# Assignment: Lanes Detection

**Date:** 2025-April-08

**Author:** Chanjung Kim 22000188

**Github:** DLIP_CJ

IncludeFile

---

# Introduction

## 1. Objective

This assignment is about the process to detect the right and left lanes and draw them on the given images. Also, finds a vanishing point where two lanes intercept.

**Goal**: Correctly detect lanes and find a vanishing point where two lanes meet.

Two images are given:

- image where lanes are aligned in center
- image when a car is changing its lanes

---

## 2. Preparation

In this assignment following software installation is needed.

### Software Installation

- OpenCV 4.11, Visual Studio 2022
- DLIP_LAB.h & DLIP_LAB.cpp files

### Image set

Two images are used to detect lanes and find the vanishing point.

**Images link:**

Download 'Lane_center.jpg'

Download 'Lane_changing.jpg'

---

# Algorithm

In this section, the overview of algorithm, pre-process of images, and lane detecting algorithm are introduced.

# 1. Overview

In this section, the overview of algorithm used in this assignment is introduced in **Figure 1**.

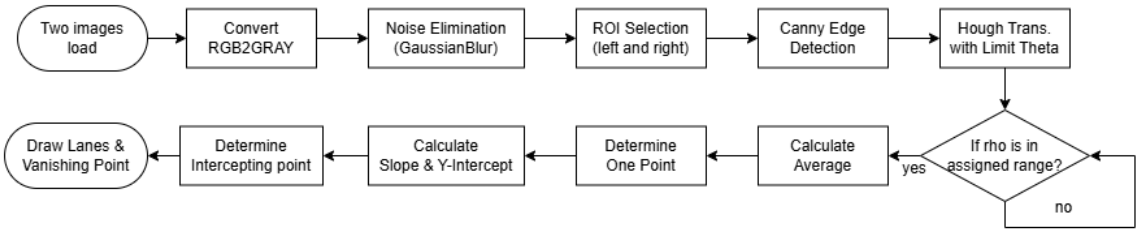



**Figure 1. Overview of Algorithm**

# 2. Pre-Process

In this section, methods used in pre-processing of input images are represented. This section includes filtering, ROI selection, and canny edge detection.

## Filtering

First of all, the input image is converted into gray scale to scale down calculation. To preserve original pixels and smooth the images, Gaussian blur method with 5x5 kernel was applied. The output images are presented in **Figure 2**.

(a) Before GaussianBlur (b) After GaussianBlur

**Figure 2. Result of Filtering**

## ROI Selection

ROI stands for Region of Interest, which greatly reduces computational resources and time. The static ROI has selected with one assumption that the camera is strongly sticked to a car. The static ROI for left lane starts from one-fourth of width and half of height with width of 1/4 of original image and height of 1/2 of original image, while right lane ROI starts from half of width and half of height with the same width and height of left ROI.

```
ROI_left = grayImag (Rect(0.25 * width, 0.5 * height, 0.25 * width, 0.5 *
height));
ROI_right = grayImag (Rect(0.5 * width, 0.5 * height, 0.25 * width, 0.5 *
height));
```

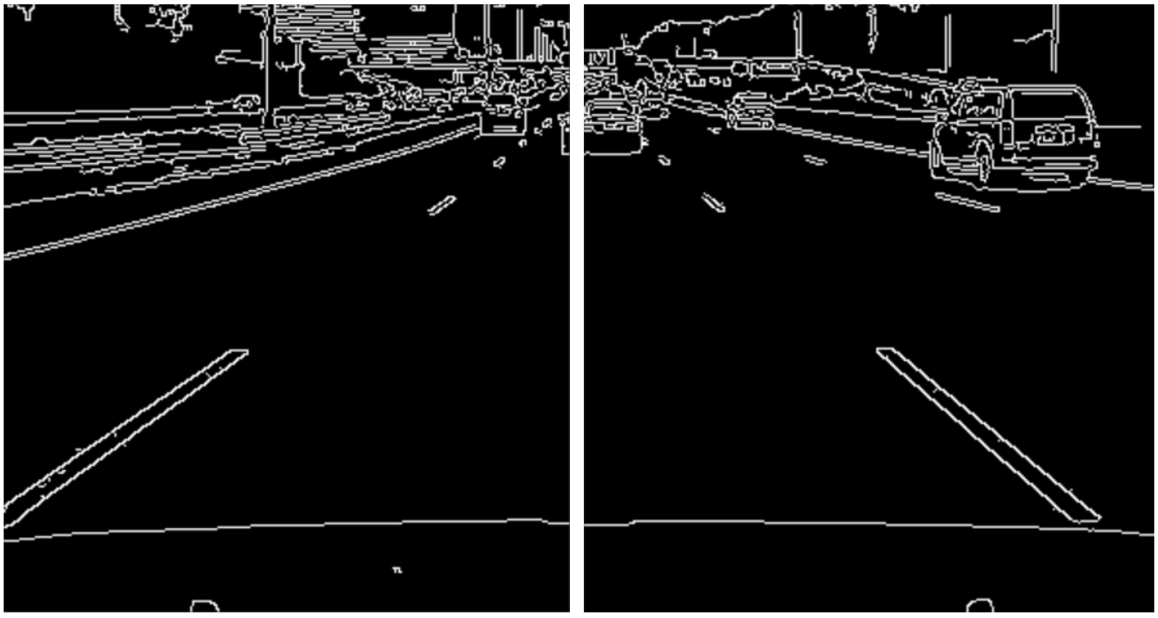
The image shown in **Figure 3** is the ROI of first image.

(a) Left ROI (b) Right ROI

**Figure 3. Result of ROI**

## Canny Edge Detection

Based on the selected ROI, edge detection using canny method was proceeded. The result is presented in **Figure 4** below.



(a) Left ROI (b) Right ROI

**Figure 4. Result of Edge Detection**

# 3. Lane Detection

To acknowledge true lane from the given ROI, two factors are considered with opencv function `HoughLines`.

- rho

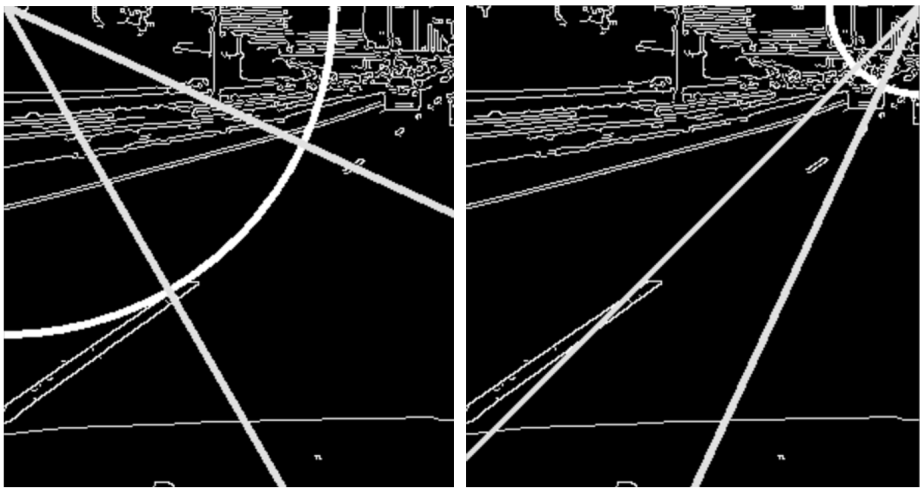
- theta

The range of parameter is analyzed in **Table 1** below according to left and right ROI. The width in **Table 1** refers to ROI width which is one-fourth of the original image.

**Table 1. Range of Parameters**

| ROI | $\rho$ (rho) | $\theta$ (theta) [deg] |
|---|---|---|
| **Left** | $\rho > 0.73 \times width$ | $25° < \theta < 60°$ |
| **Right** | $0 < \rho < 0.2 \times width$ | $115° < \theta < 135°$ |

The **Figure 5** visualizes parameter ranges in the ROI images. Since theta for right ROI is greater than 90 degree, parameter range of right ROI is expressed in left ROI for visualization.



(a) Left Range (b) Right Range

**Figure 5. Visualization of Parameter Range**

# Result and Discussion

In this section, result of algorithm will be provided with a discussion.

## 1. Final Result

The final results are presented in **Figure 6** and **Figure 7**. Left detected lane is drawn in red line, while right lane is in green. The vanishing point is encircled in orange.

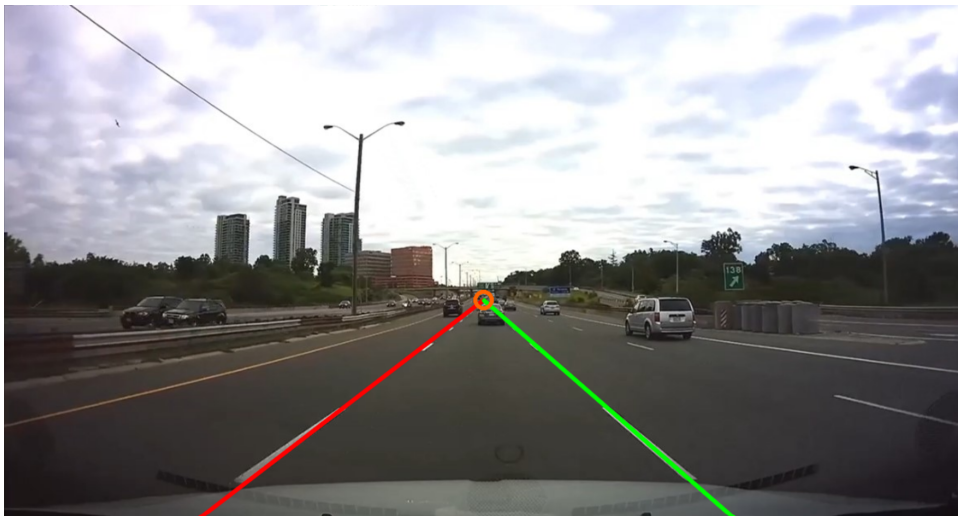


**Figure 6. Result of 1st Image**


**Figure 7. Result of 2nd Image**

## 2. Discussion

The number of detected lanes for each ROI, its average rho and theta is analyzed with **Table 2** below. The third column labeled as '# of detected lanes' are the number of detected lanes with rho and theta range. The fourth and fifth columns, theta and rho, are the average or those detected lanes to minimize the error. Using the theta and rho listed in **Table 2**, the result images are drawn as in **Figure 6** and **Figure 7**.

**Table 2. Result Table**

| Sample # | ROI | # of detected lanes | $\theta$ [deg] | $\rho$ [pixel] |
|---|---|---|---|---|
| 1 | Left | 5 | 52.60 | 216 |
| | Right | 16 | 131.00 | 33 |
| 2 | Left | 1 | 27.00 | 278 |
| | Right | 5 | 116.20 | 52 |

In the algorithm part, two significant factors, rho and theta, were mentioned to find a lane. **Table 3** clearly shows why this algorithm needed a rho range. Without rho limit, numerous lanes were detected, especially in first left ROI. To avoid Abnormal lane, those two parameter ranges were implemented.

**Table 3. Comparison of Rho Limit**

| Sample # | ROI | # of detected lanes without rho limit | # of detected lanes with rho limit |
|---|---|---|---|
| **1** | Left | 74 | 5 |
| | Right | 18 | 16 |
| **2** | Left | 8 | 1 |
| | Right | 7 | 5 |

Since this project objective is to detect lanes and find a vanishing point, the proposed algorithm has achieved the project goal successfully.

---

# Conclusion

In this assignment, detecting lanes and a vanishing point while driving a car is demanded. To avoid numerous edge, Gaussian blur method was applied as filtering. Two ROI for left and right lane were selected, and edge detection using canny method was proceed. Two significant factors, rho and theta were considered to acknowledge true lanes, which are taken to calculate average.

The result clearly detects lanes and points out the vanishing point. However, it is still doubted if it works well with real implementation with high performance. Further study is required to apply this algorithm into real situation.

---

# Reference


ykkim_gitbook

OpenCv