Write up for the project “Finding Lane Lines”

Identifying lanes of the road is very common task that human driver performs. This is important to keep the vehicle in the constraints of the lane. This is also very critical task for an autonomous vehicle to perform. Lane detection is one of the most important aspects of autonomous navigation. It is the first step in enabling the vehicle to visualize and then make sense of its environment.

The goal of this project was to create a pipeline that finds lane lines on the road.

1. My Approach

1. Convert the Original Image into HSV colorspace. Yes, HSV! Not Grayscale.

Reason for choosing HSV over Grayscale:

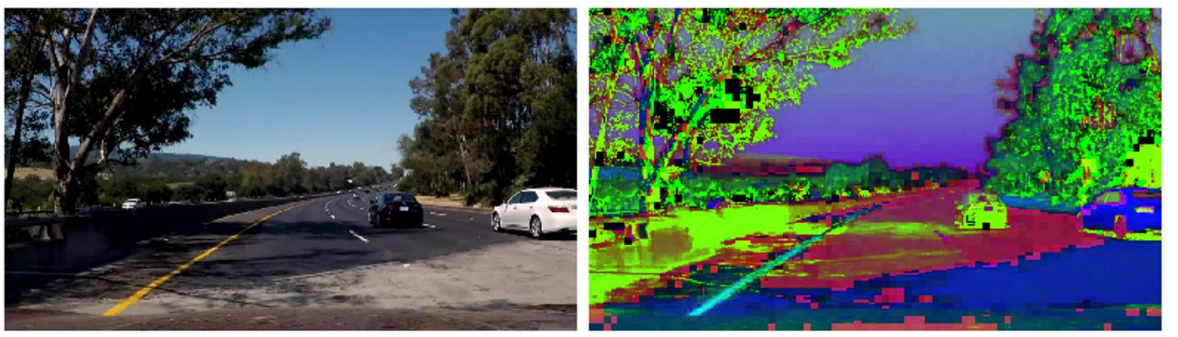
In grayscale, we have a lot of loss of information, especially when the light conditions change and when the colors change. We usually have white lanes and yellow lanes, and different light conditions as shadows, bright light, or low light, etc.

Also, we usually represent most of the images in RGB colorspace for describing colors and saving them digitally. But it is not a good option when the light conditions change.

So, we have to choose a colorspace that is more robust to the varying light conditions. So I chose the HSV Colorspace, as the color information is encoded in single channel H. This way, the information in this channel doesnt change with varying lighting conditions.

Additional point to back up why I chose HSV over Grayscale. Please refer the paper “The effect of colour space on tracking robustness” which suggests that “…the colour spaces of YCbCr and HSV give more accurate and more robust tracking results compared to grayscale and RGB images….” Link: https://ieeexplore.ieee.org/document/4582971

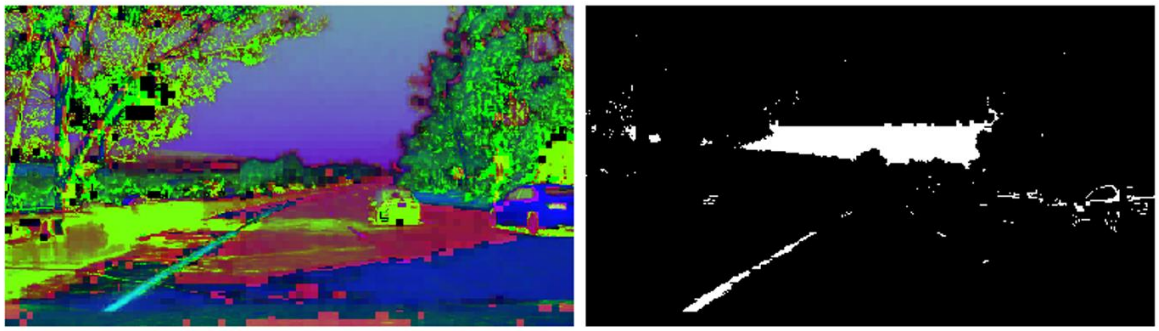
Sample image which shows the shadowed image with yellow lane on the left, and the HSV converted image on the right.



2. We then take the output of the previous step which is the Original Image converted into HSV colorspace and apply the GuassianBlur function.

This helps in removal of any noise from the image. It's an important pre-processing step for canny edge detection.

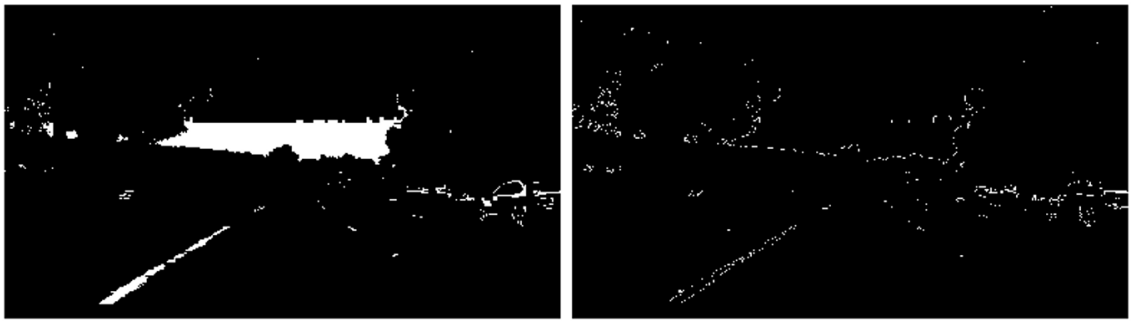
3. Masks for yellow and white colors are applied on the HSV image to extract lane lines from the image.



The Reason Why ?

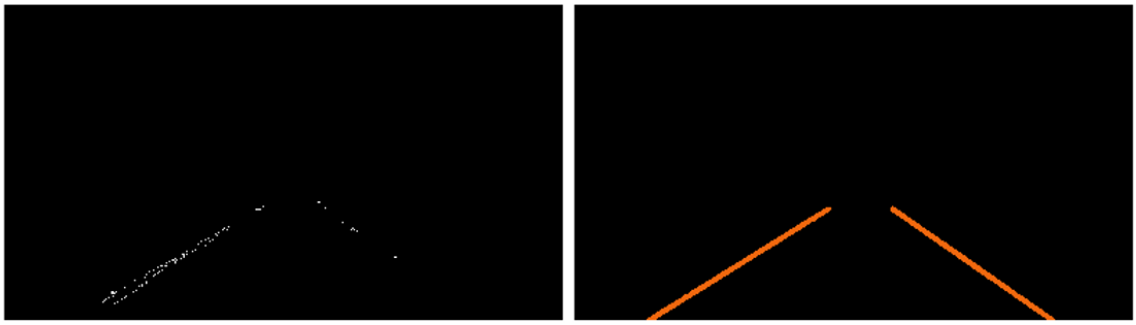
Even after conversion to HSV color space and application of Gaussian Blur, there can still be noise in the image that can interfere with the identification of lane lines. An effective way to get past this is to look for specific colors in the image, filtering out the noise in the image. In our case, we know the lane lines are primarily yellow and white in color, we can make use of this information to apply yellow and white masks on our HSV image and try to remove everything else, other than the lane lines.

4. Then the edges in the image are detected by passing the smoothed out gray image into the opencv's canny function.



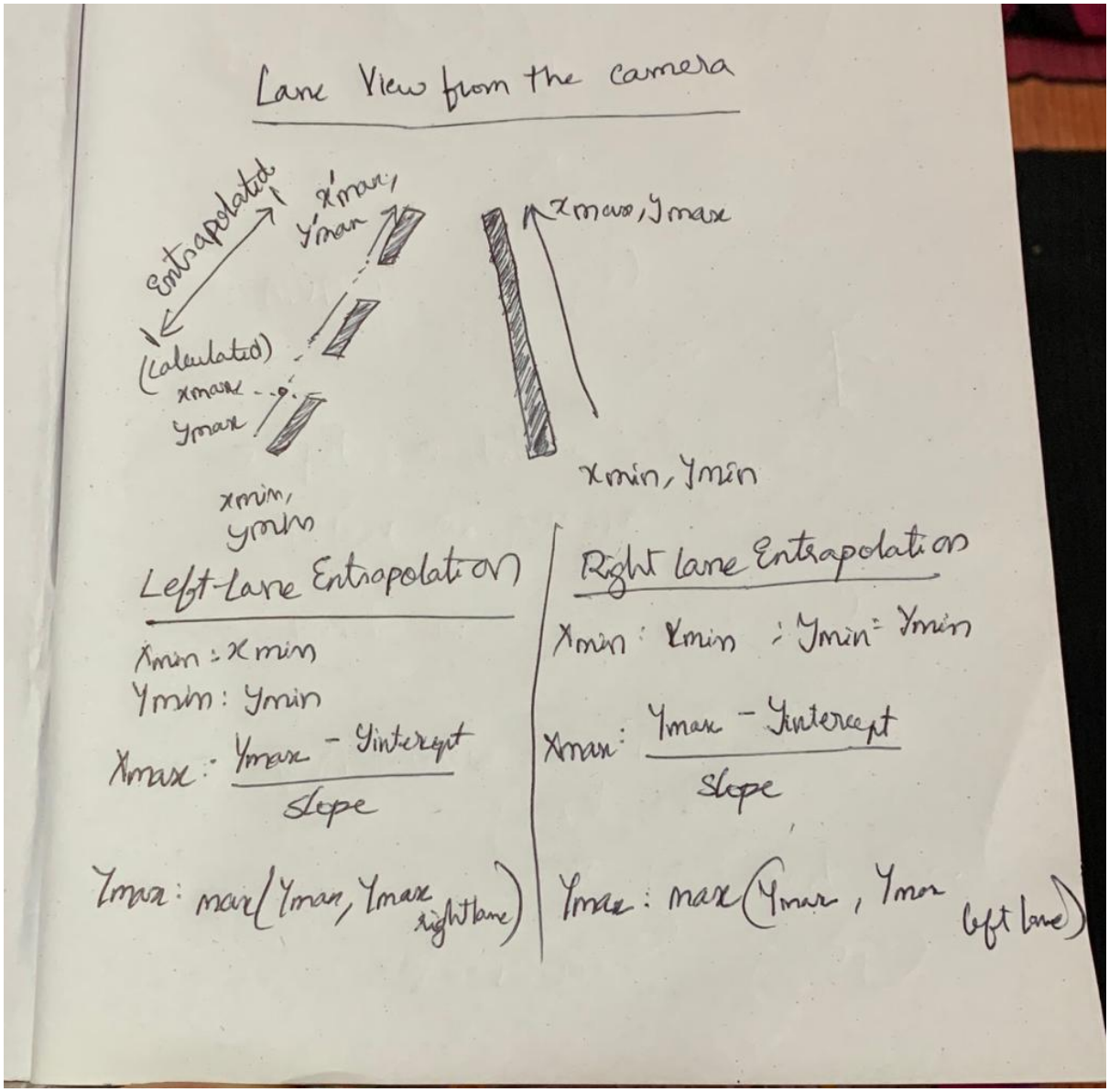
5. After the edges have been detected, only that portion of the image needs to be considered for further processing that has road lane lines in it. So we make a polygon for 4 sides which covers our region of interest. This region is highlighted in the image below. And next I only focus on that region in the image output we got in step 4. And result is shown on the right.

6. Now that the image has only the canny edges for the lane lines portion and the rest of it blacked out, we can draw hough lines on those detected edges. Here is the highlighted part of the lanes.



Drawing lines from the detected edges is one of the most important aspects for this project. For two single lines to be drawn over the edges, one for each line of the lane (left and right), the draw_line() function is modified in the following way:

DRAW-LINE ALGORITHM



I have made an extrapolation algorithm considering both left lane and the right lane to draw the extrapolated lines on the image. Below are the step to get there:

a. To extrapolate the lines returned from the hough lines and create two single lines, two endpoints (x_min, y_min, x_max. y_max) for the two lane lines and their corresponding

slope values (positive_slope, negative_slope) and intercepts (positive_intercept, negative_intercept) are needed. This forms the problem statement for the algorithm.

b. The array of lines returned from the hough_lines() function is iterated upon and for each line, the slope is calculated and stored in spearate arrays for positive and negative slopes.

c. From whether the slope is positive or negative, it can be inferred if the line belongs to the left line of the lane or the right line.

d. To remove noisy lines pertaining to edges not belonging to any lane, but still within the lane area, a threshold is defined for the value of the slope acceptable for the algorithm.

e. For each line, the intercept is also calculated and stored.

f. The y_max for both lane lines is the point at the bottom of the image, from where the lines start.

g. The slope values for both lines can be calculated by taking the averages of the corresponding positive and negative slope values. This also helps in removing some noise pertaining to the slope values.

h. Similarly, the intercept for both lane lines can be calculated by taking the mean of the stored intercepts.

i. The y_min can be found out by comparing all the y values in all the lines and taking the minimum value.

j. Now x_min and x_max can be found out by just fitting all the previously found parameters in an equation of the line.

k. The lines can now be drawn over the lane lines.

7. This image is now combined with the original image and the returned image would be the original one, with the algorithm generated lane lines drawn over it.

8. The final result looks like this:

2. Reflections

The conversion to HSV colorspace makes the pipeline robust to varying lighting conditions, but still the lines are a bit jittery. Improvements can be made to the draw_line algorithm to make better use of information from previous frames of the video while processing.

3. Conclusion:

The given task has been successfully completed, and the project meets the rubrics given.

You can also find the project on my GitHub repo:
`https://github.com/harivamsi9/Finding-Lane-Lines-using-OpenCV-for-Self-Driving-Cars.git`