DIP MINI PROJECT (ASSIGNMENT 2)

Lane Detection

ADITI ADHIKARY PES1UG19EC013 ‘A’ Section

AISHNA SHAH PES1UG19EC022 ‘A’ Section

SHREYA V PES1UG19EC288 ‘E’ Section

Problem Statement

The task that we wish to perform is that of real-time lane detection in a video, we are doing so using the popular OpenCV Library in Python for Image processing.

A lane is part of a roadway (carriageway) that is designated to be used by a single line of vehicles, to control and guide drivers and reduce traffic conflicts

Lane detection is a critical component of self-driving cars and autonomous vehicles. It is one of the most important research topics for driving scene understanding. Once lane positions are obtained, the vehicle will know where to go and avoid the risk of running into other lanes or getting off the road. This can prevent the driver/car system from drifting off the driving lane.

Theoretical details and Algorithm

The core part of the detection is to correctly extract the lines of the road from all the rest of the images.

We can do this applying the Hsv color detection. In this way we can detect object by their colors, as the lines of a road can be only yellow or white, we extract the part of the images that contains either of these two colors only.

Once we have the mask we find the edges, we **use the 'Hough transform method'.**

Hough Transform is a technique to detect any shape that can be represented mathematically. For example, it can detect shapes like rectangles, circles, triangles, or lines. We are interested in detecting lane markings that can be represented as lines.

One of the most important features of this method is that **can detect lines even when some part of it is missing**. And this comes really useful in the road when we have dashed lines, or when for some reason some part of the line is not visible.

1. We start by importing the library and loading our video.
2. OpenCV provides cv2.gaussianblur() function to apply Gaussian Smoothing on the input source image
3. Convert RGB to HSV. [*HSV (hue, saturation, value) colorspace is a model to represent the colorspace similar to the RGB color model. As hue channel models the color type, it is very useful in image processing tasks that need to segment objects based on its color. Variation of the saturation goes from unsaturated to represent shades of grey and fully saturated (no white component). Value channel describes the brightness or the intensity of the color. Next image shows the HSV cylinder.]* Set the upper and lower SV Boundaries for the color desired (*here yellow).*
4. Create the frame Mask. [***Frame mask** is nothing but a NumPy array. When we want to apply a mask to an image, we simply change the pixel values of the desired region in that image to 0, or 255, or any other number.]*

5. Use the **Canny**() method of **cv2** library to detect edges in an image.*[Canny Edge Detection]*
6. Use cv2.houghlines(), in order to detect the lines of the lane on either side of the path.
7. Using cv2.imshow(), display the output window(s) in order to observe the Lane detection.

CODE:

```
import cv2

import numpy as np

# Create a VideoCapture object and read from input file

video = cv2.VideoCapture(r'C:\Users\aditi\Documents\Semester
5\road_car_view_Trim1.mp4')

# Check if camera opened successfully

if (video.isOpened()== False):

    print("Error opening video file")

while True:

    ret,orig_frame =video.read()

    if ret == True:

        frame = cv2.GaussianBlur(orig_frame, (5, 5), 0)

        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        low_yellow = np.array([10, 100, 140])

        up_yellow = np.array([48, 255, 255])

        mask = cv2.inRange(hsv, low_yellow, up_yellow)

        edges = cv2.Canny(mask, 75, 150)

        lines = cv2.HoughLinesP(edges, 1, np.pi/180, 50, maxLineGap=50)
```

```
            if lines is not None:
                for line in lines:
                    x1, y1, x2, y2 = line[0]
                    cv2.line(frame, (x1, y1), (x2, y2), (0, 255, 0), 5)
                cv2.imshow("frame", frame)
                cv2.imshow("edges", edges)
                # Press Q on keyboard to exit
                if cv2.waitKey(25) & 0xFF == ord('q'):
                    break
        # Break the loop
        else:
            break
# When everything done, release
# the video capture object
video.release()
# Closes all the frames
cv2.destroyAllWindows()
```

RESULT:

As displayed in the output video, the lanes are detected and this is indicated by the green line produced as an effect of the Hough transform function, cv2.houghlines() used for the same.