LANE DETECTION FOR AUTONOMOUS VEHICLES

AIM:

To detect and highlight lanes on a road in an input image using computer vision techniques.

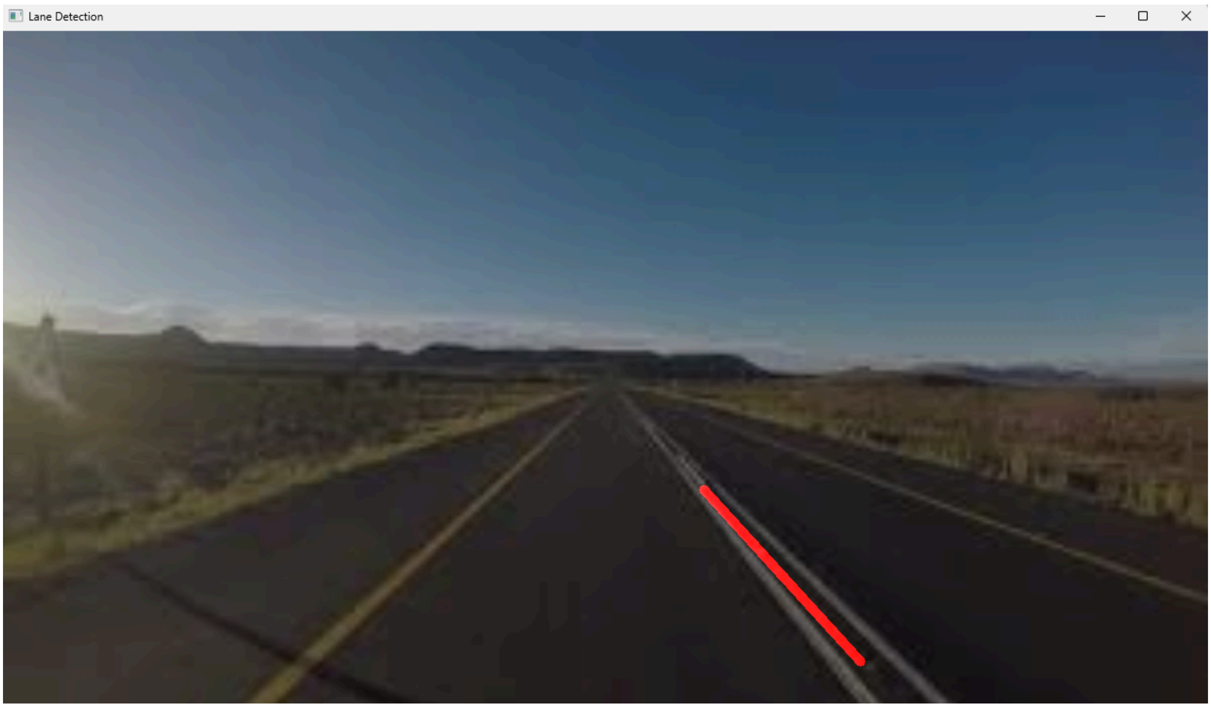
ALGORITHM:

- Import required libraries(cv2 and numpy)
- Preprocess the image by applying grayscale and blur
- Detect edges and region of interest to detect lanes.
- Use the Hough Transform to detect lines in the edge-detected image within the ROI.
- Separate the detected lines into left and right lanes based on their slopes.
- For each set of lines calculate the coordinates for displaying and draw the detected lane lines separately for left and right lanes with different colours.
- Display the final image/video with lane lines

PROGRAM CODE:

```
        import cv2
import numpy as np

def detect_lanes(image_path):
    # Read the image
    img = cv2.imread(image_path)
    if img is None:
        raise FileNotFoundError(f"Image file '{image_path}' not found.")

    # Resize the image for better visualization
    img = cv2.resize(img, (1280, 720))

    # Convert to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Apply Gaussian blur
    blur = cv2.GaussianBlur(gray, (5, 5), 0)

    # Apply Canny edge detection
    edges = cv2.Canny(blur, 50, 150)
    # Define region of interest (ROI) using a trapezoid shape
    height, width = edges.shape[:2]
    vertices = np.array([[
        (0, height),
        (width * 1/3, height * 2/3),
        (width * 2/3, height * 2/3),
        (width, height),
    ]], dtype=np.int32)
    mask = np.zeros_like(edges)
    cv2.fillPoly(mask, vertices, 255)
    masked_edges = cv2.bitwise_and(edges, mask)

    # Perform Hough Transform to detect lines
    lines = cv2.HoughLinesP(masked_edges, rho=1, theta=np.pi/180, threshold=20,
minLineLength=20, maxLineGap=300)

    # Create a blank image to draw lines on
    line_image = np.zeros_like(img)

    # Draw detected lines with an emphasis on the parallel lines
    if lines is not None:
        left_lane_lines = []
        right_lane_lines = []
        for line in lines:
            x1, y1, x2, y2 = line[0]
            slope = (y2 - y1) / (x2 - x1) if (x2 != x1) else None
            if slope is None:
                continue
            if slope < -0.5:
                left_lane_lines.append(line[0])
            elif slope > 0.5:
                right_lane_lines.append(line[0])

        # Fit lines (y = mx + b) to get average left and right lanes
        left_lane = np.mean(left_lane_lines, axis=0, dtype=np.int32)
        right_lane = np.mean(right_lane_lines, axis=0, dtype=np.int32)

        # Draw left lane line
        if len(left_lane_lines) > 0:
            cv2.line(line_image, (left_lane[0], left_lane[1]), (left_lane[2], left_lane[3]), (0, 0, 255),
10)

        # Draw right lane line
        if len(right_lane_lines) > 0:
            cv2.line(line_image, (right_lane[0], right_lane[1]), (right_lane[2], right_lane[3]), (0, 0,
255), 10)

    # Overlay the detected lanes on the original image
    output_image = cv2.addWeighted(img, 0.8, line_image, 1.0, 0.0)

    # Display the final output image
    cv2.imshow('Lane Detection', output_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Example usage
image_path = r"C:\Users\student\Downloads\images (1).jpg"
detect_lanes(image_path)
```

OUTPUT:


Lane Detection