Lane Detection

Md Irshad Alam

Semester - VII

Roll No - 17BCS073

Abhay Kumar Gupta

Semester - VII

Roll No – 17BCS067

What?

- In this project, we will use python and OpenCV to detect the lanes lines on the road.
- We will develop a processing pipeline that will work on list of images and we will apply the result on video input.
- Processing pipeline will produce output corresponding given input which we will make use of.

Motivation

- Using lane detection technique, we can detect the lane without prior knowledge of the road geometry.

- Detecting lanes play important role in developing intelligent vehicles.
  - Avoiding road accidents.
  - Assists driver in changing lane.
  - Application in self driving vehicles.

How?

1. Loading input images
2. Color Space selection
3. Canny Edge Detection
    a) Grey scaling the images
    b) Applying Gaussian smoothing
    c) Apply Canny Edge Detection
4. Determine region of Interest
5. Hough transformation
6. Average and Extrapolate the lane lines
7. Apply pipeline on video input

Loading test input

- We will take series of individual images as input.

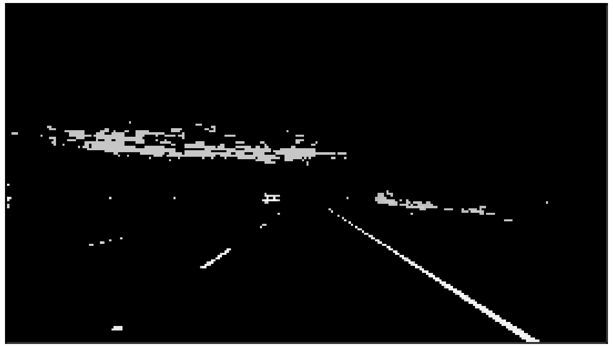
Color Space Selection & Masking

- We need to select the most suitable color space.
- It may be RGB, HSV, or HSL.
- It is found that HSL is most suited.
- we will try to retain the as much of the lane lines possible, while blacking out most other stuff.
- We will achieve this by using mask image which will black out everything except yellow and white lane lines.

Canny Edge Detection

1. Gray scaling the images

- Original input images contains three channel.
- It is very difficult to handle three channel.
- So, we convert the image to Gray scale.

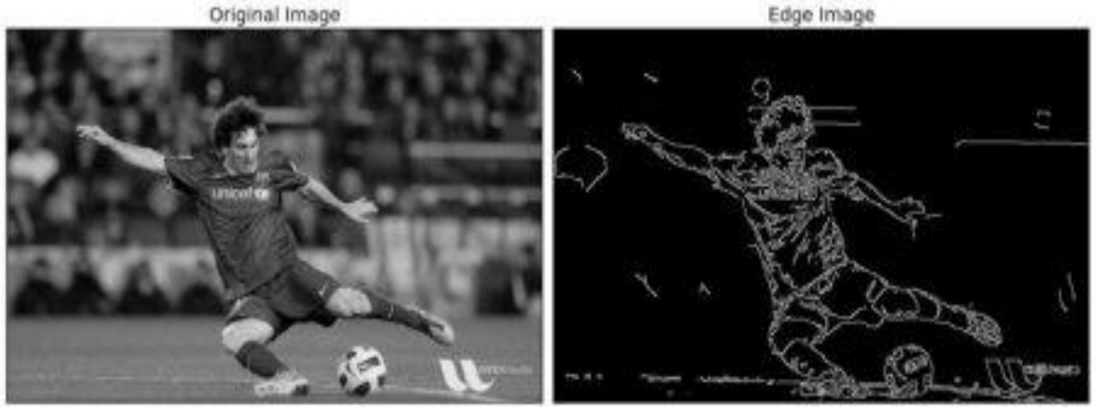


2. Gaussian smoothing

- In original input images noise is obvious.
- Which may result in unwanted edges.
- So, we filter out those noises using Gaussian filter.

Canny Edge Detection

3. Applying Canny Edge detection



Finding Intensity gradient of images

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

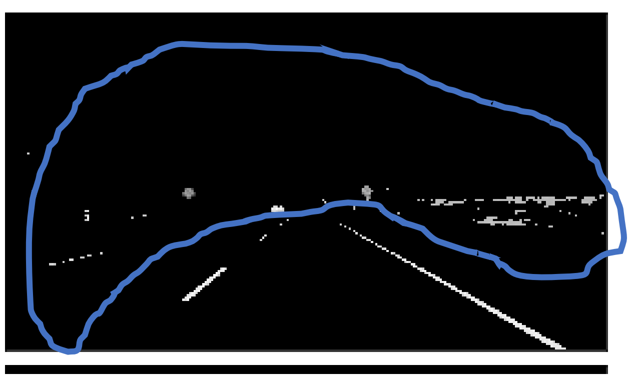
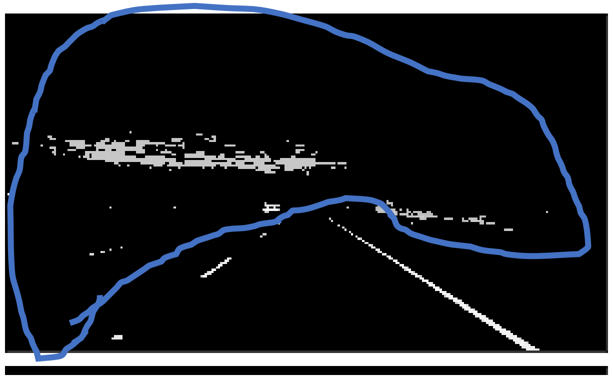
$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

G is edge gradient of particular pixel of the image

Hysteresis Thresholding

- If an edge pixel’s gradient value is higher than the high threshold value, it is marked as a strong edge pixel.
- If an edge pixel’s gradient value is smaller than the high threshold value and larger than the low threshold value, it is marked as a weak edge pixel.
- If an edge pixel's value is smaller than the low threshold value, it will be suppressed.
- The two threshold values are empirically determined and their definition will depend on the content of a given input image.

Determining region of interests.

- We are only interested in areas where lanes lines are found.
- So, we will make a mask and will cut those unwanted areas which are of no use.
-  

Hough Transformation

- The Hough Transform is a technique which can be used to isolate features of a particular shape within an image.
- It is most commonly used for the detection of regular curves, such as lines, circles, ellipses, etc.
- The linear Hough transform algorithm uses a two-dimensional array, called an accumulator, to detect the existence of a line described by

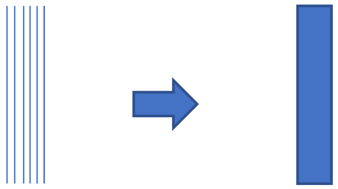
  $$r = ysin\theta + xcos\theta.$$

- Where r is the distance from the origin to the closest point on the straight line, and θ (theta) is the angle between the x-axis and the line connecting the origin with that closest point.

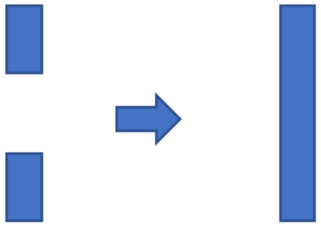
  - rho - Distance resolution of the accumulator pixels.
  - theta - Angle resolution of the accumulator in radian.
  - threshold - Only lines which are greater than threshold will be returned .
  - minLineLength - Line segments shorter than that are rejected.
  - minLineGap - Maximum allowed gap between points on the same line to link them.

Averaging & Extrapolating the Lane Lines

- We will get multiple lines detected for each lane line.
- We need to average all these lines and draw a single line for each lane line.



- We also need to extrapolate the lane line to cover the full length line.



Applying on video streams

- Finally, our ***processing pipeline*** is ready to apply on video streams.
- The output of the video streams will contains highlighted lane lines.

Programming Environment & Tools Used

- Python-OpenCV 4.4.0.40
- Python 3.7
- Matplotlib
- NumPy
- JUPYTER-Notebook 6.0.0
- Anaconda 2020.02
- GitHub

References

- International Journal of Computer Science and Mobile Computing IJCSMC, Vol. 3, Issue. 2, February 2014, pg.596 – 602
- https://opencv.org/
- https://www.python.org/
- https://en.wikipedia.org/wiki/Canny_edge_detector
- http://www.cs.unc.edu/~lazebnik/spring09/lec09_hough.pdf

Thank You !