KARNATAKA LAW SOCIETY'S

GOGTE INSTITUE OF TECHNOLOGY

UDYAMBAG, BELAGAVI – 590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

(APPROVED BY AICTE, NEW DELHI)

Course Activity Report

Lane Detection

Submitted in the partial fulfillment for the academic requirement of

VI Semester B.E.

In

Artificial Intelligence and Machine Learning - Laboratory

Submitted by:

Nikita Kodkany	2GI18CS082
Nischal Kanishk	2GI18CS083
Prathamesh Koranne	2GI18CS095
Prithvi Gudodagi	2GI18CS102

Under the Guidance of:

Dr. Umesh Kulkarni

Department of Computer Science and Engineering

KLS Gogte Institute of Technology, Belagavi

2020-2021

CERTIFICATE

This is to certify that the Seminar entitled “Lane Detection” is a bona fide record of the Seminar work done by Nikita Kodkany (2GI18CS082), Nischal Kanishk (2GI18CS083), Prathamesh Koranne (2GI18CS095) and Prithvi Gudodagi (2GI18CS102) under my supervision and guidance, in partial fulfillment of the requirements for the Outcome Based Education Paradigm in Computer Science and Engineering from Gogte Institute of Technology for the academic year 2020-2021.

Dr. Umesh Kulkarni
Asst. Professor
Computer Science and Engineering

Dr. Vijayraj Purohit
Professor and Head
Computer Science and Engineering

Place: KLS Gogte Institute of Technology
Date: 24 May 2021

Marks Allocation:

	Batch No. : 15					
1.	Project Title: Implement Recursive Descent Parsing in C	Marks Range	USN			
			2GI18CS 082	2GI18CS 083	2GI18CS 095	2GI18CS 102
2.	Problem statement (PO2)	0-1				
3.	Objectives of Defined Problem statement (PO1,PO2)	0-2				
4.	Design / Algorithm/Flowchart/Methodology (PO3)	0-3				
5.	Implementation details/Function/Procedures/Classes and Objects (Language/Tools) (PO1,PO3,PO4,PO5)	0-4				
6.	Working model of the final solution (PO3,PO12)	0-5				
7.	Report and Oral presentation skill (PO9,PO10)	0-5				
	Total	20				

*** 20 marks is converted to 10 marks for CGPA calculation**

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
2. **Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and Engineering sciences.
3. **Design/Development of solutions:**Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:**Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:**Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need

for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

8. Individual and team work: Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

8. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

8. Project management and finance: Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

ACKNOWLEDGEMENT

This group feels greatly indebted to Computer Science and Engineering Department, for the opportunity given us to undertake this "*Lane Detection*" Project. This Project includes thoughts and contribution of many individuals. And we wish to express our sincere appreciation and gratitude to them.

First and foremost, we want to extend entirely our gratitude to our lecturer Dr. Umesh Kulkarni for sharing her knowledge and profound wisdom with us. We appreciate all her comments and suggestions, which are incorporated into this project. We would also like to express our gratitude towards and group members. Without their help, support, and encouragement, this project would never had been completed.

In our respect, this project is an outcome of the learning experience we have shared with our fellow students. We dedicate this Project to all our fellow engineering students.

NIKITA KODKANY

NISCHAL KANISHK

PRATHAMESH KORANNE

PRITHVI GUDODAGI

Contents

Abstract

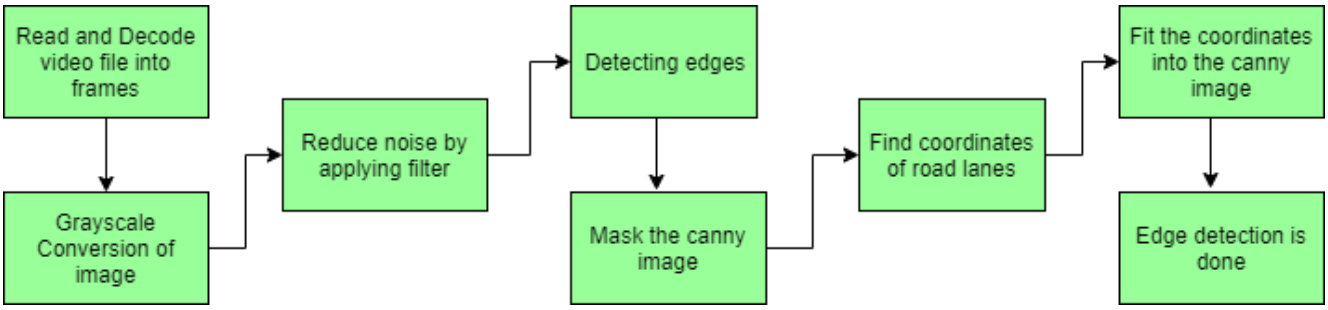
To implement real-time lane detection in a video. There are multiple ways we can perform this task. We can use learning-based approaches, such as training a deep learning model on an annotated video dataset, or use a pre-trained model. However, there are simpler methods to perform lane detection as well. We shall do it without using any deep learning model by using the popular OpenCV library in Python.

Introduction

Autonomous Driving Car is one of the most disruptive innovations in AI. Fuelled by Deep Learning algorithms, they are continuously driving our society forward and creating new opportunities in the mobility sector. An autonomous car can go anywhere a traditional car can go and does everything that an experienced human driver does. But it's very essential to train it properly. One of the many steps involved during the training of an autonomous driving car is lane detection, which is the preliminary step. Today, we are going to learn how to perform lane detection using videos.

Theory

Lane detection involves the following steps:



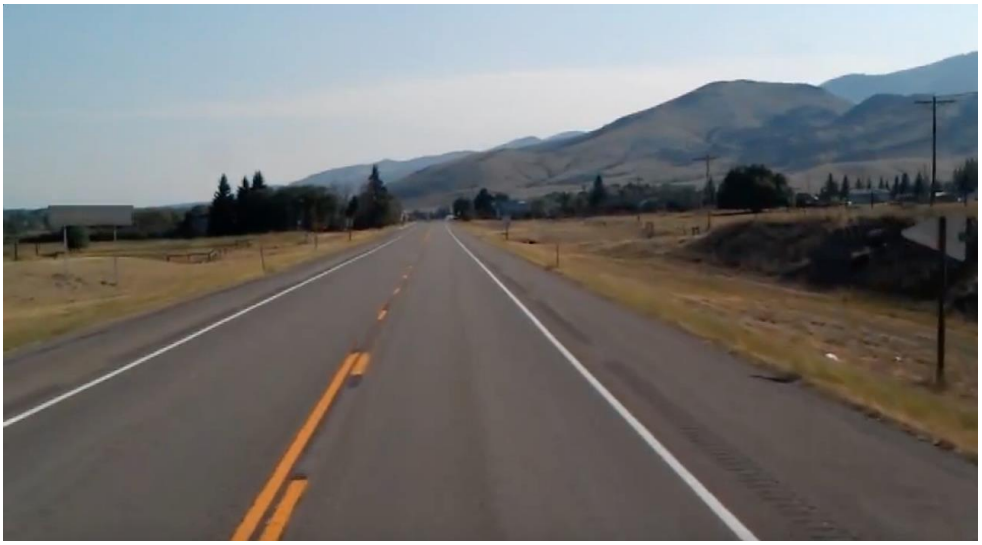
- **Capturing and decoding video file:** We will capture the video using VideoCapture object and after the capturing has been initialized every video frame is decoded (i.e. converting into a sequence of images).
- **Grayscale conversion of image:** The video frames are in RGB format, RGB is converted to grayscale because processing a single channel image is faster than processing a three-channel coloured image.
- **Reduce noise:** Noise can create false edges, therefore before going further, it's imperative to perform image smoothening. Gaussian filter is used to perform this process.
- **Canny Edge Detector:** It computes gradient in all directions of our blurred image and traces the edges with large changes in intensity. For more explanation, please go through this article: Canny Edge Detector
- **Region of Interest:** This step is to take into account only the region covered by the road lane. A mask is created here, which is of the same dimension as our road image. Furthermore, bitwise AND operation is performed between each pixel of our canny image and this mask. It ultimately masks the canny image and shows the region of interest traced by the polygonal contour of the mask.
- **Hough Line Transform:** The Hough Line Transform is a transform used to detect straight lines. The Probabilistic Hough Line Transform is used here, which gives output as the extremes of the detected lines

Program

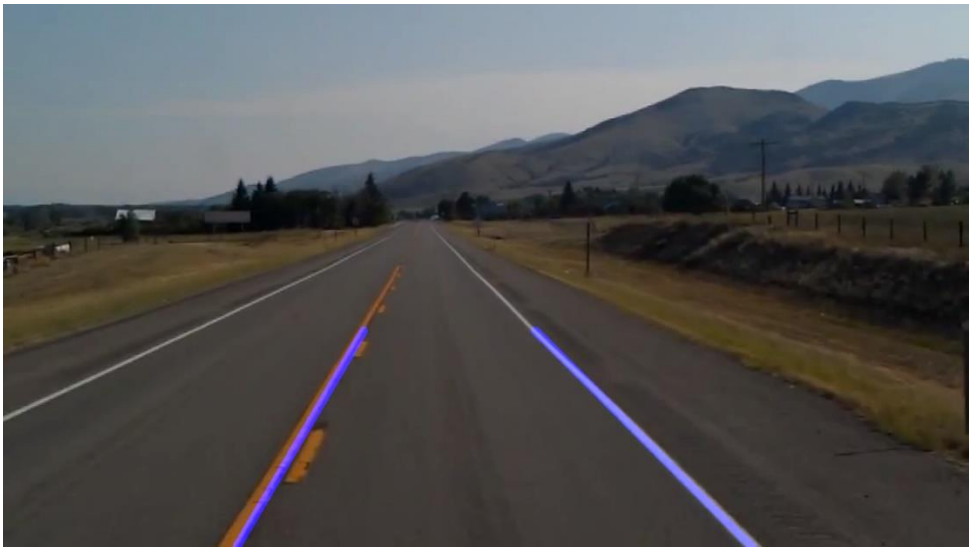
```
# Import the required libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt

def canny_edge_detector(image):

      # Convert the image color to grayscale
      gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

      # Reduce noise from the image
      blur = cv2.GaussianBlur(gray_image, (5, 5), 0)
      canny = cv2.Canny(blur, 50, 150)
      return canny

def region_of_interest(image):

      height = image.shape[0]
      polygons = np.array([
            [(200, height), (1100, height), (550, 250)]
            ])
      mask = np.zeros_like(image)

      # Fill poly-function deals with multiple polygon
      cv2.fillPoly(mask, polygons, 255)

      # Bitwise operation between canny image and mask image
      masked_image = cv2.bitwise_and(image, mask)
      return masked_image

def create_coordinates(image, line_parameters):
      slope, intercept = line_parameters
      y1 = image.shape[0]
      y2 = int(y1 * (3 / 5))
      x1 = int((y1 - intercept) / slope)
      x2 = int((y2 - intercept) / slope)
      return np.array([x1, y1, x2, y2])

def average_slope_intercept(image, lines):
      left_fit = []
      right_fit = []
      for line in lines:
            x1, y1, x2, y2 = line.reshape(4)

            # It will fit the polynomial and the intercept and slope
            parameters = np.polyfit((x1, x2), (y1, y2), 1)
            slope = parameters[0]
            intercept = parameters[1]
            if slope < 0:
                  left_fit.append((slope, intercept))
            else:
                  right_fit.append((slope, intercept))

      left_fit_average = np.average(left_fit, axis = 0)
      right_fit_average = np.average(right_fit, axis = 0)
      left_line = create_coordinates(image, left_fit_average)
      right_line = create_coordinates(image, right_fit_average)
      return np.array([left_line, right_line])
```

```
def display_lines(image, lines):
      line_image = np.zeros_like(image)
      if lines is not None:
            for x1, y1, x2, y2 in lines:
                  cv2.line(line_image, (x1, y1), (x2, y2), (255, 0, 0), 10)
      return line_image

# Path of dataset directory
cap = cv2.VideoCapture("./test2.mp4")
while(cap.isOpened()):
      _, frame = cap.read()
      canny_image = canny_edge_detector(frame)
      cropped_image = region_of_interest(canny_image)

      lines = cv2.HoughLinesP(cropped_image, 2, np.pi / 180, 100,
                                          np.array([]), minLineLength = 40,
                                          maxLineGap = 5)

      averaged_lines = average_slope_intercept(frame, lines)
      line_image = display_lines(frame, averaged_lines)
      combo_image = cv2.addWeighted(frame, 0.8, line_image, 1, 1)
      cv2.imshow("results", combo_image)

      # When the below two will be true and will press the 'q' on
      # our keyboard, we will break out from the loop

      # # wait 0 will wait for infinitely between each frames.
      # 1ms will wait for the specified time only between each frames
      if cv2.waitKey(1) & 0xFF == ord('q'):
            break

# close the video file
cap.release()

# destroy all the windows that is currently on
cv2.destroyAllWindows()
```

Input



Output



Conclusion

We covered a simple technique for lane detection. We did not use any model or complex image features. Instead, our solution was purely based on certain image pre-processing operations.

However, there are going to be many scenarios where this solution will not work. For example, when there will be no lane markings, or when there is too much of traffic on the road, this system will fail. There are more sophisticated methods to overcome such problems in lane detection.

Reference

1. Ben Coppin, Artificial Intelligence Illuminated, Jones and Bartlett, 2004
2. Tom M. Mitchell, “Machine Learning”, Mcgraw-Hill Education (Indian Edition),2013.