A

Major Project On

Road Lane Line Detection Using Computer Vision

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

M. SAI VENKAT SANJAY (187R1A05G2)
K. AKASH (187R1A05K8)
K. ABHILASH (187R1A05F8)

Under the Guidance of

G. Vijay Kumar
(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2019-2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**Road Lane Line Detection Using Computer Vision**" being submitted by **M. SAI VENKAT SANJAY (187R1A05G2), K. AKASH (187R1A05K8) & K. ABHILASH (187R1A05F8)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2022-23

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

G. Vijay Kumar
(Assistant Professor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held_____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **G. Vijay Kumar,** Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Dr. Punyaban Patel, Ms. Shilpa, Dr. M. Subha Mastan Rao & J. Narasimharao** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

M. SAI VENKAT SANJAY (187R1A05G2)

K. AKASH (187R1A05K8)

K. ABHILASH (187R1A05F8)

ABSTRACT

In recent times many technological advancements are coming in the domain of road safety as accidents has been increasing at an alarming rate and one of the crucial reason for such accidents is lack of driver's attention. Technical advancements should be there to reduce the frequency of the accidents and stay safe. One of the way to achieve the same is through Lane Detection Systems which work with the intention to recognize the lane borders on road and further prompts the driver if he switches and moves to erroneous lane markings. Lane detecting system is an essential component of many technologically intelligent transport system. Although it's a complex goal to achieve because of vacillating road conditions that a person encounters specially while driving at night or even in daylight. Lane boundaries is detected using a camera that captures the view of the road, mounted on the front of the vehicle. The approach used in this paper changes the image taken from the video into a set of sub-images and generates image-features for each of them which are further used to detect the lanes present on the roads. There are proposed numerous ways to detect the lane markings on the road. Feature-based or model-based are the two categories of the lane detection techniques. Down-level characteristics for example lane-mark edges are used by the feature-based functions.

LIST OF FIGURES/TABLES

SCREENSHOT NO	LIST OF SCREENSHOTS	PAGE NO.

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

The traffic safety becomes more and more convincing with the increasing urban traffic. Exiting the lane without following proper rules is the root cause of most of the accidents on the avenues.

1.2 PROJECT PURPOSE

Lane discipline is crucial to road safety for drivers and pedestrians alike. The system has an objective to identify the lane marks. It's intent is to obtain a secure environment and improved traffic surroundings. The functions of the proposed system can range from displaying road line positions to the driving person on any exterior display, to more convoluted applications like detecting switching of the lanes in the near future so that one can prevent concussions caused on the highways. Actuate detection of lane roads is a critical issue in lane detection and departure warning systems. If an automobile crosses a lane confinement then vehicles enabled with predicting lane borders system directs the vehicles to prevent collisions and generates an alarming condition.

1.3 PROJECT FEATURES

. The algorithm followed in this paper is to detect lane markings on the road by giving the video of the road as an input to the system by using computer vision technology and primarily designed with the objective of reducing the frequency of accidents. System can be installed in cars and taxis in order to prevent the occurrence of accidents due to reckless driving on the roads. In school buses as it will guarantee the safety of the children.

2.SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1 PROBLEM DEFINITION

Given an image captured from a camera attached to a vehicle moving on a road in which captured road may or may not be well levelled, or have clearly delineated edges, or some prior known patterns on it, then road detection from a single image can be applied to find the road in an image so that it could be used as a part in automation of driving system in the vehicles for moving the vehicle in correct road. In this process of finding the road in the image captured by the vehicle, we can use some algorithms for vanishing point detection using Hough transform space, finding the region of interest, edge detection using canny edge detection algorithm and then road detection. We use thousands of images of different roads to train our model so that the model could detect the road which is present in the new image processed through the vehicle.

2.2 EXISTING SYSTEM

In the current existing system is permitted only to use in ideal road conditions such as runway. This could not be used in general roads because the edge detection used till now was Simulink Edge Detection which is implemented in MATLAB. The secondary thing is in current system Hough transform Space is only used for angle rotation and has very limited road dataset to detect the objects in single dimension of an image.

2.2.1 DISADVANTAGES OF EXISTING SYSTEM

Following are the disadvantages of existing system:

- They only consistently work for structured roads which have noticeable markings or bord.

2.3 PROPOSED SYSTEM

In our proposed system we use Canny Edge Detection replacing the Simulink Edge Detection which is recent and efficient implementation in Python instead of MATLAB. Since, Python is the Scripting and Statistical Modelling Language it supports faster execution for mathematical functions which could be used by Canny Edge Detection technique. Secondly, we use Hough Transform Space for 3-Dimensional Object detection which could faster and accurate compared to single dimension object detection.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

Following are the Advantages of proposed system:

- There are proposed numerous ways to detect the lane markings on the road.
- Lane detecting system is an essential component of many technologically intelligent transport system.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 BEHAVIORAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

For developing the application, the following are the Hardware Requirements:

- Operating system: windows, linux
- Processor: minimum intel i3
- Ram: minimum 4 gb
- Hard disk : minimum 250gb

2.5.2 SOFTWARE REQUIREMENTS:

For developing the application, the following are the Software Requirements:

- Python idel 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

3. ARCHITECTURE

3.ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for classification, starting from input to final prediction.

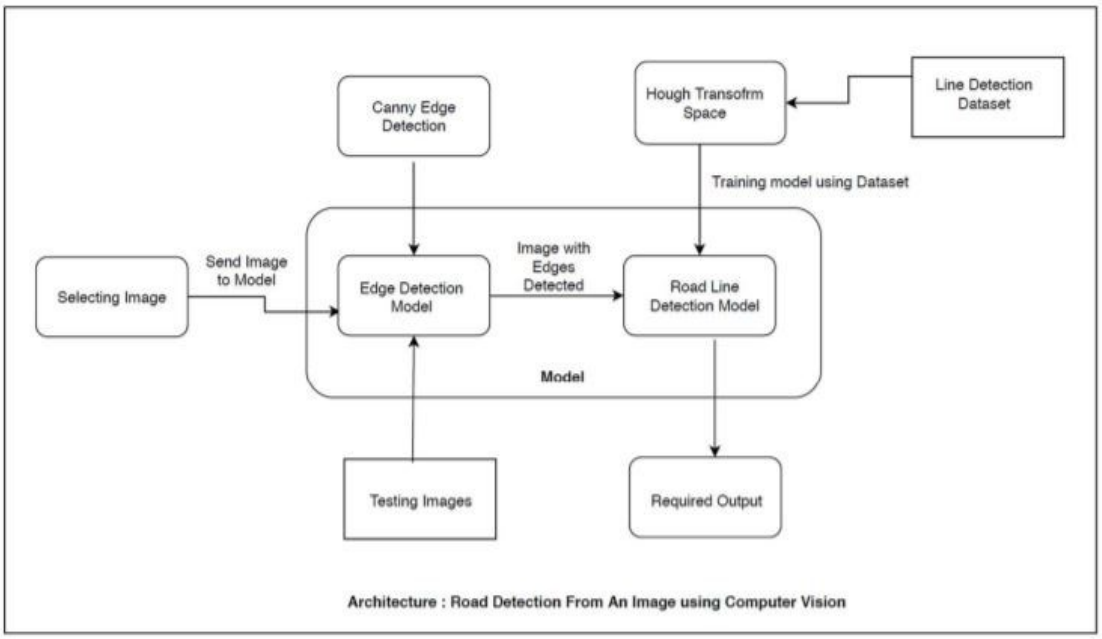

Figure 3.1: Project Architecture of Diagnosis of lung cancer based on ct scan using cnn

3.2 DESCRIPTION

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

3.3 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.

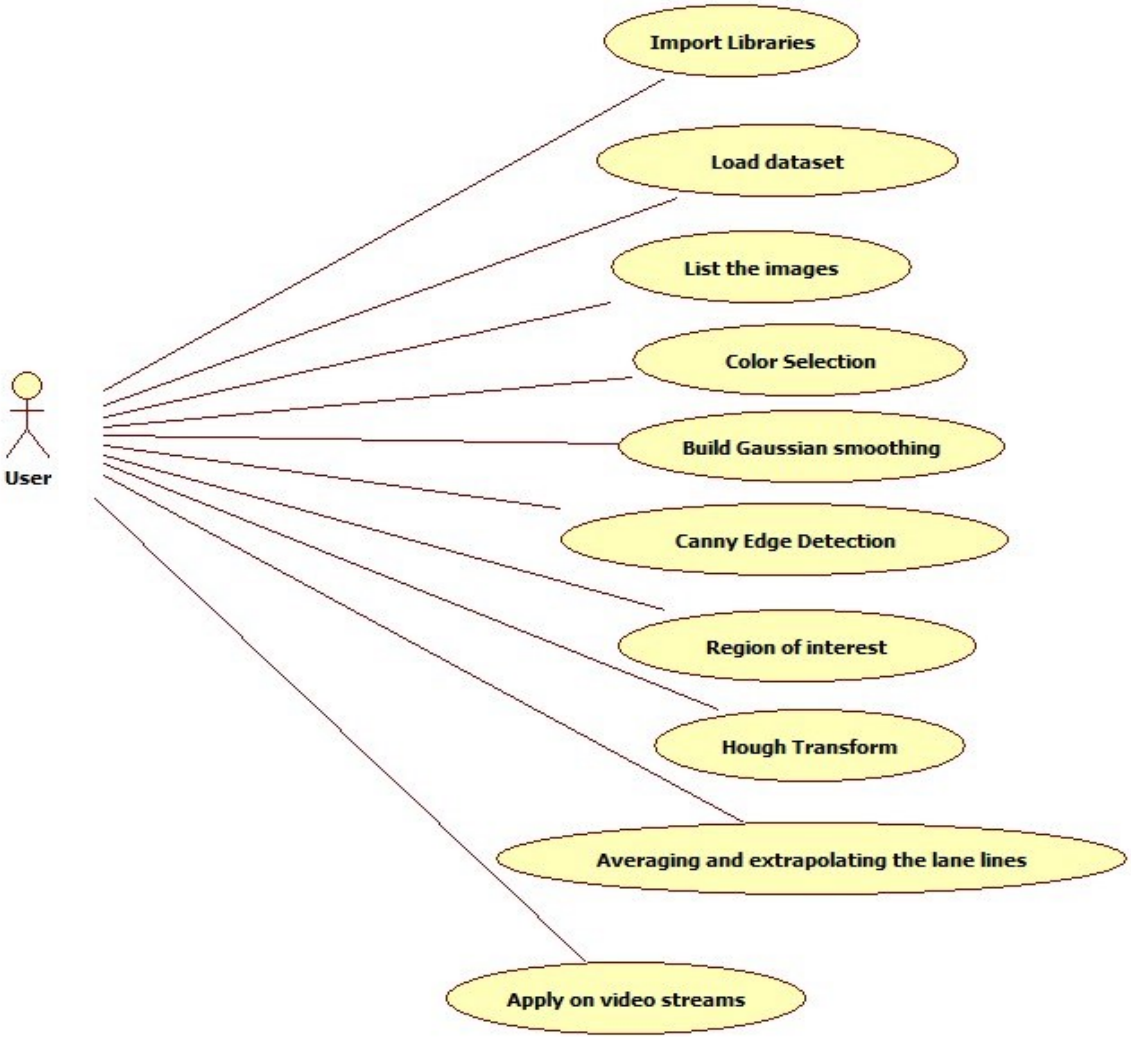

Figure 3.3: Use Case Diagram for Diagnosis of lung cancer based on ct scan using cnn

3.4 CLASS DIAGRAM

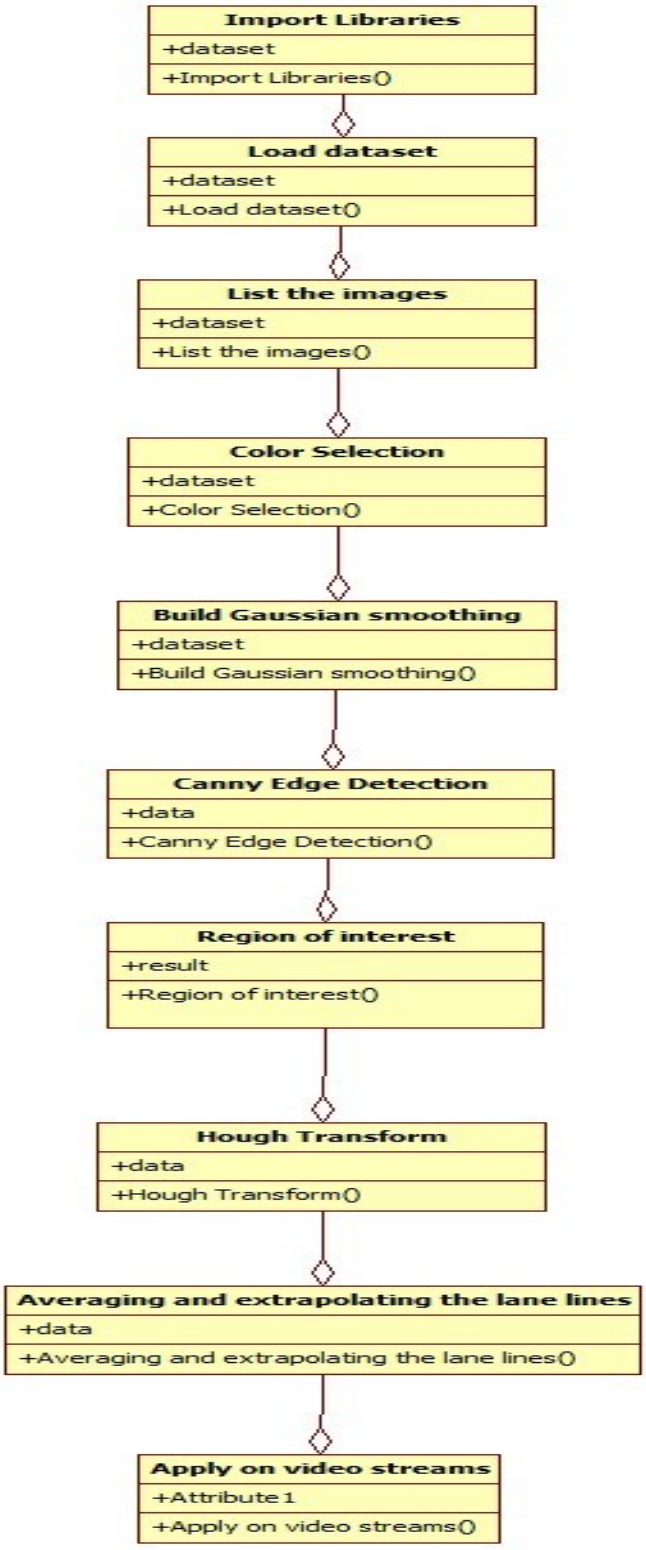

Figure 3.3: Class Diagram for Diagnosis of lung cancer based on ct scan using cnn

3.5 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

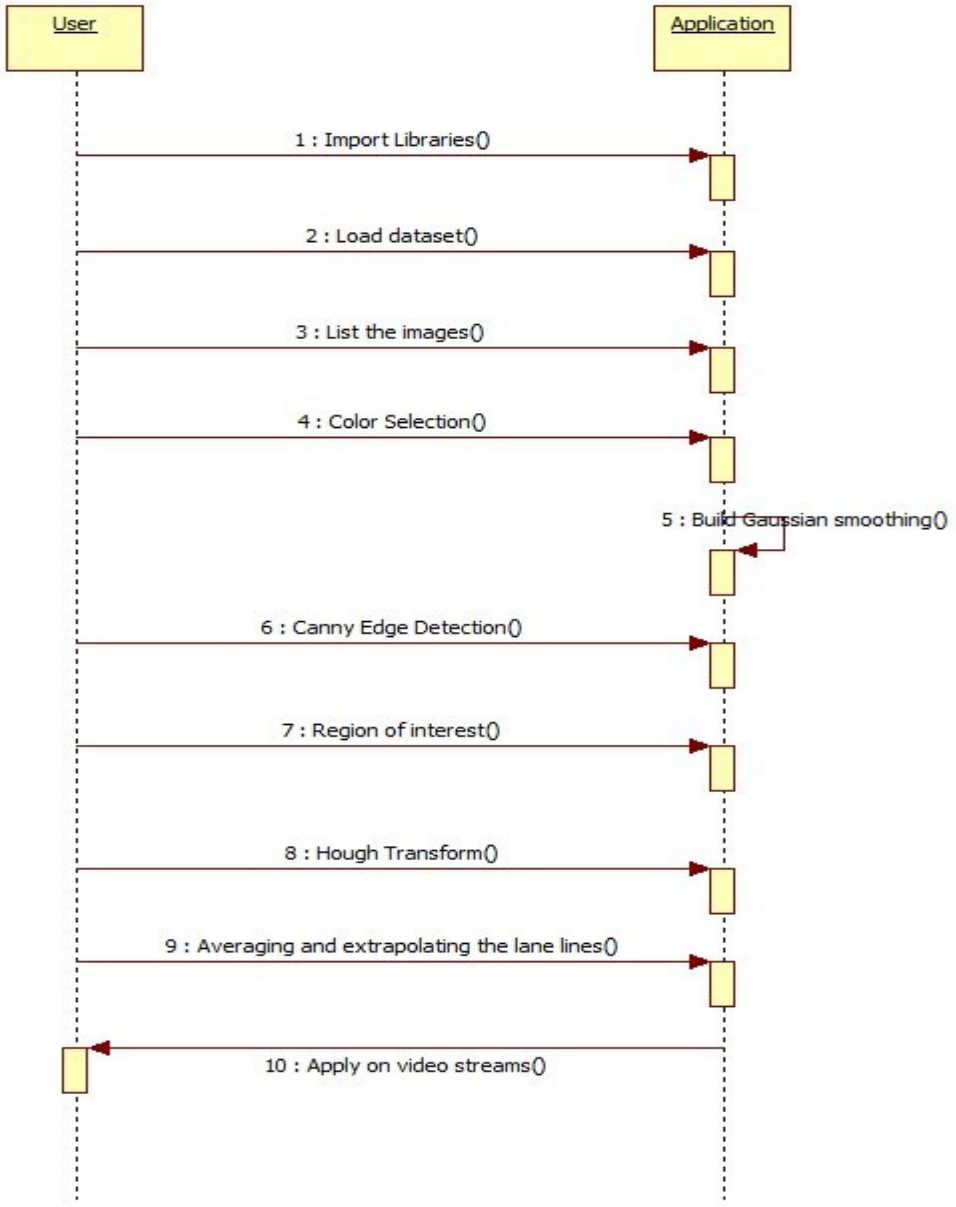

Figure 3.4: Sequence Diagram for Diagnosis of lung cancer based on ct scan using cnn

3.6 ACTIVITY DIAGRAM

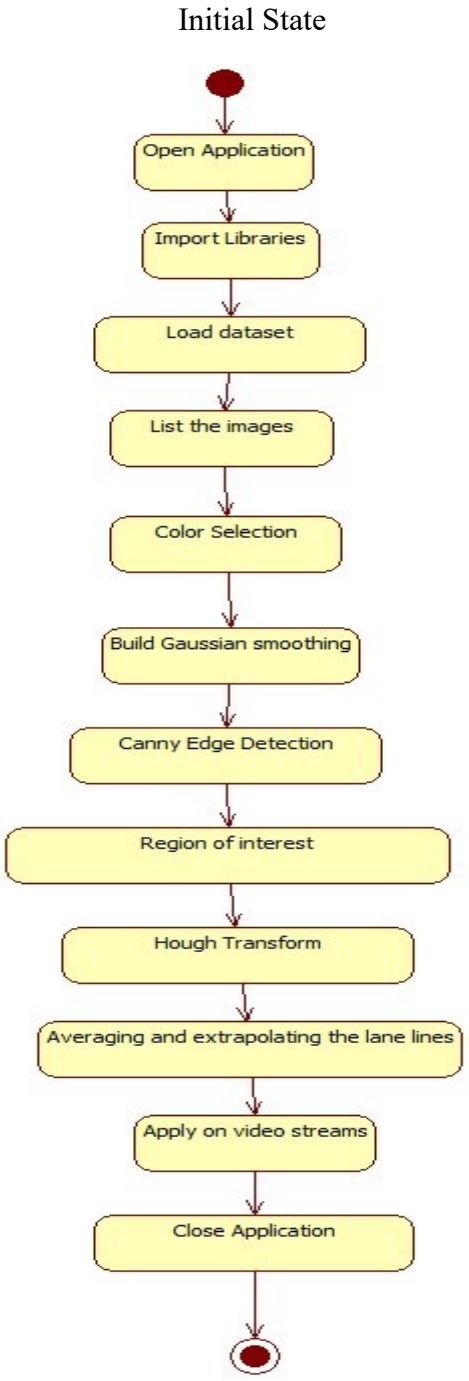

Figure 3.5: Activity Diagram for Diagnosis of lung cancer based on ct scan using cnn

4.IMPLEMENTATION

4.IMPLEMENTATION

4.1 SAMPLE CODE

```
#Importing some useful packages
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import cv2
import os
import glob
from moviepy.editor import VideoFileClip


def list_images(images, cols = 2, rows = 5, cmap=None):
"""
Display a list of images in a single figure with matplotlib.
Parameters:
images: List of np.arrays compatible with plt.imshow.
cols (Default = 2): Number of columns in the figure.
rows (Default = 5): Number of rows in the figure.
cmap (Default = None): Used to display gray images.
"""
plt.figure(figsize=(10, 11))
for i, image in enumerate(images):
plt.subplot(rows, cols, i+1)
#Use gray scale color map if there is only one channel
cmap = 'gray' if len(image.shape) == 2 else cmap
plt.imshow(image, cmap = cmap)
plt.xticks([])
plt.yticks([])
plt.tight_layout(pad=0, h_pad=0, w_pad=0)
plt.show()

#Reading in the test images
test_images = [plt.imread(img) for img in glob.glob('test_images/*.jpg')]
#list_images(test_images)

def RGB_color_selection(image):
"""
```

```
Apply color selection to RGB images to blackout everything except for white
and yellow lane lines.
Parameters:
image: An np.array compatible with plt.imshow.
"""
#White color mas
lower_threshold = np.uint8([200, 200, 200])
upper_threshold = np.uint8([255, 255, 255])
white_mask = cv2.inRange(image, lower_threshold, upper_threshold)

#Yellow color mask
lower_threshold = np.uint8([175, 175,   0])
upper_threshold = np.uint8([255, 255, 255])
yellow_mask = cv2.inRange(image, lower_threshold, upper_threshold)

#Combine white and yellow masks
mask = cv2.bitwise_or(white_mask, yellow_mask)
masked_image = cv2.bitwise_and(image, image, mask = mask)

return masked_image

#list_images(list(map(RGB_color_selection, test_images)))

def convert_hsv(image):
"""
Convert RGB images to HSV.
Parameters:
image: An np.array compatible with plt.imshow.
"""
return cv2.cvtColor(image, cv2.COLOR_RGB2HSV)

#list_images(list(map(convert_hsv, test_images)))

def HSV_color_selection(image):
"""
Apply color selection to the HSV images to blackout everything except for
white and yellow lane lines.
Parameters:
image: An np.array compatible with plt.imshow.
"""
#Convert the input image to HSV
converted_image = convert_hsv(image)
```

```
#White color mask
lower_threshold = np.uint8([0, 0, 210])
upper_threshold = np.uint8([255, 30, 255])
white_mask = cv2.inRange(converted_image, lower_threshold,
upper_threshold)
#Yellow color mask
lower_threshold = np.uint8([18, 80, 80])
upper_threshold = np.uint8([30, 255, 255])
yellow_mask = cv2.inRange(converted_image, lower_threshold,
upper_threshold)

#Combine white and yellow masks
mask = cv2.bitwise_or(white_mask, yellow_mask)
masked_image = cv2.bitwise_and(image, image, mask = mask)

return masked_image

#list_images(list(map(HSV_color_selection, test_images)))

def convert_hsl(image):
"""
Convert RGB images to HSL.
Parameters:
image: An np.array compatible with plt.imshow.
"""
return cv2.cvtColor(image, cv2.COLOR_RGB2HLS)

#list_images(list(map(convert_hsl, test_images)))

def HSL_color_selection(image):
"""
Apply color selection to the HSL images to blackout everything except for
white and yellow lane lines.
Parameters:
image: An np.array compatible with plt.imshow.
"""
#Convert the input image to HSL
converted_image = convert_hsl(image)

#White color mask
lower_threshold = np.uint8([0, 200, 0])
upper_threshold = np.uint8([255, 255, 255])
```

```
white_mask = cv2.inRange(converted_image, lower_threshold,
upper_threshold)

#Yellow color mask
lower_threshold = np.uint8([10, 0, 100])
upper_threshold = np.uint8([40, 255, 255])
yellow_mask = cv2.inRange(converted_image, lower_threshold,
upper_threshold)

#Combine white and yellow masks
mask = cv2.bitwise_or(white_mask, yellow_mask)
masked_image = cv2.bitwise_and(image, image, mask = mask)

return masked_image

#list_images(list(map(HSL_color_selection, test_images)))

color_selected_images = list(map(HSL_color_selection, test_images))

def gray_scale(image):
"""
Convert images to gray scale.s
Parameters:
image: An np.array compatible with plt.imshow.
"""
return cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

gray_images = list(map(gray_scale, color_selected_images))
#list_images(gray_images)

def gaussian_smoothing(image, kernel_size = 13):
"""
Apply Gaussian filter to the input image.
Parameters:
image: An np.array compatible with plt.imshow.
kernel_size (Default = 13): The size of the Gaussian kernel will affect the
performance of the detector.
It must be an odd number (3, 5, 7, ...).
"""
return cv2.GaussianBlur(image, (kernel_size, kernel_size), 0)

blur_images = list(map(gaussian_smoothing, gray_images))
#list_images(blur_images)
```

```
def canny_detector(image, low_threshold = 50, high_threshold = 150):
"""
Apply Canny Edge Detection algorithm to the input image.
Parameters:
image: An np.array compatible with plt.imshow.
low_threshold (Default = 50).
high_threshold (Default = 150).
"""
return cv2.Canny(image, low_threshold, high_threshold)

edge_detected_images = list(map(canny_detector, blur_images))
#list_images(edge_detected_images)

def region_selection(image):
"""
Determine and cut the region of interest in the input image.
Parameters:
image: An np.array compatible with plt.imshow.
"""
mask = np.zeros_like(image)
#Defining a 3 channel or 1 channel color to fill the mask with depending on
the input image
if len(image.shape) > 2:
channel_count = image.shape[2]
ignore_mask_color = (255,) * channel_count
else:
ignore_mask_color = 255
#We could have used fixed numbers as the vertices of the polygon,
#but they will not be applicable to images with different dimesnions.
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, top_right, bottom_right]],
dtype=np.int32)
cv2.fillPoly(mask, vertices, ignore_mask_color)
masked_image = cv2.bitwise_and(image, mask)
return masked_image

masked_image = list(map(region_selection, edge_detected_images))
#list_images(masked_image)
```

```
def hough_transform(image):
"""
Determine and cut the region of interest in the input image.
Parameters:
image: The output of a Canny transform.
"""
rho = 1             #Distance resolution of the accumulator in pixels.
theta = np.pi/180    #Angle resolution of the accumulator in radians.
threshold = 20       #Only lines that are greater than threshold will be
returned.
minLineLength = 20   #Line segments shorter than that are rejected.
maxLineGap = 300     #Maximum allowed gap between points on the same
line to link them
return cv2.HoughLinesP(image, rho = rho, theta = theta, threshold =
threshold,
minLineLength = minLineLength, maxLineGap = maxLineGap)

hough_lines = list(map(hough_transform, masked_image))

def draw_lines(image, lines, color = [255, 0, 0], thickness = 2):
"""
Draw lines onto the input image.
Parameters:
image: An np.array compatible with plt.imshow.
lines: The lines we want to draw.
color (Default = red): Line color.
thickness (Default = 2): Line thickness.
"""
image = np.copy(image)
for line in lines:
for x1,y1,x2,y2 in line:
cv2.line(image, (x1, y1), (x2, y2), color, thickness)
return image

line_images = []
for image, lines in zip(test_images, hough_lines):
line_images.append(draw_lines(image, lines))

#list_images(line_images)

def average_slope_intercept(lines):
"""
Find the slope and intercept of the left and right lanes of each image.
```

```
Parameters:
lines: The output lines from Hough Transform.
"""
left_lines    = [] #(slope, intercept)
left_weights  = [] #(length,)
right_lines   = [] #(slope, intercept)
right_weights = [] #(length,)

for line in lines:
for x1, y1, x2, y2 in line:
if x1 == x2:
continue
slope = (y2 - y1) / (x2 - x1)
intercept = y1 - (slope * x1)
length = 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):
"""
Converts the slope and intercept of each line into pixel points.
Parameters:
y1: y-value of the line's starting point.
y2: y-value of the line's end point.
line: The slope and intercept of the line.
"""
if line is None:
return None
slope, intercept = line
x1 = int((y1 - intercept)/slope)
x2 = int((y2 - intercept)/slope)
y1 = int(y1)
y2 = int(y2)
return ((x1, y1), (x2, y2))
```

```
def lane_lines(image, lines):
"""
Create full lenght lines from pixel points.
Parameters:
image: The input test image.
lines: The output lines from Hough Transform.
"""
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_line = pixel_points(y1, y2, right_lane)
return left_line, right_line


def draw_lane_lines(image, lines, color=[255, 0, 0], thickness=12):
"""
Draw lines onto the input image.
Parameters:
image: The input test image.
lines: The output lines from Hough Transform.
color (Default = red): Line color.
thickness (Default = 12): Line thickness.
"""
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, 1.0, line_image, 1.0, 0.0)


lane_images = []
for image, lines in zip(test_images, hough_lines):
lane_images.append(draw_lane_lines(image, lane_lines(image, lines)))


#list_images(lane_images)

#Import everything needed to edit/save/watch video clips
from moviepy import *
from IPython.display import HTML
from IPython.display import Image
```

```
def frame_processor(image):
"""
Process the input frame to detect lane lines.
Parameters:
image: Single video frame.
"""
color_select = HSL_color_selection(image)
gray         = gray_scale(color_select)
smooth       = gaussian_smoothing(gray)
edges        = canny_detector(smooth)
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):
"""
Read input video stream and produce a video file with detected lane lines.
Parameters:
test_video: Input video.
output_video: A video file with detected lane lines.
"""
input_video = VideoFileClip(os.path.join('test_videos', test_video),
audio=False)
processed = input_video.fl_image(frame_processor)
processed.write_videofile(os.path.join('output_videos', output_video),
audio=False)
```

5.RESULT

5. RESULT

Screenshot 5.1 After application

Screenshot 5.2 Road lane line

Screenshot 5.3 Error Detection

Screenshot 5.4 Region Of Interest

6.TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.
Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.3 TEST CASES

6.3.1 CLASSIFICATION System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

7.CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

When we drive, we use our eyes to decide where to go. The lines on the road that show us where the lanes are act as our constant reference for where to steer the vehicle. Naturally, one of the first things we would like to do in developing a self-driving vehicle is to automatically detect lane lines using an algorithm. The road detection region of interest (ROI), must be flexible. When driving up or down a steep incline, the horizon will change and no longer be a product of the proportions of the frame. This is also something to consider for tight turns and bumper to bumper traffic. This project is entirely based on image processing and road detection in self-driving vehicles in which has a great scope in future. We have completed the entire implementation using specific algorithms to detect the road clearly. If the people's thought hasn't changed about the self-driving cars being safe, these cars are already safe and are becoming safer. Only if they believe and give a try to technology, they get to enjoy the luxury of computerized driving.

7.2 FUTURE SCOPE

This model can be updated and tuned with more efficient mathematical modelling, whereas the classical OpenCV approach is limited and no upgrade is possible as the approach is not efficient It is unable to give accurate results on the roads which do not have clear markings present on the roads. Also it cannot work for all climatic conditions This technology is increasing the number of applications such as traffic control, traffic monitoring, traffic flow, security etc.

8.BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 REFERENCES

1. General Road Detection From A Single Image, TIP-05166-2009, ACCEPTED, Hui Kong , Member, IEEE, Jean-Yves Audibert, and Jean Ponce , Fellow, IEEE Willow Team, Ecole Normale Superieure / INRIA / CNRS, Paris, France Imagine team, Ecole des Ponts ParisTech, Paris, France.

2. J. C. McCall and M. M. Trivedi, "Video based lane estimation and tracking for driver assistance: Survey, system, and evaluation," IEEE Trans. on Intelligent Transportation Systems, pp. 20–37, 2006.

3. K.-Y. Chiu and S.-F. Lin, "Lane detection using color-based segmentation," IEEE Intelligent Vehicles Symposium, 2005. 1

4. H. Kong, J.-Y. Audibert, and J. Ponce, "Vanishing point detection for road detection," CVPR, 2009.

5. Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," Image and Vision Computing, pp. 269–280, 2004

6. A. Lookingbill, J. Rogers, D. Lieb, J. Curry, and S. Thrun, "Reverse optical flow for selfsupervised adaptive autonomous robot navigation," IJCV, vol. 74, no. 3, pp. 287–302, 2007.

7. A. Broggi, C. Caraffi, R. I. Fedriga, and P. Grisleri, "Obstacle detection with stereo vision for off-road vehicle navigation," IEEE International Workshop on Machine Vision for Intelligent Vehicles, 2005.

8. J. Sparbert, K. Dietmayer, and D. Streller, "Lane detection and street type classification using laser range images," IEEE Proceedings in Intelligent transportation Systems, pp. 456– 464, 2001.

9. J. B. Southhall and C. Taylor, "Stochastic road shape estimation," ICCV, pp. 205–212, 2001.

8.3 GITHUB LINK

9. PAPER PUBLICATION

10. CERTIFCATES

10. CERTIFICATES

International Journal For Advanced Research In Science & Technology

A peer reviewed international journal www.ijarst.in

ISSN: 2457-0362

Certificate

This is to Certify that Prof./Dr./Mr./Ms./ M. SAI VENKAT SANJAY from CMR Technical Campus, Medchal, Hyderabad, Telangana, India.Presented a Paper Entitled "A NOVEL ROAD LANE LINE DETECTION USING COMPUTER VISION TECHNIQUE" In the Organizing Commitee of Asian Science Research

Published in International Journal For Advanced Researchs In Science & Technology. Research (IJARST),Vol-13,Issue-04 Apr-2023

Editor In Chief
N.C KARNAKARAN


International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
IJARST
ISSN: 2457-0362
Certificate
This is to Certify that Prof./Dr./Mr./Ms./ K. AKASH from CMR Technical Campus, Medchal, Hyderabad, Telangana, India.Presented a Paper Entitled "A NOVEL ROAD LANE LINE DETECTION USING COMPUTER VISION TECHNIQUE" In the Organizing Commitee of Asian Science Research
Published in International Journal For Advanced Researchs In Science & Technology. Research (IJARST),Vol-13,Issue-04 Apr-2023
Google scholar
ROAD
DIRECTORY OF OPEN ACCESS SCHOLARLY RESOURCES
NISCAIR
Editor In Chief
N.C KARNAKARAN


International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
IJARST
ISSN: 2457-0362
Certificate
This is to Certify that Prof./Dr./Mr./Ms./ K. ABHILASH from CMR Technical Campus, Medchal, Hyderabad, Telangana, India.Presented a Paper Entitled "A NOVEL ROAD LANE LINE DETECTION USING COMPUTER VISION TECHNIQUE" In the Organizing Commitee of Asian Science Research
Published in International Journal For Advanced Researchs In Science & Technology. Research (IJARST),Vol-13,Issue-04 Apr-2023
Google scholar
ROAD
DIRECTORY OF OPEN ACCESS SCHOLARLY RESOURCES
NISCAIR
Editor In Chief
N.C KARNAKARAN

International Journal For Advanced Research In Science & Technology

A peer reviewed international journal www.ijarst.in

ISSN: 2457-0362

Certificate

This is to Certify that Prof./Dr./Mr./Ms./ G. VIJAY KUMAR from CMR Technical Campus, Medchal, Hyderabad, Telangana, India.Presented a Paper Entitled "A NOVEL ROAD LANE LINE DETECTION USING COMPUTER VISION TECHNIQUE" In the Organizing Commitee of Asian Science Research

Published in International Journal For Advanced Researchs In Science & Technology. Research (IJARST),Vol-13,Issue-04 Apr-2023

Editor In Chief
N.C KARNAKARAN