"KUNTAL CHAUDHURY"

"LANE LINE DETECTION"

```
!git clone https://github.com/chuanenlin/lane-detector.git
%cd lane-detector
%ls
```

```
Cloning into 'lane-detector'...
remote: Enumerating objects: 17, done.
remote: Total 17 (delta 0), reused 0 (delta 0), pack-reused 17
Unpacking objects: 100% (17/17), 6.31 MiB | 11.85 MiB/s, done.
/content/lane-detector/lane-detector/lane-detector
checkpoint1.py  checkpoint3.py  checkpoint5.py  input.mp4  solution.py
checkpoint2.py  checkpoint4.py  detector.py     README.md
```

Step 1: Importing Libraries

```
import cv2 as cv
from google.colab.patches import cv2_imshow
import numpy as np


def do_canny(frame):
    gray = cv.cvtColor(frame, cv.COLOR_RGB2GRAY)
    blur = cv.GaussianBlur(gray, (5, 5), 0)
    canny = cv.Canny(blur, 50, 150)
    return canny


def do_segment(frame):
    height = frame.shape[0]
    polygons = np.array([
                            [(0, height), (800, height), (380, 290)]
                        ])
    mask = np.zeros_like(frame)
    cv.fillPoly(mask, polygons, 255)
    segment = cv.bitwise_and(frame, mask)
    return segment


def calculate_lines(frame, lines):
    left = []
    right = []
    for line in lines:
        x1, y1, x2, y2 = line.reshape(4)
        parameters = np.polyfit((x1, x2), (y1, y2), 1)
        slope = parameters[0]
        y_intercept = parameters[1]
        if slope < 0:
            left.append((slope, y_intercept))
        else:
            right.append((slope, y_intercept))
    left_avg = np.average(left, axis = 0)
    right_avg = np.average(right, axis = 0)
    left_line = calculate_coordinates(frame, left_avg)
    right_line = calculate_coordinates(frame, right_avg)
    return np.array([left_line, right_line])

def calculate_coordinates(frame, parameters):
    slope, intercept = parameters
    y1 = frame.shape[0]
    y2 = int(y1 - 150)
    x1 = int((y1 - intercept) / slope)
    x2 = int((y2 - intercept) / slope)
    return np.array([x1, y1, x2, y2])



def visualize_lines(frame, lines):
    lines_visualize = np.zeros_like(frame)
    if lines is not None:
```

```
        for x1, y1, x2, y2 in lines:
            cv.line(lines_visualize, (x1, y1), (x2, y2), (0, 255, 0), 5)
    return lines_visualize
```

Master Function

```
def lane_detector (frame):
    canny = do_canny(frame)
    segment = do_segment(canny)
    hough = cv.HoughLinesP(segment, 2, np.pi / 180, 100, np.array([]), minLineLength = 100, maxLineGap = 50)
    lines = calculate_lines(frame, hough)
    lines_visualize = visualize_lines(frame, lines)
    output = cv.addWeighted(frame, 0.9, lines_visualize, 1, 1)
    return output
```

```
import cv2
cap = cv2.VideoCapture('input.mp4')
ret, frame = cap.read()
frame_height, frame_width, _ = frame.shape
out = cv2.VideoWriter('output.avi',cv2.VideoWriter_fourcc('M','J','P','G'), 10, (frame_width,frame_height))
print("Processing Video...")
while cap.isOpened():
  ret, frame = cap.read()
  if not ret:
    out.release()
    break
  output = lane_detector(frame)
  out.write(output)
out.release()
print("Done processing video")
```

```
Processing Video...
Done processing video
```