Project 1 : Finding Lane Lines on the road

My Pipeline consists of 5 steps.

Step 1: Convert the images to Grayscale , using cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)

Why should we convert?
We are interested in detecting white or yellow lines on images, which show a particularly high contrast when the image is in grayscale. Remember that the road is black, so anything that is much brighter on the road will come out with a high contrast in a grayscale image.

Step 2: Convert **GrayScale images to Guassian Blur**

Perform Guassian Blur Technique on Gray scale image to smoothen the edges of an image to reduce noise.

I did using the kernel size of 7.

cv2.GaussianBlur(grayscale_img, (kernel_size, kernel_size), 0)

Step 3: We apply **Canny Edge detection** on Guassian Blurred images to identify edges in an image and discard all other data.
low_threshold = 50
high_threshold = 150

cv2.Canny(blurred_img, low_threshold, high_threshold)

Step 4: Region of Interest:

We will determine a region of interest and discard any lines outside of this polygon.

Step 5 : Hough Transform

We need to do Hough Transform to extract lines and color them
rho = 1
theta = (np.pi/180) * 1
threshold = 15
min_line_length = 20
max_line_gap = 10

Separate left and Right Lane

we can separate left and right by finding Slope.

- Left line : Gradient will be negative [as value of y decreases with increase in x value]
- Right line : Gradient will be positive [as value of y decrease with decrease in x value]

  if two points (x1,y1) (x2,y2) are given : Slope = (y2-y1)/(x2-x1)

Lane Extrapolation

Trace full line from the bottom of the screen to the highest point of our region of interest.

Identify potential shortcomings with your current pipeline

This sort of a pipeline is extremely ineffecient and unsafe for lane finding:

- the region of interest mask assumes the lane is in the very center of the image
- different cameras will have different settings/outputs and will require different thresholds on canny detection and different kernel sizes with blurring

- it tries to fit a line to a road which very well could be curved; it should be a quadratic or cubic fit
- in general, I felt like a lot of the parameters were tuned specifically to these three inputs, while a camera running on a self-driving car will see lane lines that haven't been trained for/tuned to before and must still find the lane lines
- on extremely light roads, the camera will not find the lane lines because the edges won't contrast enough

Possible suggested Improvements to pipeline:

1. A possible improvement would be to manage our low and high threshold values in the pipeline to detect the lanes irrespective of the light color patches.

2. Another improvement which will benefit grately is image crop & stabelization. The videos once stablized will further help us keep our slope value constant throughout the detection and mapping process.

3. Use a camera calibration technique