Capstone Project On Lane Detection

Name: **Shreya Shrivastava**

Course: **Artificial Intelligence And Machine Learning**

Batch Four

Duration: **12 Months**

Problem statement: **To perform lane detection**

Prerequisites:

The libraries as well as things required in order for the program to work:

I. **Python 3.6** : The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic. Second option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url : https://www.anaconda.com/download/

II. **OpenCV** **:** OpenCV can be downloaded from the following url: https://sourceforge.net/projects/opencvlibrary/. It is strongly recommended to download OpenCV in a virtual environment.

III. **ADDITIONAL PACKAGES** **:** You will also need to download and install numpy after you install either python or anaconda from the steps above. If you have chosen to install python 3.6,then run the following commands in command prompt/terminal to install this package :

NUMPY: pip install -U numpy

If using Anaconda then run the following commands in anaconda prompt to install these packages:

NUMPY: conda install -c anaconda numpy

IV. METHODS USED:

A. CANNY EDGE DETECTION

B. HOUGH TRANSFORM

THE PROJECT :

1. Importing the libraries and writing a few helper functions to do basic image pre-processing.

```
import cv2
import numpy as np

def grey(image):
    image = np.asarray(image)
    return cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)


def gauss(image):
    return cv2.GaussianBlur(image, (5, 5), 0)


def canny(image):
    edges = cv2.Canny(image,50,150)
    return edges
def region(image):
    height, width = image.shape

    triangle = np.array([
                        [(100, height), (475, 325), (width, height)]
                        ])
    mask = np.zeros_like(image)
    mask = cv2.fillPoly(mask, triangle, 255)
    mask = cv2.bitwise_and(image, mask)
    return mask
```

2. The display_lines function draws the lanes.
The average function optimizes the markings
The make_points function determine the equation of the line

```
def display_lines(image, lines):
    lines_image = np.zeros_like(image)
    #make sure array isn't empty
    if lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line
            #draw lines on a black image
            cv2.line(lines_image, (x1, y1), (x2, y2), (255, 0, 0), 10)
    return lines_image
def average(image, lines):
    left = []
    right = []
    for line in lines:
        print(line)
        x1, y1, x2, y2 = line.reshape(4)
        #fit line to points, return slope and y-int
        parameters = np.polyfit((x1, x2), (y1, y2), 1)
        print(parameters)
        slope = parameters[0]
        y_int = parameters[1]
        #lines on the right have positive slope, and lines on the left have neg slope
        if slope < 0:
            left.append((slope, y_int))
        else:
            right.append((slope, y_int))
    #takes average among all the columns (column0: slope, column1: y_int)
    right_avg = np.average(right, axis=0)
    left_avg = np.average(left, axis=0)
    #create lines based on averages calculates
    left_line = make_points(image, left_avg)
    right_line = make_points(image, right_avg)
    return np.array([left_line, right_line])
def make_points(image, average):
    print(average)
    slope, y_int = average
    y1 = image.shape[0]
    y2 = int(y1 * (3/5))
    #determine algebraically
    x1 = int((y1 - y_int) // slope)
    x2 = int((y2 - y_int) // slope)
    return np.array([x1, y1, x2, y2])
```

3. Integrating the above helper functions to implement lane detection.

```
video = "/Users/shreyashrivastava/Desktop/sample.mp4"
cap = cv2.VideoCapture(video)
print("here")
num = 0
while(cap.isOpened()):
    _, frame = cap.read()
    print(num)
    num += 1

    gaus = gauss(frame)
    edges = cv2.Canny(gaus,50,150)
    isolated = region(edges)
    lines = cv2.HoughLinesP(isolated, 2, np.pi/180, 50, np.array([]), minLineLength=40, maxLineGap=5)
    averaged_lines = average(frame, lines)
    black_lines = display_lines(frame, averaged_lines)
    lanes = cv2.addWeighted(frame, 0.8, black_lines, 1, 1)
    cv2.imshow("frame", lanes)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

5. Final Results.