Lane Detection Study

The lane recognition algorithm is largely divided into a method using Hough transform and a method using inverse projection.

I will study about Hough transform method using Canny edge detection algorithm.

After detecting an edge on an image using Canny edge detection algorithm, the part representing lane among the edges is detected by using Hough transform.

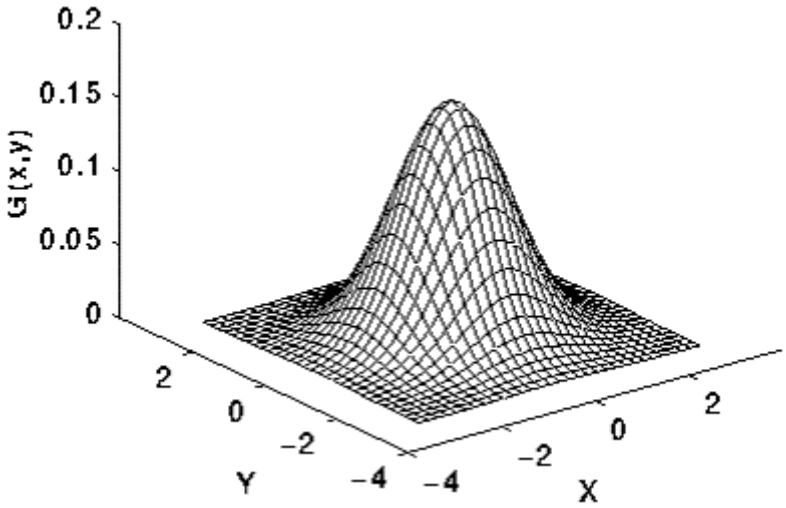
- Edges will be detected by Canny edge detection algorithm
- Lanes will be detected by Hough transform method based on the above result.

Steps

1. Preprocessing

Before applying the algorithms, we will have to reduce noises in the images to enhance the accuracy.

There are 3 ways to do this,

1. median filtering
    - the process of setting a region within an image, and making all pixel values the same. (median)
2. sharpening
    - Make a scrambled image vivid.
3. gaussian filtering
    - **We will perform filtering using a Gaussian normal distribution with a mean of 0.**
    - The farther away from 2D the less weight
    - 

- $$\begin{bmatrix} \frac{1}{32} & \frac{1}{32} & \frac{1}{16} & \frac{1}{32} & \frac{1}{32} \\ \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{16} & \frac{1}{32} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{16} & \frac{1}{32} \\ \frac{1}{32} & \frac{1}{32} & \frac{1}{16} & \frac{1}{32} & \frac{1}{32} \end{bmatrix}$$

2. Applying Canny Edge Detection Algorithm

This step can be divided into 2 parts,

1. Finding the edge of an object or area in the image
2. Finding the road lane edges among the edges

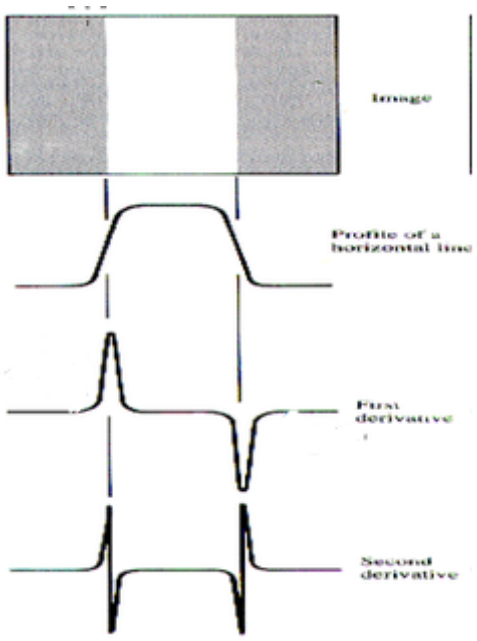
Canny Edge Detection Steps

1. Smoothing using Gaussian Filtering
2. Finding gradients using sobel mask
    - where the pixel values change dramatically
        - because, road = grey, lane = white
        - 

3. Non-maximum suppression
    - based on the detected edges from the above step,
    - compare adjacent pixels of those pixels,
    - remove non-maximum value pixels.
4. Double Thresholding
    - classifying weak edge and strong edge based on two thresholds
5. Edge tracking
    - connect the disconnected edges.

Basically, edge detection is a method of finding a boundary between two regions with different intensities.

Mathematical Explanation



If we look at the above image, we can see that the colors are dark, bright, dark.

if we draw a graph according to the brightness change on the x-axis, a waveform is generated.

If the waveform is first differentiated, a waveform indicating the intensity slope change is created.

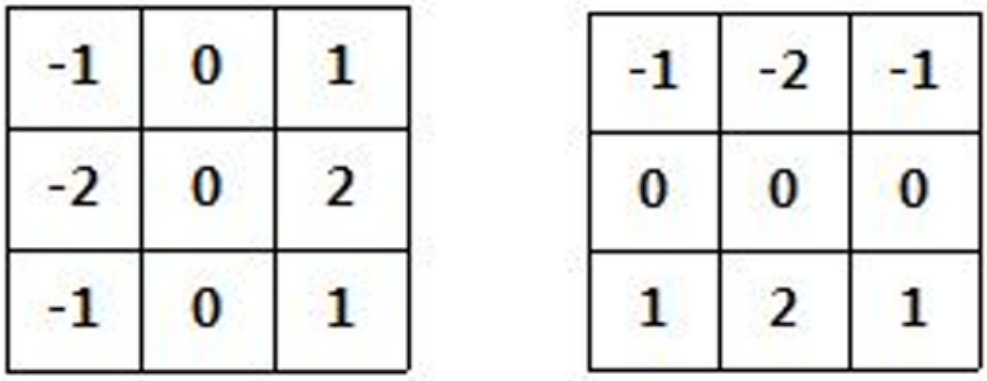
If the result of the above waveform is second differentiated, we can check the position of the light and dark parts of the pixel representing the edge.

The part that pops out in the graph is the edge.

How to apply it in the implementation?

In programming, we can replace it with a filtering operation using mask which allow us to extract the boundaries.

- **Sobel Mask**
    - Detects edges in all directions -> but reacts more sensitively to edges in diagonal directions
    - Noise-resistant because pixel values are averaged.

- 

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

    - The left mask is for vertical direction, right one is for horizontal direction.
    - In canny detection, the edge is obtained by detecting the edge by calculating the magnitude of the two values obtained by convolving these two sobel masks with the image.

Through this mask, pixels except edges (or considered as edges) will have 0 values.

Good notes to understand 2D convolution

- http://www.songho.ca/dsp/convolution/convolution2d_example.html
- http://www.songho.ca/dsp/convolution/convolution.html

After this process, we will use non-maximum suppression

- To make these edges sharper or thinner.

3. Non-Maximum Suppression

Non-maximum suppression refers to thinning the edges obtained through image processing.

Edges found using gaussian filtering and sobel mask are blurred, in other words, crushed/scrambled edges. So we have to use this method to find a clearer line.

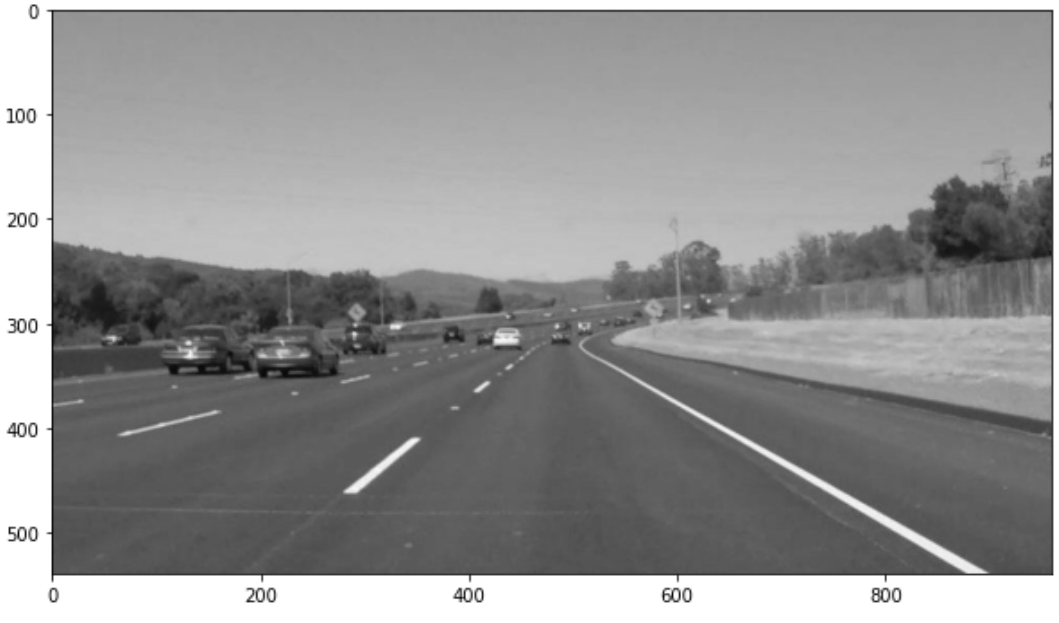
Current Programming Process

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
%matplotlib inline

img = mpimg.imread('image.jpeg')
plt.figure(figsize=(10,8))
print("This image is:", type(img), 'with dimensions:', img.shape)
plt.imshow(img)
plt.show()
```

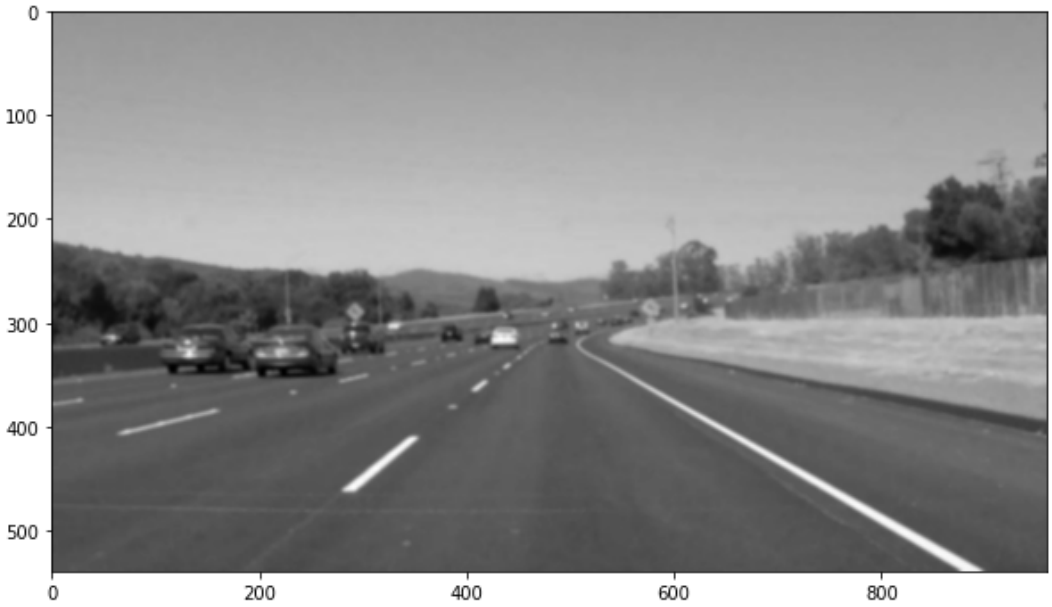
```
This image is: <class 'numpy.ndarray'> with dimensions: (540, 960, 3)
```


```
def grayscale(img):
    return cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
gray = grayscale(img)
plt.figure(figsize=(10,8))
plt.imshow(gray, cmap='gray')
plt.show()
```



```
def gaussian_blur(img, kernel_size):
    return cv2.GaussianBlur(img, (kernel_size, kernel_size),0)
#how blurry we want
kernel_size = 7
blur_gray = gaussian_blur(gray, kernel_size)

plt.figure(figsize=(10,8))
plt.imshow(blur_gray, cmap='gray')
plt.show()
```




```
def canny(img, low_threshold, high_threshold):
    return cv2.Canny(img, low_threshold, high_threshold)
low_threshold = 50
high_threshold = 200
edges = canny(blur_gray, low_threshold, high_threshold)

plt.figure(figsize=(10,8))
plt.imshow(edges, cmap='gray')
plt.show()
```