Finding Lane Lines on the Road

Writeup ---

Finding Lane Lines on the Road

The goals / steps of this project are the following:

- Make a pipeline that finds lane lines on the road
- Reflect on the pipeline in the following report

Reflection

1. Lane Detection Pipeline

The report covers in detail the image processing pipeline used to draw the lane lines in an image/video as the car is driving. This is one of the first steps for designing an autonomous vehicle as the lane lines act as reference for vehicle steering.

Image processing techniques are used to isolate lane lines, separate and draw according to right and left lanes.

The pipeline is as follows:

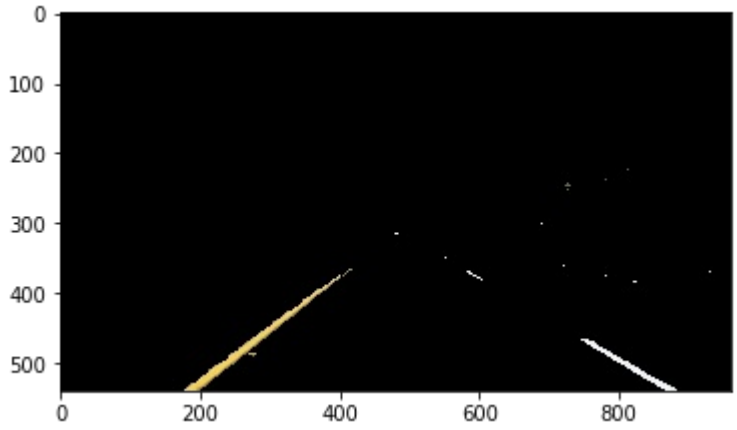
1. Convert from RGB to HSV color space
2. Apply greyscale conversion for easier processing and detecting lanes
3. Apply Gaussian blur to smoothen the images
4. Apply Canny edge detection on the smoothened greyscale image
5. Define a region of interest to remove the lines outside the region
6. Apply the Hough transform to find the lanes within the region
7. Classify the right and left line segments
8. Draw the lane markings for the left and right lanes

The example image used to demonstrate each step of the pipeline processing is as below:

1.1 Convert from RGB to HSV Color space

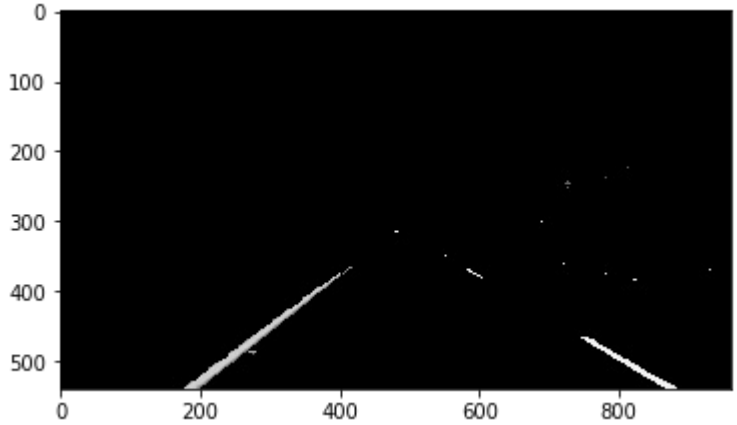
I decided to convert the RGB color space to HSV to better isolate the lanes in different lighting situations. HSV stands for hue, saturation, and value. Converting the image to HSV color space allow s for more resistance to changes in lighting, since the hue value w ill remain the consistent even in dark or very light scenarios. This results in color consistency across different lighting situations. This is evident in the challenge video, w here there is some shadow s from the trees on the left, creating a darkened yellow lane line. The algorithm processes those frames and is able to detect the yellow hue.

The follow ing image show s the image filtered according to HSV values for yellow and w hite, masked over the original image.



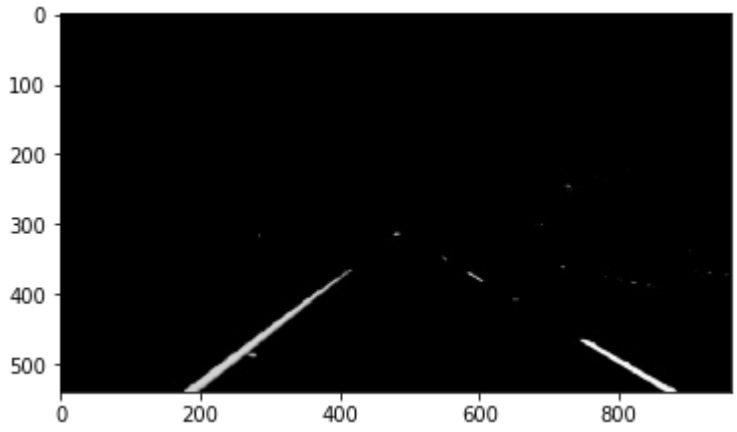
1.2 Convert from HSV to Greyscale

The reason for greyscale conversion is to observe higher contrast in various scenarios against the black of the road. The image below is the HSV filtered image converted into greyscale, w hich is then used for further pre-processing.



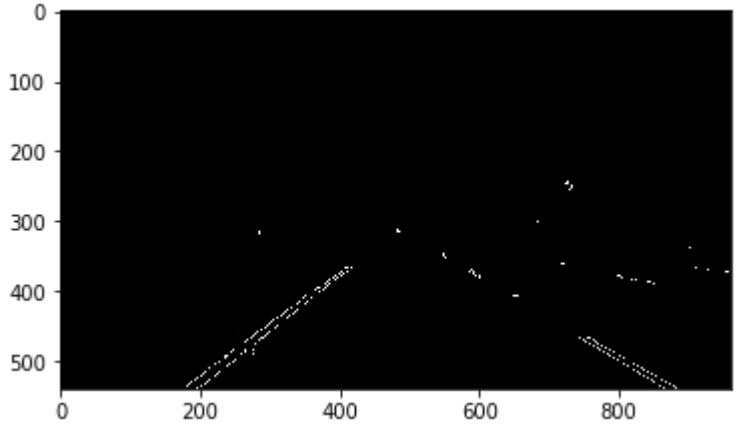
1.3 Gaussian Smoothening

This technique is used to apply a blurring feature to the image, w hich is used to reduce noise of the signal w ithin the image. The Gaussian filter applies a spatially w eighted average across the image w hich produces a smoothening effect. The amount of blur is dependent on the size of the kernel. The kernel size for the filter applied w as 5, therefore a spatial average w as computed across 5x5 filter size. This is a useful pre-processing technique before edge detection to reduce the noise that could be picked up by the edge detection algorithm. The resulting output of the Gaussian kernel applied to the greyscale image is seen below.



1.4 Canny Edge Detection

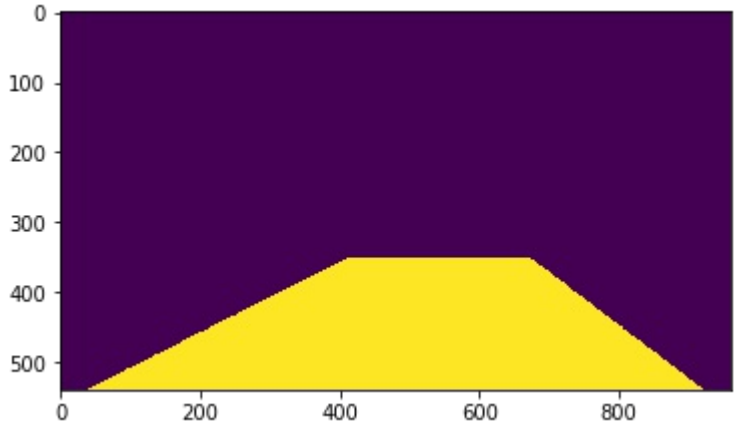
The purpose of applying the canny edge detection algorithm on the image is to determine the lines within the image. The intensity gradient of the image is identified by filtering the image with vertical and horizontal Sobel kernels. The edge gradients and orientations are obtained and then non-maximum suppression scan is used to remove pixels that may not constitute an edge. Those with local maximum gradient form a 'thin edge' and finally, hysteresis thresholds are used to pick up stronger edges. The point is included if it is above the highest threshold, or within the range of thresholds and are next to edges that are above the highest threshold. Those that are below the lowest threshold are not included as an edge. The threshold value range that was chosen were 50 to 100 and display a binary map of edges observed in the image. The edges for the original image are seen below.



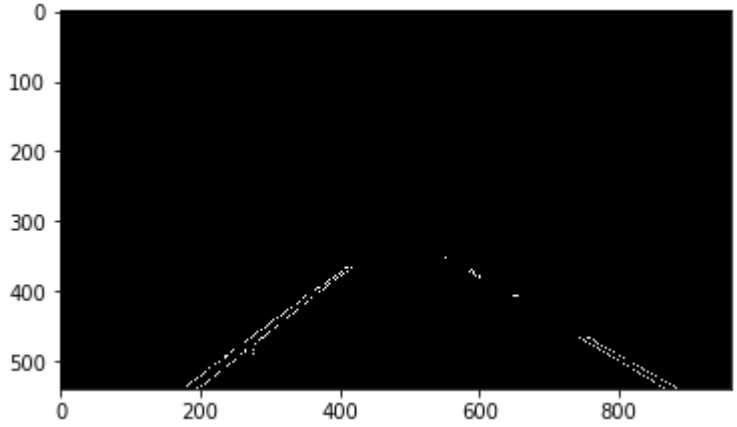
1.5 Region of interest

I selected a region of interest in which those points that are within the defined polygon remain, whereas those that are outside are removed. There is an assumption that the lanes in the video remain in roughly the same region and that the elevation does not change throughout the video. Selecting an ROI helps to reduce the lines detected that don't conform to any lane.

The ROI specified can be seen with the mask below. This is applicable across other variations of images.

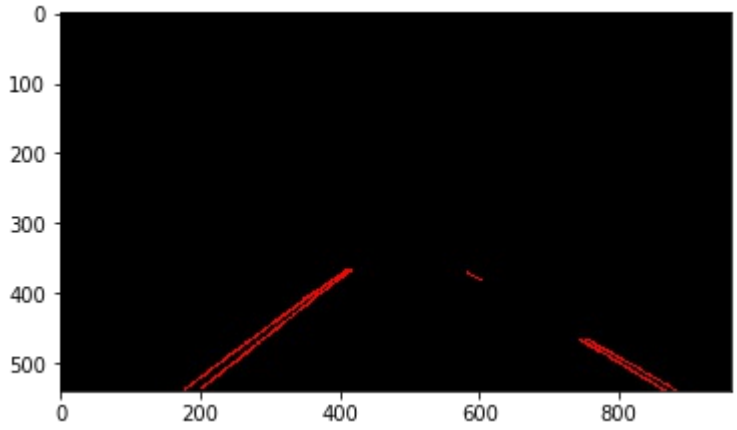


As a result, the image with the ROI mask is seen below:



1.6 Hough transform

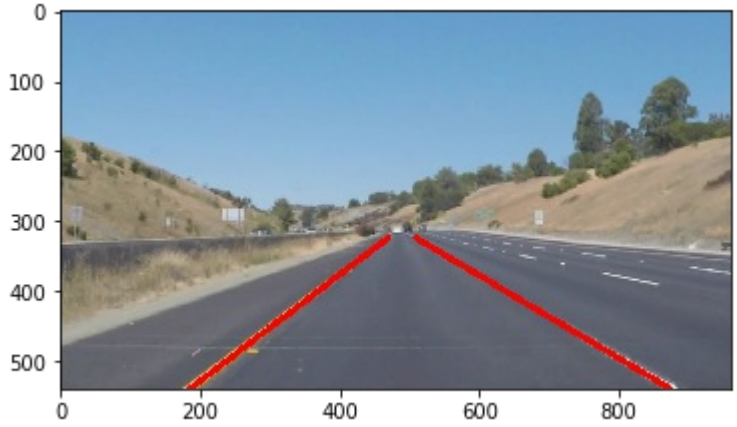
The Hough transform is a method used to detect lanes from the canny edge output. Each line (y = mx + b) can represented as a point (b,m) and the lines that have the same point value correspond to the same line. However, this doesn't work for vertical lines with infinite slope. Therefore, the line is transformed into polar coordinates (into Hough space), and represented with (ρ,θ). ρ is the perpendicular distance from the origin to the line, and θ is the angle formed by the perpendicular line and horizontal axis. Each line is plotted in a ρ vs θ plot and the brightness of areas correspond to the most probable lines. Parameters were input specifying the ρ and θ accuracies, and the threshold which is the minimum vote for a line to be considered. Additional parameters were also specified, the minLineLength and maxLineGap. The threshold chosen was 15, minLineLength=20, and maxLineGap=10. The Hough lines detected in the image are seen below:



The weighted image for the Hough transform lines over the original image can be seen:

1.7 Drawing Lines Over the Lanes

In order to draw a single line on the left and right lanes, I added functions that would take in the list of lines from the Hough transform and separate them according to slope. The left lane has a positive slope and the right lane has a negative slope. Therefore, I derived a function that would take in the line segment values, calculate the slopes and y-intercepts, and separate left and right lanes depending on the sign of the slope. I then took the averages of the slope and y-ints for the indexes for which the slopes were within 2 standard deviations of the mean. This is an attempt to reduce outliers in the calculation of the slopes that would be used to draw over the lanes, and produce an overall better representation of the lane line. These left and right average slopes and y-ints were used to obtain a linear fit model of the lane lines which are then drawn in red over the original image.



The pipeline was then tested against three videos, in order of increasing difficulty. There is occasional jittery-ness of the drawn lines due to rapidly changing slopes between frames, most notably in the challenge video (where the car is turning).

2. Potential Shortcomings with the Current Pipeline

One potential shortcoming would be what would happen when the car was driving at an elevation different from that seen in the video. The ROI is more selective for the test images and videos provided in this exercise. However, if the elevation changed, the ROI might crop out important areas in the scene and those lanes would not be detected. This pipeline is also not very robust for bumpy roads as well since the camera angle would change quite often and it may be hard to detect lane lines.

Another shortcoming could be if the car was driving in the middle lane and both lane markings were dotted. There might be some issues trying to correlate canny edge lines into the Hough transform. This shouldn't be a problem though, as the pipeline works on each left and right sides having dotted lanes, with the other lane a solid line. However, this pipeline would not work in situations of road without any lighter lane markings. As well, the linear fit is being performed on dotted lines, with less data points. The weighting of each point influences the slopes greatly, which causes may cause more fluctuations in the drawing of the lines since the fit will be slightly altered. There are some improvements that can be made to account for these changes.

3. Possible Improvements to the Pipeline

A possible improvement would be to link the frames together and use information obtained in previous frames to smoothen the lines drawn over the lanes for the video duration. This will reduce the fluctuations in change of the linear fit for the lines drawn. One can store slopes and y-intercept values from previous frames, and use those to recompute the mean with the current frame values. Of course, there would be a weighting applied to more recent frames, and this method makes the drawing of lane lines more smooth.

Another improvement could be to increase the polynomial degree of fitting for the drawn lines. For the challenge video, the lane lines drawn correspond to straight lines calculated from the curve in the road. To draw a better representation of the curve, one can use higher degree polynomials to fit the bend.

One can also use RANSAC, which stands for random sample consensus, for removing outliers during polynomial fitting. This is similar to what was performed in this pipeline, which is removing the outliers within a standard deviation, before the line fitting. This method may result in a linear fit that more appropriately describes the lane lines and a model will always be computed.