Self-Driving Vehicle: Lane Detection

Name - Amith Korada

Course - AI & ML
(Batch - 4)

Duration - 12 Months

Problem Statement- Implement an auto-encoder model that takes the image as input and outputs the images with the lanes marked

Prerequisites -

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has the latest version of python. The following URL https://www.python.org/downloads/ can be referred to as download python.

The second and easier 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/ You will also need to download and install the below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run the below commands in command prompt/terminal to install these packages pip install -U sci-kit-learn pip install NumPy pip install scipy if you have chosen to install anaconda then run the below commands in anaconda prompt to install these packages conda install -c sci-kit-learn conda install -c anaconda numpy conda install -c anaconda scipy.

1. Importing necessary libraries -

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

2. Canny Edge detection -

```
def canny_edge_detector(image):
    gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    blur = cv2.GaussianBlur(gray_image, (5, 5), 0)
    canny = cv2.Canny(blur, 50, 150)
    return canny
```

3. Masking the canny image after finding the region of interest-

```
def region_of_interest(image):
    height = image.shape[0]
    polygons = np.array([[(200, height), (1100, height), (550, 250)]])
    mask = np.zeros_like(image)
    cv2.fillPoly(mask, polygons, 255)
    masked_image = cv2.bitwise_and(image, mask)
    return masked_image
```

4. Finding coordinates of the road lane-

```
def create_coordinates(image, line_parameters):
    try:
        slope, intercept = line_parameters
    except TypeError:
        slope, intercept = 0,0

    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])
```

5. Differentiating the left and right lanes-

```
def average_slope_intercept(image, lines):
    left_fit = []
    right_fit = []
    for line in lines:
        x1, y1, x2, y2 = line.reshape(4)
        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)
    mid_line = np.array((left_line + right_line)/2).astype('int32')

    return np.array([left_line, right_line, mid_line])
```

6. Marking the lanes on the input image-

```
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
```

7. Loading the input video and detecting the straight lines in the image using Houghline method and calling the above functions-

```
cap = cv2.VideoCapture("test.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)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()

cv2.destroyAllWindows()
```