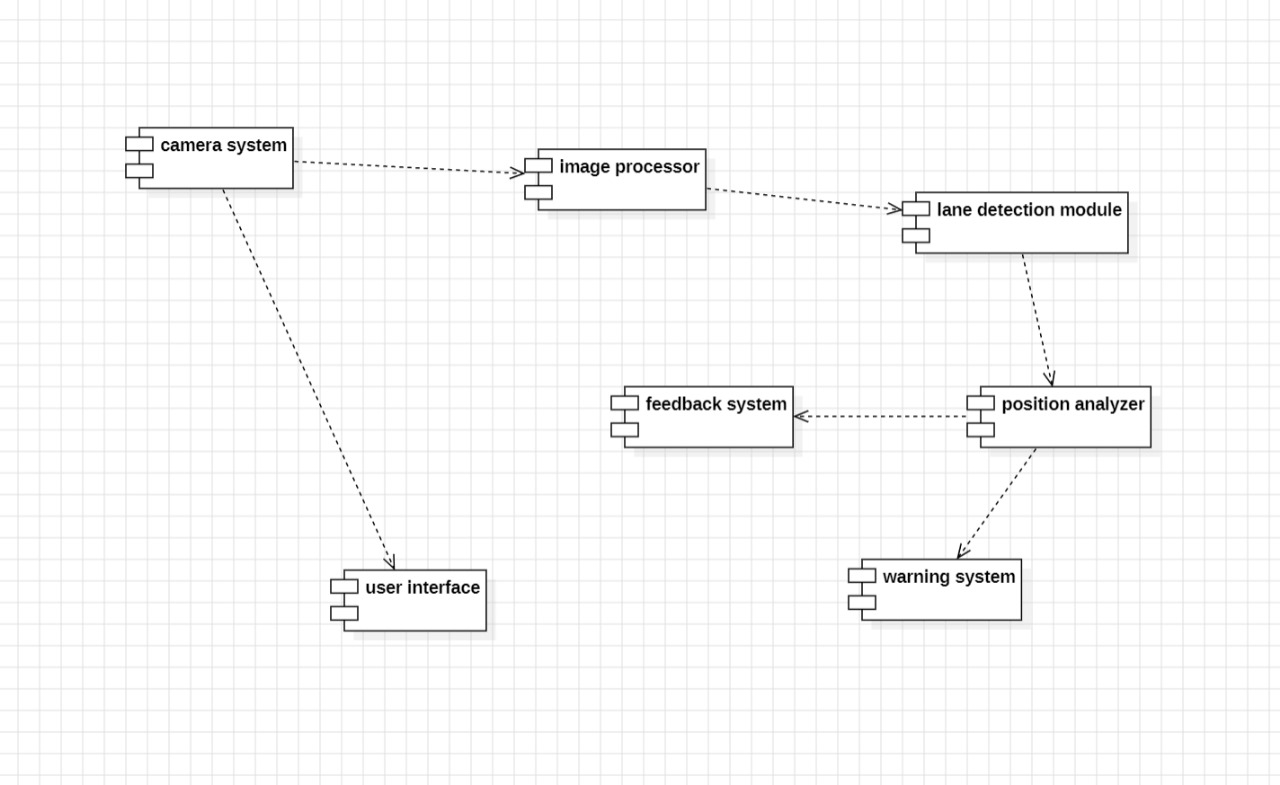


Figure-4.3.3.1 Component Diagram

## SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) diagram that illustrate the sequence of messages between object of an interaction. It consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.

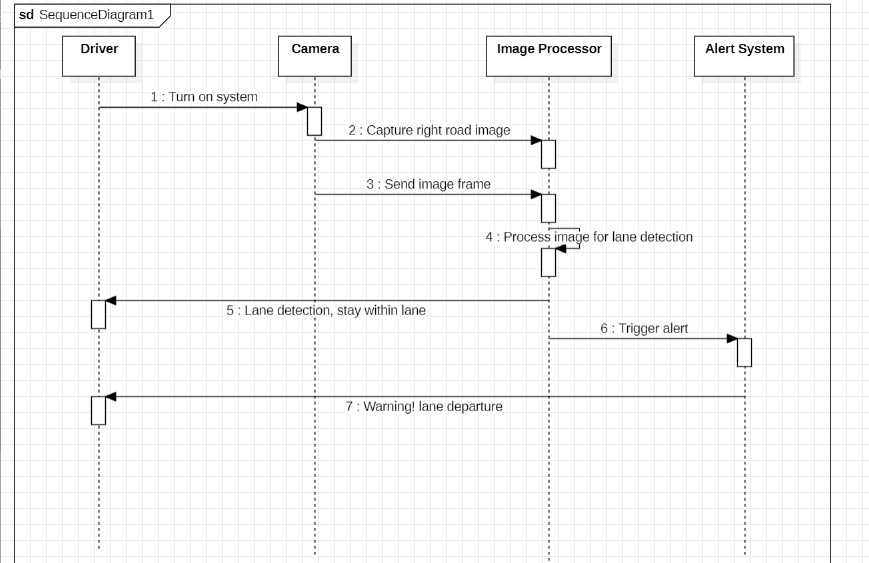


Figure-4.3.4.1 Sequence Diagram

## STATE CHART DIAGRAM:

In UML, an State chart diagram is an illustrate of all the possible behavioral states of a software system. Component may be exhibit and the various state changes it’s predicted to undergo over the course of its operation.

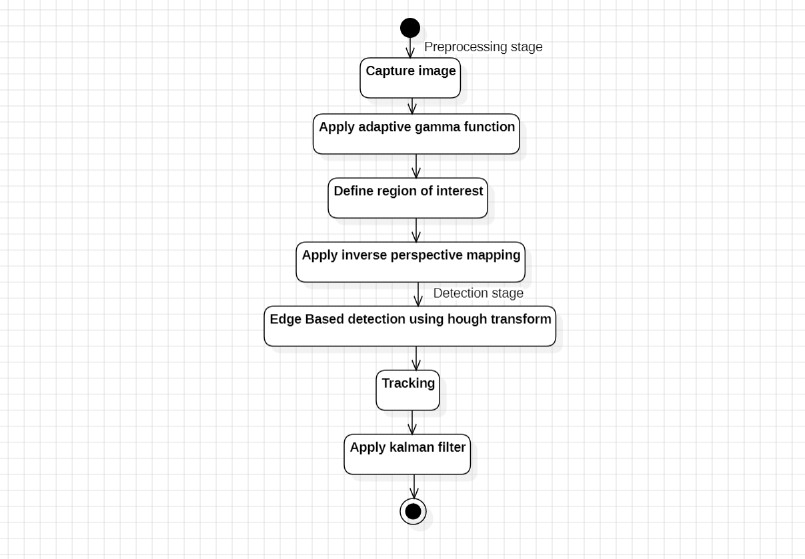


Figure-4.3.5.1 State chart Diagram

## TECHNOLOGY DESCRIPTION

**Numpy:**

Numpy is one of the primary equipments in Python for scientific computing. It offers comprehensive support for really big multi-dimensional matrices and arrays and a set of mathematical operations on such arrays. In this code, numpy is mostly used in array operations where masks are created, slopes and intercepts are to be calculated. The specific standard library used with greater effect here is Numpy, the efficiency of the code is guaranteed by this system, especially when working with large matrices such as frames from a video stream.

## OpenCV:

OpenCV is now an enormously popular computer vision library that offers a set of utilities for the real-time processing of images and videos. In this code, OpenCV is used extensively for several tasks like image masking, edge detection, Hough transform, and drawing lines.

## MoviePy:

MoviePy is a Python library used for video handling and generally, for some of the tasks that can be given to a video cutter, joiner, and processor. In the present work, the MoviePy module is used to load the required input video file and then sequentially traverse through the frames of the video file in the frame\_processor() function. On completion processing, MoviePy generates a new output video file for the process. The integration with MoviePy allows for using the detected lane algorithm on entire videos, process frame by frame in a pipeline.

## Google Colab Integration:

The code snippet is designed to run in the Google Colab which is a cloud based Google own environment to run Jupyter notebook. The upload() function from Google Colab enables the user to upload or bring the video file in the environment containing the Colab so that the before mentioned

processes can be performed on it. In addition, cv2\_imshow, which was made by Google Colab developers in the notebook environment for the time being is to be used, but it is not called in this script.

## Image Processing Techniques:

Several Image processing techniques are implemented in this code:

## Grayscale Conversion:

The frames are changed to black and white for simpler processing since color data will not be used in analysis since it will not be needed.

## Edge Detection with Canny:

Then Canny Edge Detection is used to detect the edges in the frame which might be lane lines.

## Region of Interest masking:

It is a mask that assists in focusing on the area of the image where lanes can be seen, and hence does not react to edges which are very seldom to reveal any lanes’ positions.

## Hough Line Transform:

The Hough Transform is used in order to find the lane markings on a picture after the edges have been detected.

# CHAPTER 5

## IMPLEMENTATION

## Objectives and scope:

Define the objective of the lane detection system. They include lane keeping assistance, the warnings of lane departure, or complete autonomous driving. Perhaps the most significant limitation throughout the planning of the project is understanding the parameters surrounding the project.

## Gather Data Sets:

Some few datasets available for this purpose are LaneNet dataset, CULane that has images with labeled lanes. If existing datasets are not available or not enough, you will have to record your own data with vehicle-mounted cameras.

## Set up Development Environment:

Such projects are often implemented using Python since it has rich libraries and it is relatively easy to use. Make use of OpenCV for handling vision data, TensorFlow, or PyTorch for handling AI-related tasks and NumPy for operations involving numbers.

## Preprocess Data Image Processing:

This comprises scaling of images, scaling the pixel intensities, and data creation to improve the capability of the model. The points to consider while learning are when creating a custom dataset, lanes have to be accurately labeled for training.

## Develop Lane Detection Algorithm:

**Edge Detection:** Some of the approaches include Canny Edge Detection to detect edges in the pictures.

**Hough Transform:** Identification of lines in images by considering the edges of the images.

**Region of Interest(ROI):** Concentration is made on that part of the image where lanes are supposed to be.

**Implement the Detection System:**

**Integration:** Combine the lane detection algorithm with a system that deals with the video that come from the cameras. This many include video processing framework such as the OpenCV for real time video processing.

**Post Processing:** Smoothing and filtering should be carried out in order to get better outcome of lane detection and make the process more stable.

## MODULES:

* + 1. **Image Processing Module:**

The image processing module reduces the noise of the raw image, which is add-on to making the image simpler so as to make further analysis easier. The key functions are grayscale conversion which helps in changing the RGB image into grayscale image so that further process could be easier and more efficient.

## Edge Detection Module:

The edge Detection module identifies where the lane lines are most probable to be in the segmented image. The key functions are Canny Edge Detection which helps in identifying edges in the image, the method finds areas where there is a steep variations of intensity.

* + 1. **Selection of Region of Interest Module:**

The region of interest module reduces the dimensionality of the image reducing the computational time and is mainly centered on the region where lanes are common. The key functions are Masking which puts a mask on the edge-detecting image to get the region of interest only.

* + 1. **Line Detection Module Purpose:**

The line detection module finds possible lane lines in the area that is given as a region of interest. The key functions are Hough Transform which is capable of detecting lines in the edge-detected masked image. Another key function is Line filtering which excludes unwanted or noisy lines due to slope and position of the line in the set.

## Input/Output Module:

The purpose of input/output module is to handle IO related task like uploading of the video file and saving the output that was generated by the function. The key functions are file uploading and file saving where file uploading is an element that enables the user to select a video file which will be uploaded on the website and file saving is the path of the output file in which the processed video is saved.

* 1. **EXECUTABLE CODE**

import numpy as np

import cv2

from google.colab.patches import cv2\_imshow

from moviepy.editor import VideoFileClip

def region\_selection(image):

mask = np.zeros\_like(image)

if len(image.shape) > 2:

channel\_count = image.shape

ignore\_mask\_color = (255,) \* channel\_count

else:

ignore\_mask\_color = 255

rows, cols = image.shape[:2]

bottom\_left = [cols \* 0.1, rows \* 0.95]

top\_left = [cols \* 0.4, rows \* 0.6]

bottom\_right = [cols \* 0.9, rows \* 0.95]

top\_right = [cols \* 0.6, rows \* 0.6]

vertices = np.array([[bottom\_left, top\_left, bottom\_right, top\_right]],dtype=np.int32)

cv2.fillPoly(mask, vertices, ignore\_mask\_color)

masked\_image = cv2.bitwise\_and(image, mask)

return masked\_image

def hough\_transform(image):

rho = 1

theta = np.pi/180

threshold = 20

minLineLength = 20

maxLineGap = 500

return cv2.HoughLinesP(image, rho=rho,theta=theta,threshold=threshold,minLineLength=minLineLength, maxLineGap=maxLineGap)

def average\_slope\_intercept(lines):

left\_lines = []

left\_weights = []

right\_lines = []

right\_weights = []

for line in lines:

for x1,y1,x2,y2 in line:

if x1 == x2:

continue

slope = (y2 – y1)/ (x2 – x1)

intercept = np.sqrt((y2 – y1) \*\* 2 + (x2 – x1) \*\* 2)

if slope < 0:

left\_lines.append((slope, intercept))

left\_weights.append(length)

else:

right\_lines.append((slope. Intercept))

right\_weights.append(length)

left\_lane = np.dot(left\_weights, left\_lines) / np.sum(left\_weights) if len(left\_weights) > 0 else None

right\_lane = np.dot(right\_weights, right\_lines) / np.sum(right\_weights) if len(right\_weights) > 0 else None

return left\_lane, right\_lane

def pixel\_points(y1,y2,line):

if line is None:

return None

slope, intercept = line

x1 = int((y1 – intercept) / slope)

x2 = int((y2 – intercept) / slope)

return ((x1, int(y1)), (x2, int(y2)))

def lane\_lines(image, lines):

left\_lane, right\_lane = average\_slope\_intercept(lines)

y1 = image.shape[0]

y2 = y1 \* 0.6

left\_line = pixel\_points(y1,y2,left\_lane)

right\_lane = pixel\_points(y1,y2,right\_lane)

return left\_line, right\_line

def draw\_lane\_lines(image, lines, color = [255,0,0], thickness = 12):

line\_image = np.zeros\_like(image)

for line in lines:

if line is not None:

cv2.line(line\_image, \*line, color, thickness)

return cv2.addWeighted(image, cv2.COLOR\_BGR2GRAY)

kernel\_size = 5

blur = cv2.GaussianBlur(grayscale, (kernel\_size, kernel\_size), 0)

low\_t = 50

high\_t = 50

edges = cv2.Canny(blur, low\_t, high\_t)

region = region\_selection(edges)

hough = hough\_transform(region)

result = draw\_lane\_lines(image, lane\_lines(image, hough))

return result

def process\_video(test\_video, output\_video):

input\_video = VideoFileClip(test\_video, audio = False)

processed = input\_video.fl\_image(frame\_processor)

processes.write\_videofile(output\_video, audio = False)

#Uploading the file

from google.colab import files

uploaded = files.upload()

#Make sure ‘input.mp4’ is the name of your uploaded video file

Input\_file = list(uploaded\_keys())[0]

#calling driver function

Process\_video(input\_file, ‘output.mp4’)

## 6.1 Introduction to testing:

## CHAPTER -6 TESTING

Testing in software development is a critical step that seeks to identify faults, issues, or any shortcomings in a system. It guarantees that the application passes through the expected quality standards, fulfils the required functions, and runs without any malfunctions. For the lane detection system on roads, testing was essential to verify the efficiency of the image processing and lane detection algorithms, the precision of lane recognition in various conditions, and the reliability of the system in real-world driving conditions. Several testing approaches were made and used as a way to confirm the various aspects of the system, thus providing reliable and accurate lane detection.

## Unit Testing:

In unit testing certain elements of the application are tested in isolation with the aim of ascertaining their functionality. As for the lane detection system, the unit tests were created to check the behavior of elements including edge detection, region of interest, Hough Transform, and lane line interpolation. By performing the unit testing, it was a wanted desire to detect any mistakes in the early stages of the development process and be sure that each unit would operate as it was supposed to before proceeding to the stage of merging it into the lane detection system.

## 6.2.2. Integration Testing:

In an effort to assess the ability of the various components of the lane detection system to integrate seamlessly, integration testing was employed. This included their interface involving the edge detection response with the selected region, the detected lines in the Hough Transform, and the integration process that produced the lane lines on the road. This was done during the Integration testing where the test team focused on checking if the system is functional and whether there is a problem of data flow and inconsistency in the interfaces or where two distinct sections of the application may not be in harmony or possess different functionality. That was done to make sure that the lane detection system provide most accurate result where driving happens in realistic condition.

**6.2.3. Functional Testing:**

Functional testing is crucial to ensure that the LDM works as intended and complies with the expected functional requirements. In this work, functional testing included verifying functionality of the adopted method to detect lane lines on the different types of road and under different scenarios. These are the straight roads, curved roads, roads with faded markings and multilane highway. When the actual output was compared with the expected output of a specific function. It was established that the system performed as expected. Functional testing also involved checking how the system performs with testing scenarios like the following: low light, rain or shadow, it tested for graceful degradation in performance without much reduction in reliability

**CHAPTER - 7**

**RESULTS**



Figure-7.1 Input



Figure-7.2 Output

# CHAPTER 8

* 1. **CONCLUSION**

The lane detection is one of the key technologies for the ADAS and AVs; it is the key component for any other safety and navigation systems. This project involves the application of various approaches in computer vision and machine learning for enhancing driving safety. Lane detection is one of the crucial image processing technologies that assist in increasing protection for vehicles and help auto-driving systems. However, the current features and difficulties are improving through technology and algorithms that are making the lane detection system even better and suitable for various applications.

# FUTURE SCOPE

Future research will focus on the use of different and more advanced image processing filters to increase the accuracy of lane detection. It may encompass adaptive filters that automatically respond to varying lighting conditions or those that reduce noise attributed to wetness or reflective road surfaces, thereby hence making the lane detection system even more robust.

# REFERENCES

1.“Real-time Lane Detection Analysis At Night” by Manthan Shenoy, Piyush Sonkusle(published on Research Gate)

2.“Road Lane Line Detection Using OpenCV” by Dr. A.K.Madan, Divyansha Jharwal(published on IRJMETS)

# PLAGIARISM CHECK :

