Lane Boundary Detection and Line Fit Analysis Report

Manish Patel
2024MCS2460

February 18, 2025

Introduction

The objective of this assignment is to implement a lane boundary detection system using image processing techniques without relying on built-in edge detection and line detection functions. Additionally, an analysis of line fit quality using intersection-based evaluation is performed.

This report covers two main tasks:

- **Task 1: Lane Boundary Detection** – Detecting lane boundaries in road images.
- **Task 2: Intersection-Based Line Fit Analysis** – Analyzing line intersections in grass field images.

Task 1: Lane Boundary Detection

Approach and Methodology

Step 1: Grayscale Conversion

A custom grayscale conversion function is implemented using the formula:

$$\text{Gray}(x, y) = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B$$

This reduces the image to a single channel, emphasizing intensity information.

Step 2: Gaussian Blur

A **5x5 Gaussian kernel** is used to smooth the image and reduce noise:

$$\text{Kernel} = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

The image is padded using the **reflect** method, and convolution is applied using nested loops.

Step 3: Edge Detection (Sobel Operator)

Sobel operators in **x** and **y** directions are applied:

$$\text{Sobel}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{Sobel}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Gradient magnitude is computed as:

$$\text{Gradient Magnitude} = \sqrt{G_x^2 + G_y^2}$$

Thresholding is applied with **low (50)** and **high (200)** thresholds:

- **Strong edges (255)**: Above **high_threshold**.
- **Weak edges (128)**: Between **low_threshold** and **high_threshold**.

Step 4: Line Detection (Hough Transform)

OpenCV's HoughLinesP is used for probabilistic Hough Transform with parameters:

- $\rho = 1$ pixel.
- $\theta = \pi/180$.
- **Threshold** = 250.
- **Min Line Length** = 100 pixels.
- **Max Line Gap** = 50 pixels.

A **Custom Hough Transform** is also implemented but not used in final output. It accumulates votes for each (ρ, θ) and groups points into lines.

Step 5: Drawing Lane Lines

Detected lines are drawn on the original image in **green (0, 255, 0)**.

Results and Observations

- I accept that the system does not effectively detects lane boundaries.
- Performance degrades in cases of shadows, occlusions, or faded lane lines.
- **Gaussian blur** reduces noise, improving edge detection stability.
- **Sobel operator** substitutes Canny detection but is more sensitive to noise.
- **Hough Transform** is robust but sensitive to tuning parameters like **threshold, line length, and gap**.

Task 2: Intersection-Based Line Fit Analysis

Approach and Methodology

Step 1: Line Detection

Same grayscale, Gaussian blur, edge detection, and **Hough Line Detection** pipeline is applied to detect lines in grass field images.

Step 2: Intersection Computation

Intersection points of detected lines are computed using the line intersection formula:

$$\text{Intersection Point (px, py)} =$$

$$\frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

Valid intersections are within image bounds.

Step 3: Centroid Computation

Centroid of intersection points is computed as:

$$\text{Centroid}(x_c, y_c) = \left(\frac{\sum x_i}{n}, \frac{\sum y_i}{n} \right)$$

Step 4: Line Fit Quality (Sum of Distances)

Sum of distances from each intersection to the centroid is computed using:

$$\text{Distance} = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}$$

Results and Observations

- for part2 my code is taking too much time.
- for part2 i was failed to separate the grass images and other images. item my part 2 code runs on the every image which is wrong
- Poorly fitted lines result in **widespread intersections**, increasing distance sum.
- Intersection analysis is sensitive to **the number and accuracy of detected lines**.

Challenges and Limitations

Edge Detection

The Sobel operator is sensitive to noise and sometimes detects extra edges compared to the Canny detector.

Parameter Tuning

Hough line detection depends on choosing proper values for **threshold**, **line length**, and **gap**. Wrong values can miss lines or detect too many small lines.

Custom Hough Transform

The custom Hough Transform works well but is slow compared to OpenCV's built-in version, especially for large images.

Intersection Analysis

Intersection detection depends on correct line detection. Missing or wrong lines can give inaccurate intersection points.

Conclusion

- I tried to give my best to this assignment.
- In part1 most of the result images contains the detected lines.
- I have failed to detect the lines in the images which contains high brightness and high grass noise.
- i have used chatgpt sometimes, like in the case of some syntax errors or semantic errors etc.

References

- OpenCV Documentation: `https://docs.opencv.org/`