

LANE LINE DETECTION USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

MAGESHWARI M (191421601023)

&

VINODHINI D (191421601051)

Under the guidance of

PREETI VERMA,

In the partial fulfillment for the award of the degree of

Bachelor of Computer Applications



JULY 2021



www.crescent.education

BONAFIDE CERTIFICATE

Certified that this project report “Secure Cloud Storage Against Packet Injection Attack” is the bonafide work of **MAGESHWARI M (191421601023)** and **VINODHINI D (191421601051)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

SIGNATURE

PREETI VERMA,

Supervisor

Assistant Professor

Department of Computer Applications

B.S. Abdur Rahman Crescent Institute of

Science and Technology, Seethakathi

Estate Vandalur, Chennai -634048.

Dr. S. PAKKIR MOHIDEEN

Head of the Department

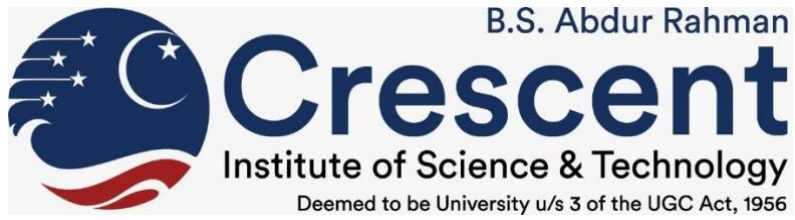
Associate Professor and Head

Department of Computer Applications

B.S. Abdur Rahman Crescent Institute

of Science and Technology, Seethakathi

Estate, Vandalur, Chennai-63048.



VIVA-VOCE EXAMINATION

The viva-voce examination of the project work titled “**LANE LINE DETECTION USING MACHINE LEARNING**” submitted by **MAGESHWARI M (191421601023)** and **VINODHINI D (191421601051)** is held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I thank the Almighty for showering His blessings upon me in completing the project. I submit this project with a deep sense of gratitude and reverence for my beloved parents for their moral support and encouragements.

I sincerely express my heartfelt gratitude to **Dr. A. Peer Mohamed Vice Chancellor**, B.S. Abdur Rahman Crescent Institute of Science and Technology and **Dr. A. Azad, Registrar**, for furnishing every essential facility for doing my project.

I owe my sincere gratitude to **Dr. Venkatesan Selvam**, Professor and Dean School of Computer, Information and Mathematical Science (SCIMS). **Dr. S. Pakkir Mohideen**, Associate Professor and Head, Department of Computer Applications for providing strong oversight of vision, strategic direction, encouragement and valuable suggestions in completing my project work.

I convey my earnest thanks to my project guide **PREETI VERMA**, Assistant Professor, Department of Computer Applications, for her valuable guidance and support throughout the project.

I extend my sincere thanks to all my faculty members for their valuable suggestions, timely advice and support to complete the project.

MAGESHWARI M

VINODHINI D

ABSTRACT

Lane detection plays a key role in building intelligent traffic system. How to improve the accuracy of lane recognition and the ability of curve detection has always been the focus of research. Aiming at the detection of yellow and white lane lines and

curves .this paper introduces a new lane line detection method based on machine vision, which combines the yellow lane line processed in HLS color with the white lane line processed in grayscale space. Through canny edge detection, inverse perspective transformation and sliding window polynomial fitting method to achieve real-time lane detection. The experiment shows that the algorithm can accurately detect the curve and the lane line under the light changing circumstances, and can calculate the vehicle's deviation distance according to the lane line detection results. At the same time, the algorithm shows good reliability and robustness under multiple working condition

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	
	LIST OF TABLES	
	LIST OF ABBREVIATION	
1	INTRODUCTION	7
	1.1 GENERAL	7
	1.2 EXISTING SYSTEM	8
	1.2.1 LITERATURE SURVEY	8
	1.2.2 DISADVANTAGE OF EXISTING SYSTEM	9
	1.3 PROPOSED SYSTEM	9

	1.3.1 ADVANTAGES	9
	1.4 ORGANISATION OF CHAPTERS	9
2	PROBLEM DEFINITION AND METHODOLOGY	10
	2.1 PROBLEM DEFINITION	10
	2.2 METHODOLOGY	10
	2.3 DISCUSSION OF ALGORITHM	10
	2.3.1 CANNY EDGE DETECTION	10
3	DEVELOPMENT PROCESS	11
	3.1 REQUIREMENT ANALYSIS	12
	3.4 ARCHITECTURE DIAGRAM	13
	3.4.1 FLOW DIAGRAM	14
	3.4.1 CLASS DIAGRAM	15
	3.5 IMPLEMENTATION	15
	3.6 TESTING	16
	3.6.1 TYPES OF TESTING	17
	3.7 TEST CASE	18
	3.8 MODULES	18
	3.8.1 IMAGE PROCESSING	18
	3.8.2 CANNY EDGE DETECTION	19
	3.8.3 INVERSE PERSPECTIVE	20
4	RESULTS AND DISCUSSION	21
	4.1 RESULTS	21
	4.2 DISCUSSION	21
5	CONCLUSION AND FUTURE ENHANCEMENT	26
	5.1 CONCLUSION & FUTURE WORKS	26
	APPENDICES: SAMPLE CODE	27
	REFERENCES	29

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	ARCHITECTURE DIAGRAM	13
3.2	DATA FLOW DIAGRAM	14
3.3	CLASS DIAGRAM	15
3.4	IMPLEMENTATION OF MODULES	16
4.1	HLS COLOR	22
4.2	YELLOW LINE DETECTION	23
4.3	WHITE LINE DETECTION	24
4.4	CANNY EDGE DETECTION	25

CHAPTER 1

INTRODUCTION

This chapter deals with the general introduction of the project, existing system and the proposed system for this project. Existing system elucidates the current system and its limitations and proposed system provides a solution to overcome those limitations.

1.1GENERAL

It plays a key role to improve the accuracy of lane line detection and the ability of curve detection has always been the focus of research. Aiming at the detection of yellow and white lane lines and curves, it combines the yellow lane line processed in HSV space with the white lane line processed in gray scale space. Through canny

edge detection, inverse perspective transformation and sliding window polynomial fitting method to achieve real-time lane detection. The experiment shows that the algorithm can accurately detect the curve and the lane line under the light changing circumstances, and can calculate the vehicle's deviation distance according to the lane line detection results. At the same time, the algorithm shows good reliability and robustness under multiple working conditions.

1.2 EXISTING SYSTEM

In existing system they used sobel edge detection to detect edge, and to extract white lane line they used grayscale space. Traditional hough transform is used to carry out the line fitting.

1.2.1 LITERATURE SURVEY

(i) Gulcehre C, Moczulski M, et al. Noisy activation functions[c]. international Conference on Machine learning. 2016: 3059-3068

(ii) Li Mei. Research on image segmentation based on Otsu algorithm[D]. Hefei: Hefei university of technology, 2011.

(iii) Logistics, pp. 980-985, 2009. [4] C. Rotaru, T. Graf, and J. Zhang, "Color image segmentation in HIS space for automotive applications", J. of Real-Time Image Processing, vol. 3, no. 4, pp.311-322, 2008.

(iv) J. G. Kuk, J. H. An, H. Ki, and N. I. Cho. "Fast lane detection & tracking based on Hough transform with reduced memory requirement". Annual Conference on Intelligent Transportation Systems,Madeira Island, Portugal, September, 2010.

(v) Yang Changhong, Zou Xiong, Xu Jiali. "A Novel Edge Detection Algorithm Based on Distance", Journal of Physics: Conference Series, 2019.

(vi) Q. Li, N. Zheng, and H. Cheng, "Springrobot: A prototype autonomous vehicle and its algorithms for lane detection", IEEE Trans. Intell. Transp. Syst, Vol. 5, No. 4, pp. 300-308, Dec. 2004.

1.2.2 DISADVANTAGE OF EXISTING SYSTEM

- The traditional method is used to directly conduct image gray processing, but there is a lot of noise in the processed image.

- Another disadvantage is using Sobel edge detection, that it is simple and more time-efficient. But the edges are rough.
- It uses hough transform to carry out line fitting. however, this detection method can only perform well while detecting the straight lane lines.

1.3 PROPOSED SYSTEM

In proposed system we use canny edge detection to detect edge, and to extract white lane line we use grayscale space, to extract yellow lane line we use HSV space. Sliding window polynomial fitting is used to carry out the straight and curve line fitting.

1.3.1 ADVANTAGES OF PROPOSEDD SYSTEM

- We use different color lane line in HSV space and gray space separately, which can greatly improve the accuracy and processing efficiency of lane line recognition.
- Here we are using canny edge detection which produces smoother edges than sobel edge detection.
- The sliding window is adopted here to detect the pixel points of lane line, The detection accuracy of straight lane lines and curve lane lines and the robustness of the algorithm is significantly improved.

1.4 ORGANISATION OF CHAPTERS

Organization of the report is the short summary of the forthcoming chapters. In this, Chapter 1 deals with the introduction, existing system, and the proposed system of the project were discussed. Chapter 2 deals with problem definition and methodology. Chapter 3 deals with the development process, the requirement analysis, system design, implementation and testing of the system are discussed. The results of the project are analyzed and concluded in the final chapter 4.

CHAPTER 2

PROBLEM DEFINITION AND METHODOLOGY

In this previous chapter, the existing and the proposed system of the project are discussed. This chapter deals with the problem definition and methodology. The problem definition discusses about the objective of the project and the methodology used to develop the project.

2.1 PROBLEM DEFINITION

- ❖ Lane Line detection is a critical component for self-driving cars and also for computer vision in general.
- ❖ In this project we detect the white and yellow lane line and also curve lane line based on machine vision.
- ❖ We use python programming language in this project.
- ❖ For the initial step we detect lane markers then the last step is testing and fitting the lane line.
- ❖ In vision based lane detection image processing and edge& curve detection is the initial problem.
- ❖ We use HSV space & grayscale space to process the image with different color to avoid the noise in image.
- ❖ To detect the edge we use canny edge detection.
- ❖ To detect the curve lines we use sliding window polynomial fitting method.

2.2 METHODOLOGY

- ❖ In this project we are aiming at the detection of yellow lane line & white lane line and curves in real time.
- ❖ The yellow lane line is processed in HLS space and white lane line is processed in grayscale space.
- ❖ In this project we are using canny edge detection.

2.3 DISCUSSION OF ALGORITHM

Canny edge detection algorithm

2.3.1 CANNY EDGE DETECTION

- Noise reduction
- Gradient calculation
- Non maximum suppression
- Double threshold
- Edge tracking by hysteresis

NOISE REDUCTION

Noise reduction is the process of removing noise from a signal. Noise reduction techniques exist for audio and images. Noise reduction algorithms may distort the signal to some degree. All signal processing devices, both analog and digital, have traits that make them susceptible to noise

GRADIENT CALCULATION

The Gradient calculation step detects the edge intensity and direction by calculating the gradient of the image using edge detection operators. The result is almost the expected one, but we can see that some of the edges are thick and others are thin. Non-Max Suppression step will help us mitigate the thick ones

NON MAXIMUM SUPPRESSION

Non-Maximum Suppression. Ideally, the final image should have thin edges. Thus, we must perform non-maximum suppression to thin out the edges. The principle is simple: the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions.

DOUBLE THRESHOLD

The Canny edge detection algorithm uses double thresholding. Edge pixels stronger than the high threshold are marked as strong; edge pixels weaker than the low threshold are suppressed and edge pixels between the two thresholds are marked as weak.

EDGE TRACKING BY HYSTERESIS

Usually a weak edge pixel caused from true edges will be connected to a strong edge pixel while noise responses are unconnected. To track the edge connection, blob analysis is applied by looking at a weak edge pixel and its 8-connected neighborhood pixels.

CHAPTER 3

DEVELOPMENT PROCESS

Development process includes requirement analysis, definition, design, testing and implementation. The input requirements, output requirements and Resource requirements. They are as follows.

3.1 REQUIREMENT ANALYSIS

3.1.1 INPUT REQUIREMENT

We have to input the image to preprocessing training and testing data

3.1.2 OUTPUT REQUIREMENT

It will detect the line and highlight the lane area to obtain the final result of lane line detection

3.2 RESOURCE REQUIREMENT

The hardware and software requirements for this system were analyzed and the required configuration is as given below. The following are the software requirements for the project.

3.2.1 SYSTEM REQUIREMENT

SOFTWARE REQUIREMENT

- Operating system :windows 10
- Coding language :python
- Front end tool :Google colab
- Front end :python
- Back end tool :python

HARDWARE REQUIREMENT

- Processor :Intel
- Ram : 4GB
- Hard disk :250GB and above

3.3 SYSTEM DESIGN

The design explains the various modules of the project, the overall architecture of the project, workflows and data flows of the project etc. Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the discipline of systems analysis, systems architecture and systems engineering.

3.4 ARCHITECTURE DIAGRAM

This Architecture Diagram explains how the user interacts with the detector and gets the output based on the input.

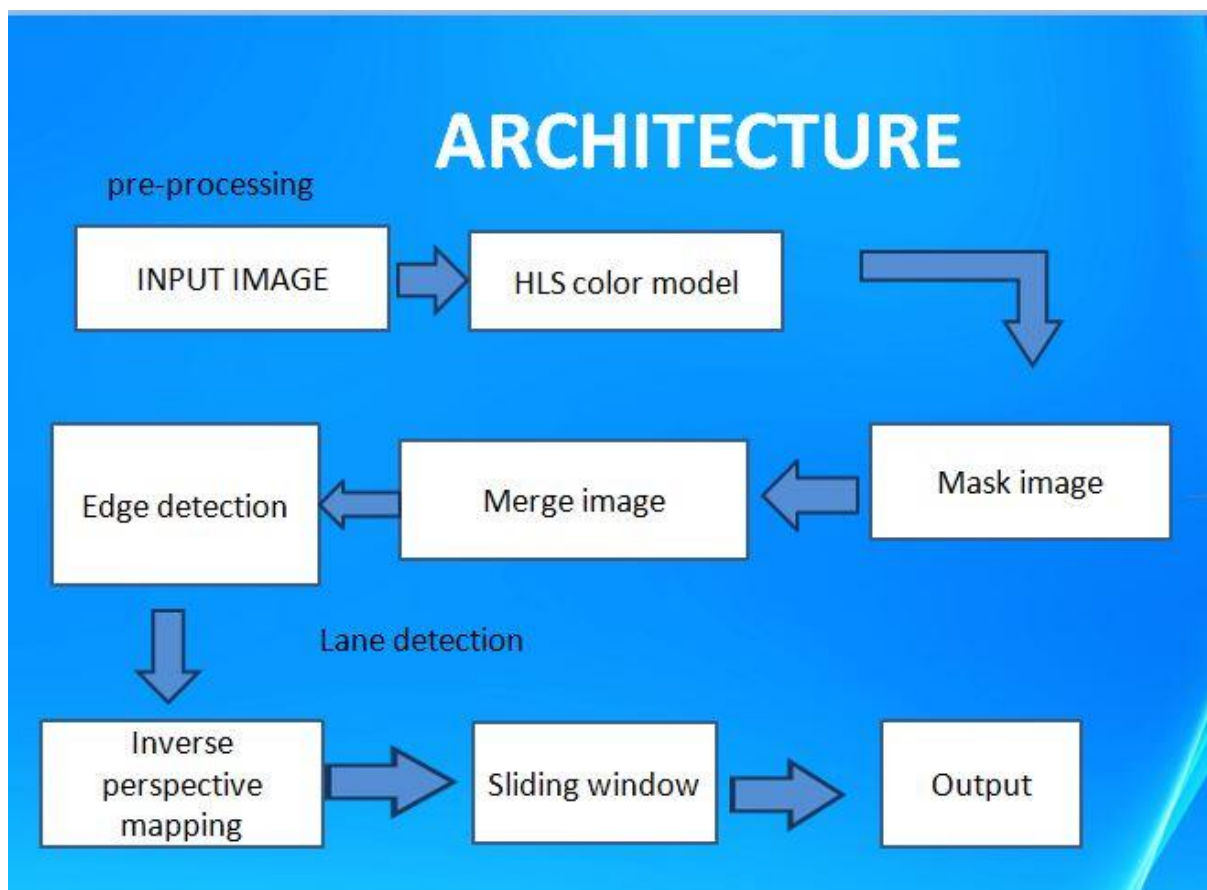


FIGURE 3.1 Architecture diagram

3.4.1 DATA FLOW DIAGRAM

This flow chart explains from start to end, it shows how it works and how the tasks are given and how the output is generated based on the tasks. The following figure 3.1 represents the flow chart diagram.

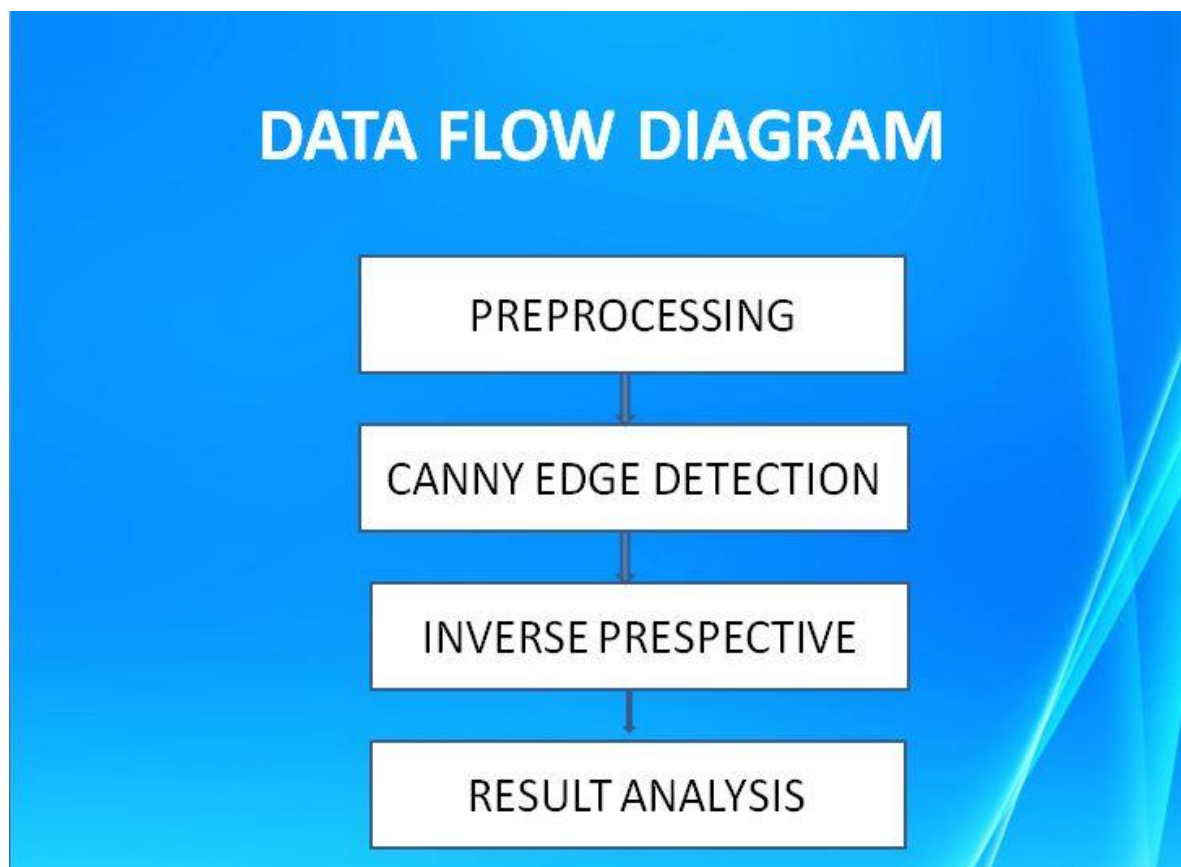


FIGURE 3.2 DATA FLOW DIAGRAM

3.4.2 Class diagram

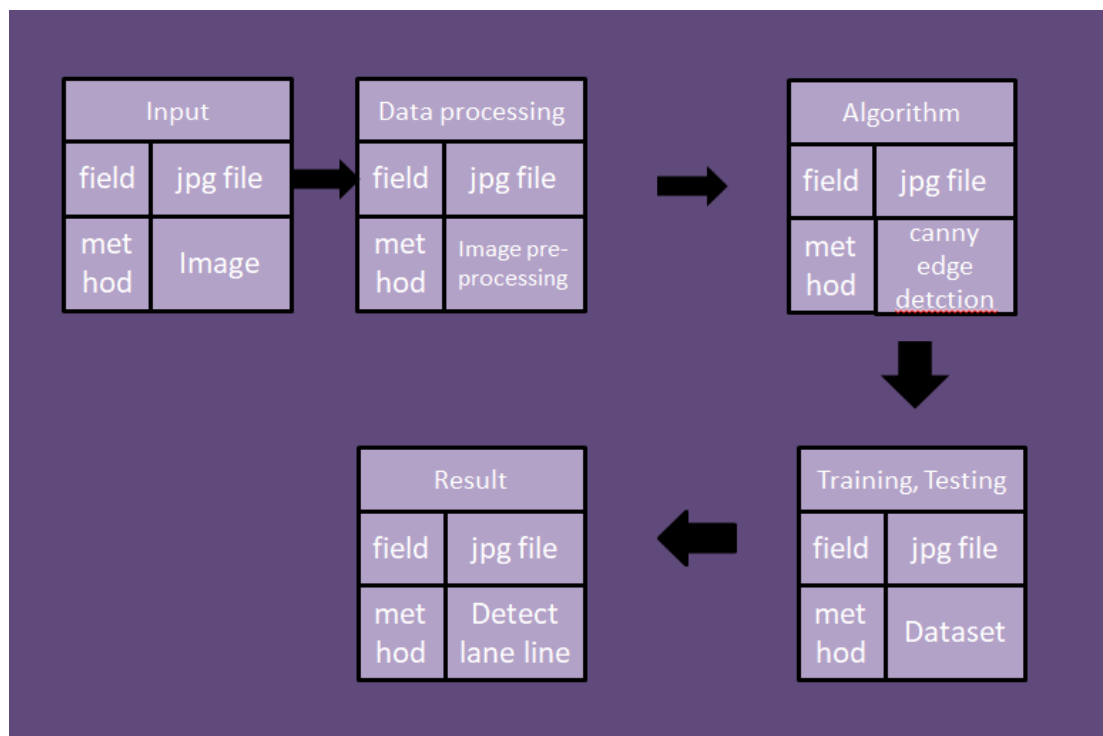
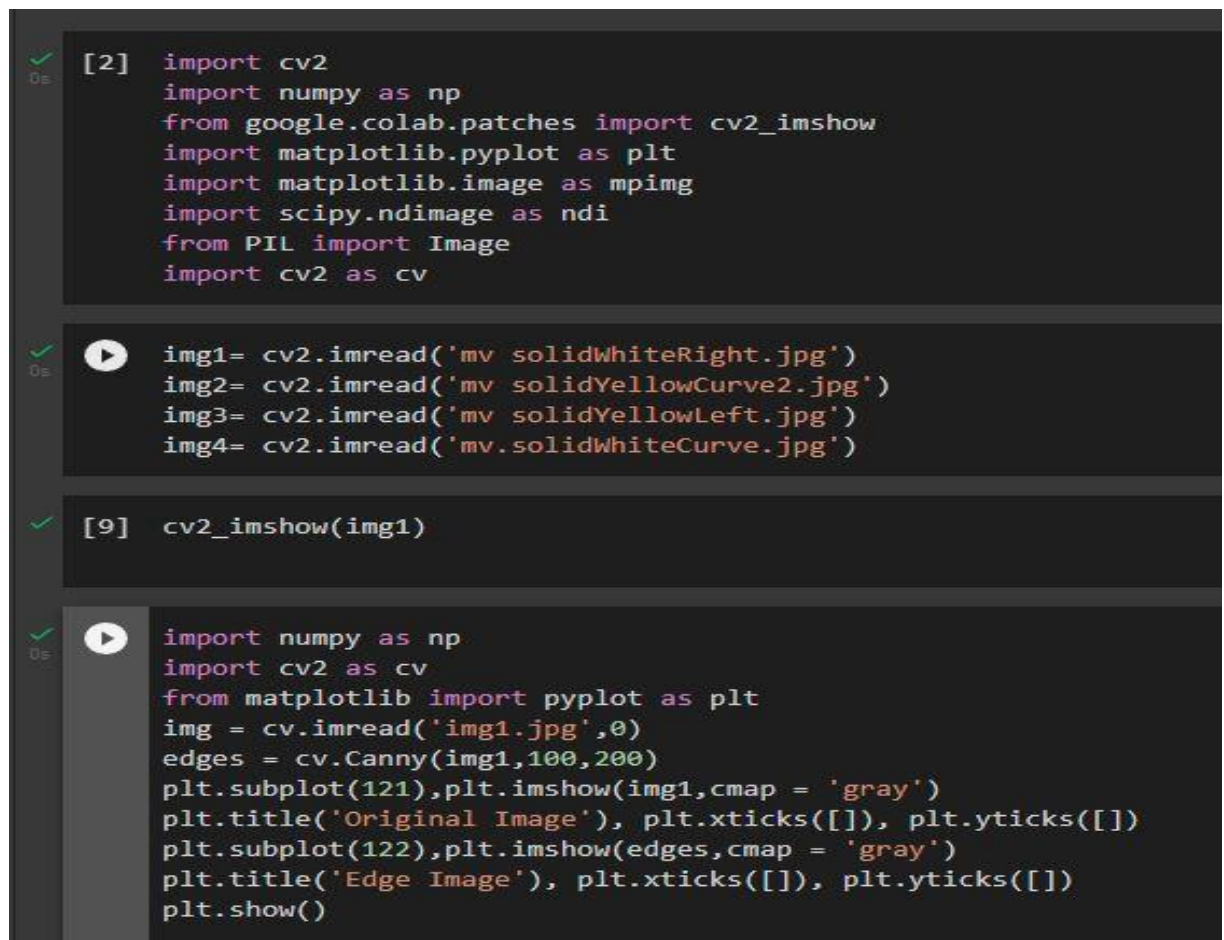


Figure 3.3 class diagram

3.5 IMPLEMENTATION

In the implementation we can see implementation's made in this project .The following figure 3.1 and figure 3.2 represents the implementation.



```
[2] import cv2
import numpy as np
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import scipy.ndimage as ndi
from PIL import Image
import cv2 as cv

img1= cv2.imread('mv_solidWhiteRight.jpg')
img2= cv2.imread('mv_solidYellowCurve2.jpg')
img3= cv2.imread('mv_solidYellowLeft.jpg')
img4= cv2.imread('mv_solidWhiteCurve.jpg')

[9] cv2_imshow(img1)

import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('img1.jpg',0)
edges = cv.Canny(img1,100,200)
plt.subplot(121),plt.imshow(img1,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
plt.show()
```

Figure 3.4 Implementation of modules

3.6 TESTING

In the testing part first the source code is tested to check whether the code executes the expected output. Then the Functionality of the project is tested to check the expected output is obtained. Then the interface of the project is tested to check the interface works properly. Then the modules are tested separately to check whether the expected output is obtained. Testing is a process of executing a program with the intent of finding an error. The objective of software testing is to uncover errors. Software testing, depending on the testing method employed, can be implemented at any time in the development process, however most of the effort is employed after the requirements have been defined and coding process has been completed. Testing is usually performed for the following purposes:

- To improve quality

- For verification and validation
- For reliability estimation

3.6.1 TYPES OF TESTING

- Functional Testing
- Integration Testing
- Source code Testing
- Module Level Testing

FUNCTIONAL TESTING

Functional testing is mainly used as a Quality Assurance process. This is a very simple process where each function is provided with an appropriate input and is verified against an expected output and with boundary values. This would help in ensuring that the output is as acquired according to the expectations and would help assuring the quality.

INTEGRATION TESTING

It is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with in the interface. Integration of all the components to form the entire system and an overall testing is executed. Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together as in a on the whole approach. It is mainly done based on by taking into account of the number of modules used, how many numberof interfaces maybe required to integrate them, which had to be combined and clustering process.

SOURCE CODE TESTING

This examines the logic of the system. If we are getting the output that is required by the user, then we can say that the logic is perfect.

MODULE LEVEL TESTING

In this the error will be found at each individual module. It, encourages the programmer to find and rectify the errors without affecting the other modules.

3.7 TEST CASE

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly. A specific executable test that examines all aspects including inputs and outputs of a system and then provides a detailed description of the steps that should be taken, the results that should be achieved, and other elements that should be identified. Steps explained in a test case include all details even if it is assumed to be common knowledge. Test cases are used as a used as a technical explanation and reference guide for system.

3.8 MODULES

3.8.1 Image processing

Image processing is the process of transforming an image into a digital form and performing certain operations to get some useful information from it. The image processing system usually treats all images as 2D signals when applying certain predetermined signal processing methods. Image processing requires fixed sequences of operations that are performed at each pixel of an image. The image processor performs the first sequence of operations on the image, pixel by pixel. Once this is fully done, it will begin to perform the second operation, and so on. The output value of these operations can be computed at any pixel of the image.

3.8.2 Canny edge detection

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works. Canny edge detection is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. It has been widely applied in various computer vision systems.

The Canny edge detection algorithm is composed of 5 steps:

1. Noise reduction;

Noise reduction is the process of removing noise from a signal. Noise reduction techniques exist for audio and images. Noise reduction algorithms may distort the signal to some degree. All signal processing devices, both analog and digital, have traits that make them susceptible to noise

2. Gradient calculation;

The Gradient calculation step detects the edge intensity and direction by calculating the gradient of the image using edge detection operators. The result is almost the expected one, but we can see that some of the edges are thick and others are thin. Non-Max Suppression step will help us mitigate the thick ones

3. Non-maximum suppression;

Non-Maximum Suppression. Ideally, the final image should have thin edges. Thus, we must perform non-maximum suppression to thin out the edges. The principle is simple: the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions.

4. Double threshold;

The Canny edge detection algorithm uses double thresholding. Edge pixels stronger than the high threshold are marked as strong; edge pixels weaker than the low threshold are suppressed and edge pixels between the two thresholds are marked as weak.

5. Edge Tracking by Hysteresis

Usually a weak edge pixel caused from true edges will be connected to a strong edge pixel while noise responses are unconnected. To track the edge connection, blob analysis is applied by looking at a weak edge pixel and its 8-connected neighborhood pixels.

3.8.3 Inverse perspective

The inverse perspective mapping scheme is another method for obtaining a bird's eye view of the scene from a perspective image. The inverse perspective mapping technique can also be used to remove the perspective distortion caused by the perspective projection of a 3D scene into a 2D image

CHAPTER 4

RESULT AND DISCUSSION

Chapter 3 discussed about the requirements analysis, overall design of the project and implementation. This chapter discusses about the conclusion and future enhancements

4.1 RESULTS

The result of the project “LANE LINE DETECTION” is obtained as expected. This project is implemented using canny edge detection algorithm. The experiment shows that the algorithm can accurately detect the curve and the lane

4.2 DISCUSSION

This whole system deals with detecting the yellow line, white line and edge. The following figure represents the outputs.

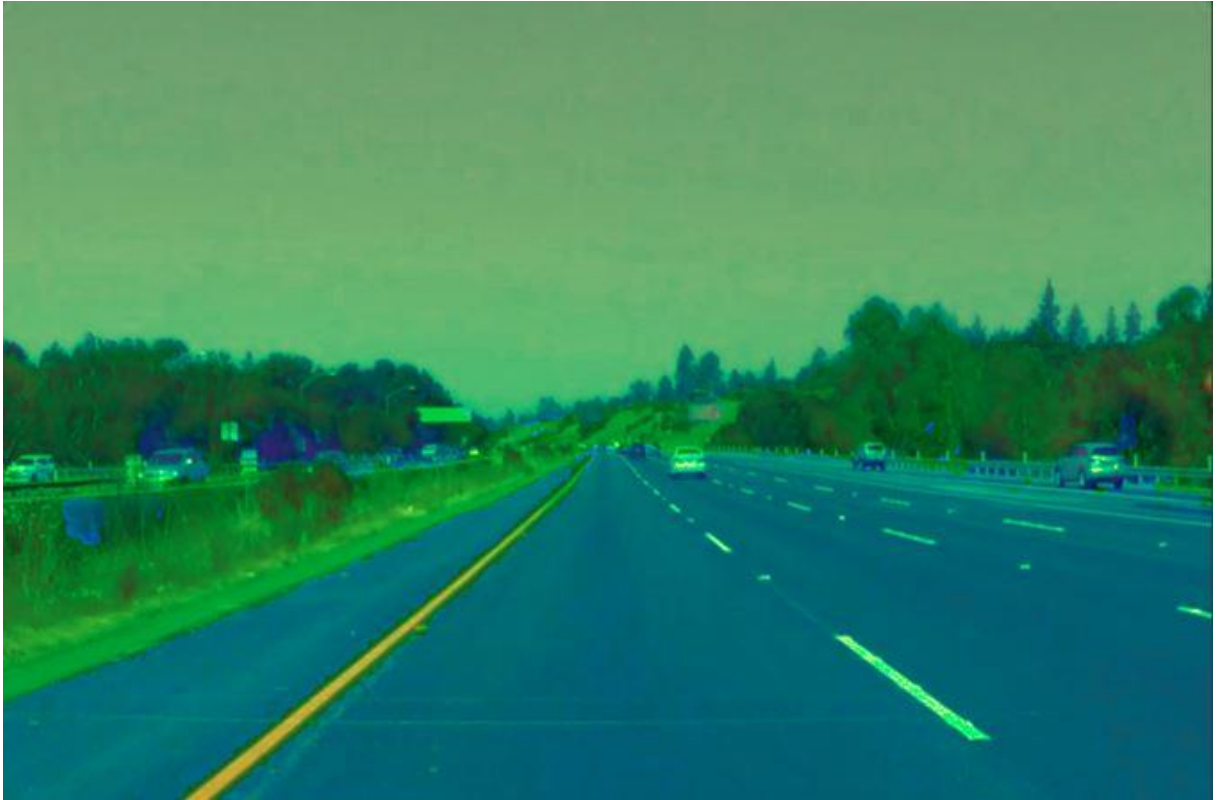


Figure 4.1 HLS color



Figure 4.2 yellow line extraction



Figure 4.3 white line extraction

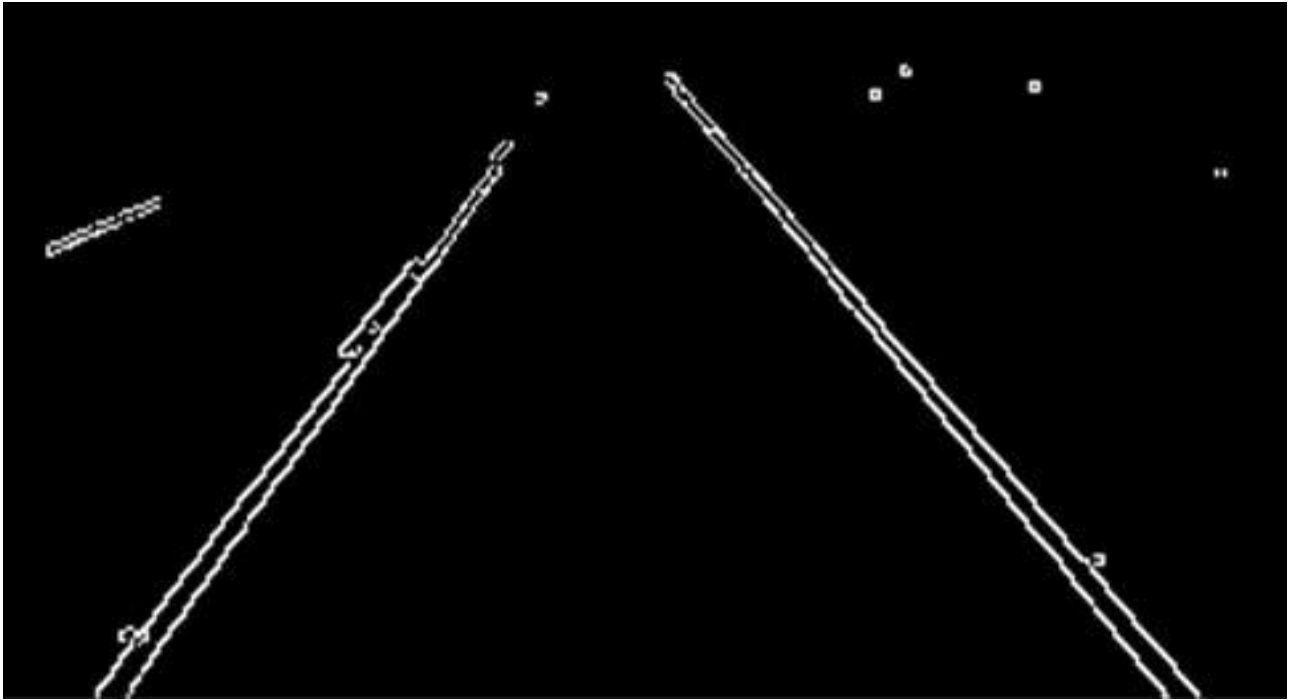


Figure 4.3 canny edge detection

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1 CONCLUSION

The images used in the test were taken from straight roads, curves. The first feature of this algorithm is that it realizes the high accuracy detection of the yellow lane line in HLS color, and the second feature is that it uses the sliding window to fit the equation of a lane line. Meanwhile, the system can be extended to the road of continuous curves by setting more base points. Compared with other lane detection models, this algorithm can adapt to more complex and larger range of lane structures, and its robustness lies in that the correct road marking information obtained from the image is only a few pixels more than the error information. Therefore, the system can adapt to a variety of road conditions, with good robustness and effectiveness.

5.2 FUTURE WORK

Future work will include a new fitting method to detect the pixel point of lane line, which will effectively help the detection accuracy of straight lane line and the robustness of the algorithm is significantly improve.

APPENDIX

SAMPLE SOURCE CODE

```
import cv2

import numpy as np
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import scipy.ndimage as ndi
from PIL import Image
import cv2 as cv
img1= cv2.imread('mv solidWhiteRight.jpg')
img2= cv2.imread('mv solidYellowCurve2.jpg')
img3= cv2.imread('mv solidYellowLeft.jpg')
img4= cv2.imread('mv.solidWhiteCurve.jpg')
cv2_imshow(img1)
hlsimg4 = cv2.cvtColor(img4,cv2.COLOR_BGR2HLS)
hlsimg2 = cv2.cvtColor(img3,cv2.COLOR_BGR2HLS)
cv2_imshow(hlsimg1)
cv2_imshow(hlsimg2)
yellow_lower = np.array([ 10, 0, 100])
yellow_upper = np.array([ 100, 255, 255])
yellow_line= cv2.inRange(hlsimg1, yellow_lower, yellow_upper)
cv2_imshow(yellow_line)
yellow_lower = np.array([ 10, 0, 100])
yellow_upper = np.array([ 100, 255, 255])
yellow_line1 = cv2.inRange(hlsimg2, yellow_lower,yellow_upper)
cv2_imshow(yellow_line1)
image = cv2.cvtColor(img1,cv2.COLOR_BGR2HLS)
lower = np.uint8([0, 200, 0])
upper = np.uint8([255, 255, 255])
white_line = cv2.inRange(image, lower, upper)
```

```

cv2_imshow(white_line)
def on_change(val):
    print(val)
    alpha = val/100
    beta = (1.0 - alpha)
    result = cv2.addWeighted(img1, alpha, img2, beta, 0.0);
    cv2.imshow('blend', result)
img1 = cv2.imread('white_line')
img2 = cv2.imread('yellow_line')
img1 = cv2.resize(white_line, (400, 400))
img2 = cv2.resize(yellow_line, (400, 400))
blended = cv2.addWeighted(img1, 0.5, img2, 0.5, 0)
cv2_imshow(blended)
#canny

img = cv.imread('blended',0)
edges = cv.Canny(blended,100,200)
plt.subplot(121),plt.imshow(blended,cmap = 'gray')
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
plt.show()
#mask
image_mask = edges
cv2.rectangle(image_mask, (0,0), (500,-15),0,480)

cv2_imshow(image_mask)
plt.imshow(image_mask)

cropped_image = image_mask[225:400,40:380]
cv2_imshow(cropped_image)

```

REFERENCES

- (i) Gulcehre C, Moczulski M, et al. Noisy activation functions. International Conference on Machine learning: 3059-3068 2016
- (ii) Li Mei Research on image segmentation based on Otsu algorithm . Hefei: Hefei University of technology, 2011.
- (iii) Rotaru, T. Graf, and J. Zhang, "Color image segmentation in HIS space for automotive applications", J. of Real-Time Image Processing, vol. 3, no. 4, pp.311-322, 2008.
- (iv) J. G. Kuk, J. H. An, H. Ki, and N. I. Cho. "Fast lane detection & tracking based on Hough transform with reduced memory requirement". Annual Conference on Intelligent Transportation Systems, Madeira Island, Portugal, September, 2010.
- (v) Yang Changhong, Zou Xiong, Xu Jiali. "A Novel Edge Detection Algorithm Based on Distance", Journal of Physics: Conference Series,
- (vi) Q. Li, N. Zheng, and H. Cheng, "Springrobot: A prototype autonomous vehicle and its algorithms for lane detection", IEEE Trans. Intell. Transp. pp. 300-308, Dec. 2004