COL780/JRL780: Computer Vision

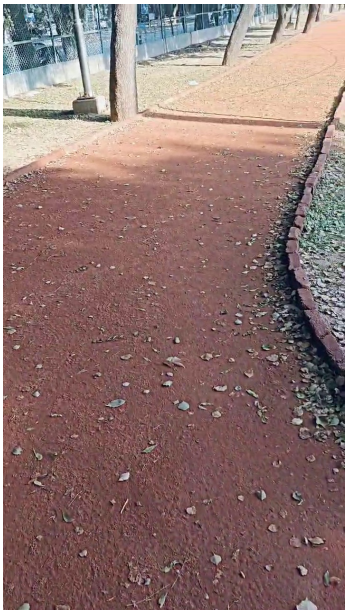
Assignment-1: Lane Boundary Detection

Deadline: XXth February, 2025

1 Introduction

Image preprocessing is a crucial first step in any computer vision system. It helps improve image quality by reducing noise, enhancing contrast, and highlighting important features. In this assignment, you will create and implement a lane detection algorithm using image processing techniques. This will give you practical experience in improving images and detecting lane boundaries for real-world applications.

(a) Sample Image 1



(b) Sample Image 2

2 Implementation Tasks

Various algorithms that have been discussed in class can be implemented to quantitatively perform the tasks mentioned below for a given image. Some examples include Canny edge detection, Hough Transform, and Contour detection.

2.1 Task 1: Draw Lane Boundaries or Curvature

Objective

Detect lane or track boundaries in the given images. Use edge detection algorithms such as Canny or Sobel to identify potential lane edges. Refine the detected edges using Hough Line Transform (for straight lines) or polynomial fitting (for curves). Visualize the detected boundaries on the original image by overlaying lines or curves.

Steps

1. **Preprocessing:** Resize, enhance contrast, and reduce noise for better edge detection.

2. **Edge Detection:** Apply an appropriate edge detection algorithm to highlight lane boundaries.
3. **Lane Boundary Detection:** Use line/curve detection methods to identify boundaries:
    - For straight lines, you can use the Hough Line Transform.
    - For curved tracks, you can apply polynomial fitting techniques.

Expected Output

The detected lane boundaries or curvature must be overlaid on the original image for visualization. Ensure that the output clearly shows the detected lines or curves.

2.2 Task 2: Intersection-Based Line Fit Analysis

Objective

Analyze the quality of detected lines in grass images by finding intersection points and measuring their distribution.

Methodology

1. Detect multiple lines present in the grass images.
2. Compute the intersection points between each pair of detected lines.
3. Determine the centroid of all intersection points.
4. Calculate the sum of distances of each intersection point from the centroid.
5. Use this sum as a quantitative measure of the line fit quality.

Expected Output

Provide a numerical score representing the quality of the line fit based on the computed distance sum.

3 Implementation Guidelines

- Restrictions: You are NOT allowed to use any inbuilt image processing functions, except for reading and writing images or visualization purposes.
- Individual Work: This is an individual assignment. If you have discussed the assignment with other students, mention their names in your report.
- Plagiarism Policy: Any form of plagiarism will result in an automatic zero for the assignment, along with stricter penalties based on the severity of the case.

Task-Specific Output Requirements

For each task, you are required to save outputs in a specific format as described below:

- **Task 1 (Lane Boundary Detection):**
    - Save the images with detected lane boundaries in a folder named `Task1_output_images`.
    - Each output image should clearly show the detected lane boundaries overlaid on the original image.
    - The code for this task must be executed as:

```
python3 main.py 1 <input_img_dir> <output_img_dir>
```

    - Example:

```
python3 main.py 1 ./input_images ./Task1_output_images
```

- **Task 2 (Intersection-Based Line Fit Analysis):**
  - Input to this task will be an image directory.
  - Generate a CSV file named `Task2.csv` with the following format:
    - **Column 1:** Name of the image file (e.g., `image1.jpg`).
    - **Column 2:** Quantitative line fit measure.
  - The code for this task must be executed as:

    ```
    python3 main.py 2 <input_img_dir> <output_csv>
    ```

  - Example:

    ```
    python3 main.py 2 ./input_images ./Task2.csv
    ```

4 Dataset

- The dataset for this assignment is available at the provided link.
- It includes a set of sample images for testing your implementation along with input and output sample CSVs for reference.
- The sample images are random frames extracted from the videos collected, and additional frames from the same videos will be to evaluate your algorithm.

5 Submission and Evaluation Guidelines

- Your submission must include the entire implementation as Python files, including:
  - `main.py`: A single Python script that serves as the entry point for running all tasks.
  - Supporting modules (if any).
- The script `main.py` should accept the following inputs during evaluation:
  - **For Task 1:** Input image directory and output image directory.
  - **For Task 2:** Input image directory and output CSV file.
- Submit a zip file containing all the required code files and a detailed report explaining the implementation details of your algorithm with file name ENTRYNUMBER.zip (For Example: 2023JRB2677.zip).
- Use Moodle for submission of your assignment.
- Any updates to the submission instructions will be communicated through Piazza.
- Grading rubrik (out of 100 marks)

Task	Marks
Task-1	50
Task-2	30
Report	20

Table 1: Evaluation Distribution

- You will be assessed on the output of your code and your effectiveness in explaining its implementation during the demo.