Algorithm for Line finding

1. Get image with clear lane lines. (after calibration, thresholding, perspective transformation)
2. Divide image into half from middle, assuming car is in within lane. Left half image will have left lane and right half image will right lane.
3. Identify starting location of left lane in left half of image and right lane in right half of image. “Peak in Histogram” method is used. For this method, only lower half image is used. Reason for using lower half image is that lower will have relatively straight vertical line. Which means if we sum across column, the column with highest sum will likely have lane on it, including starting point of the lane. Highest sum before mid-point will be related to left lane, and mid will be related to right lane.

To identify “peak” the “highest sum” we have used scipy.find_peaks_cwt. This function use 1-D array as required input, we can also define width of peak in our case it will be width of lane. And distance between one peak and next peak to that peak, in our case it will be distance between left and right lane. Width of peak: 100 (this should cover the expected width of peaks of interest) Distance between two peaks “max_distances”: 1000 peak_indexes = find_peaks_cwt(x ,[100], max_distances=[1000]) 1000 is imperial value, based on visual assessment, see "Histogram : Location of right and Left Lane" in Task-4 below.

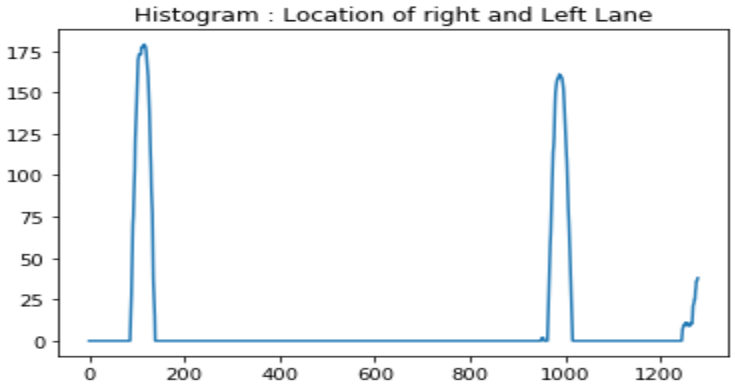
After getting peaks indexes, next step will be which peak belong to left lane and which belong to right lane. For that we order maximum strength index. To do that we first have to order indexes for that we use. np.argsort (-peak_indexes) “-“ used for descending order First index will be for right, second index for left. That is if two index are found otherwise one previous index is used.

1. After identifying starting points, we will use “sliding window” technique identify lane across the image, moving bottom to top.

2. We will use 2 order polynomial to draw a line across this predicted lane pixel points.

3. For error correction and removing outline pixels,
   a. For first frame, we will discard pixel as lane pixel, if pixel lane is more than 60 pixel away from previous identified pixel. Reason being we are assuming lane will be relatively straight so angle between current and previous prediction will be relatively less. Pixel is more than 60 we will use previous predicted location.
   b. For following frames for treatment of error propagation, I. Adjust coefficient. For that we use the standard deviations. If deviation is more than 0.0005 we are using previous image frame coefficient.
   ii. After image thresholding (with binary image mask) with we are making sure pixel count is more than 5 (to remove noise) (after filtering pixels with strength less 0.5)
   c. For smoothing we used a moving average filter, it smooths data by replacing each data point with the average of the neighboring data points defined within the span. This process is equivalent to lowpass filtering with the response of the smoothing given by the difference equation
   https://www.mathworks.com/help/curvefit/smoothing-data.html

4. **What happens when left or right line is not detected for long or its missing from start. I added/subtracted 850 based on detected lane (right or left, less or more than 500). This technique kept lane relatively smooth than before. See “challenge video” for this. see function get_polynomial_fit for implementation. 850 number came from histogram analysis (Step 4 in notebook) its average distance between left and right (see histogram in step 4). If no edge was detected we set 150 for left and 1000 for right again based on histogram.**



Limitation in project

Algorithm will fail if road is too much winding. It means in one direct when turning, you might have smaller road portion in front of you. Means you might be seeing road side edge. There is no point of drawing lane on it. see harder challenge video for this effect. One might think road curving angel might help but this situation could even arise when you reach dead end of road or you have T intersection, where in front of you is road edge. this is no point of doing image processing and draw lane on it. What might help is that we have to keep track of road itself not only lane line on it. we could train our processing pipeline color of road both day and night. Accurate Detection of road edge might help a lot.

Image Correction Sample

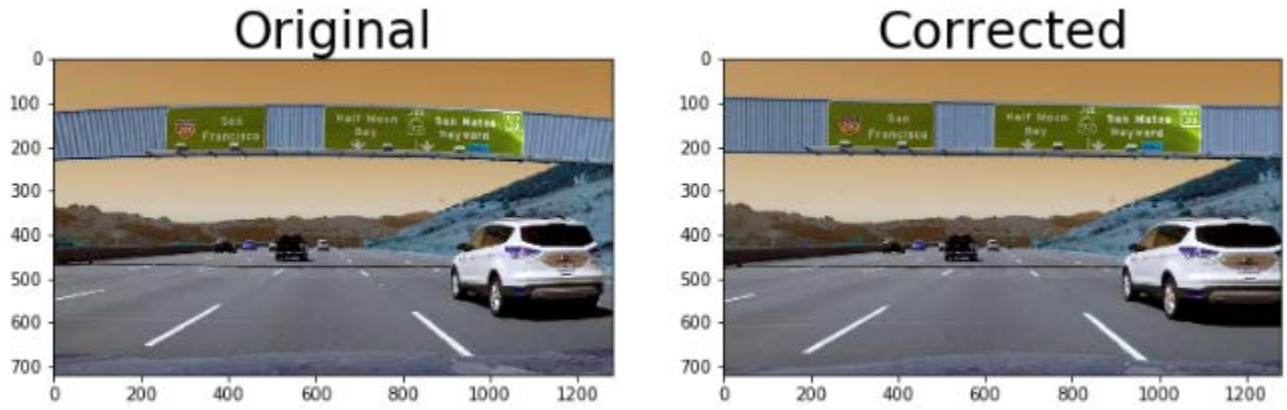
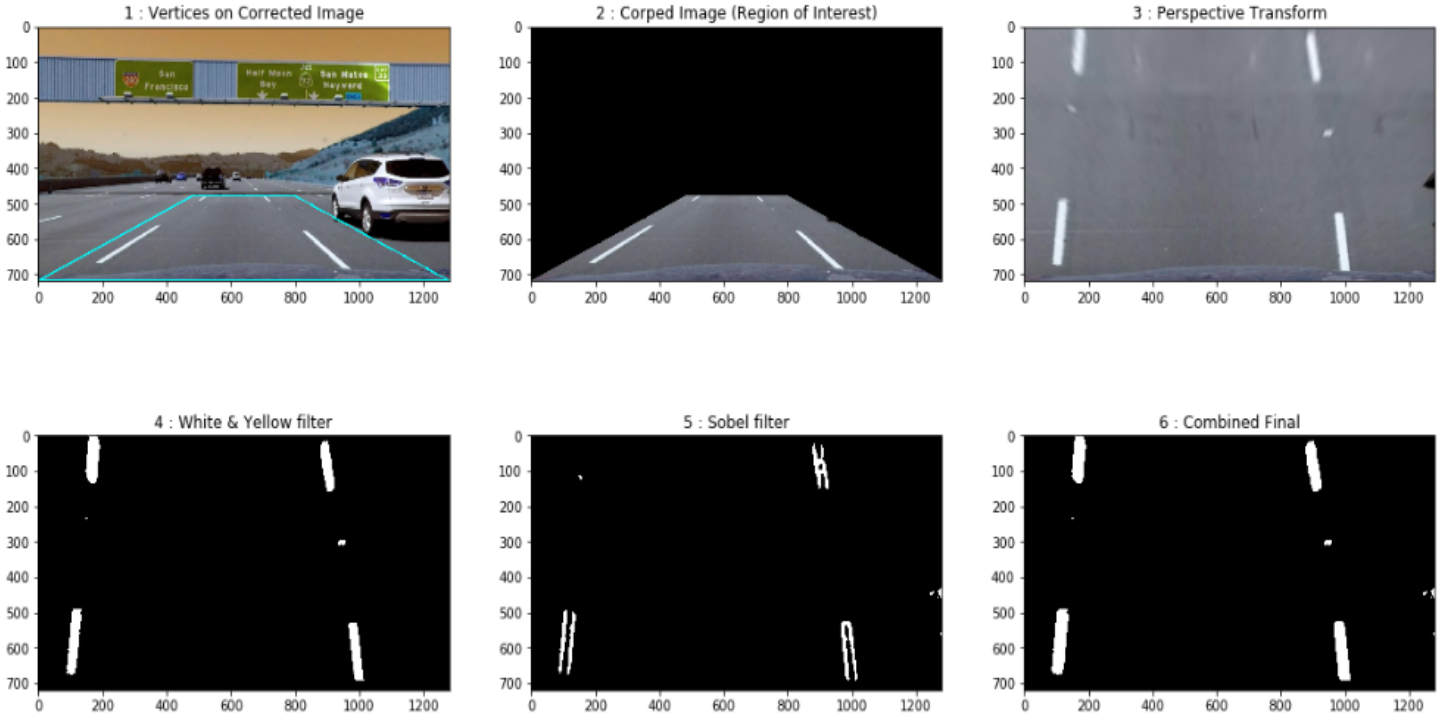
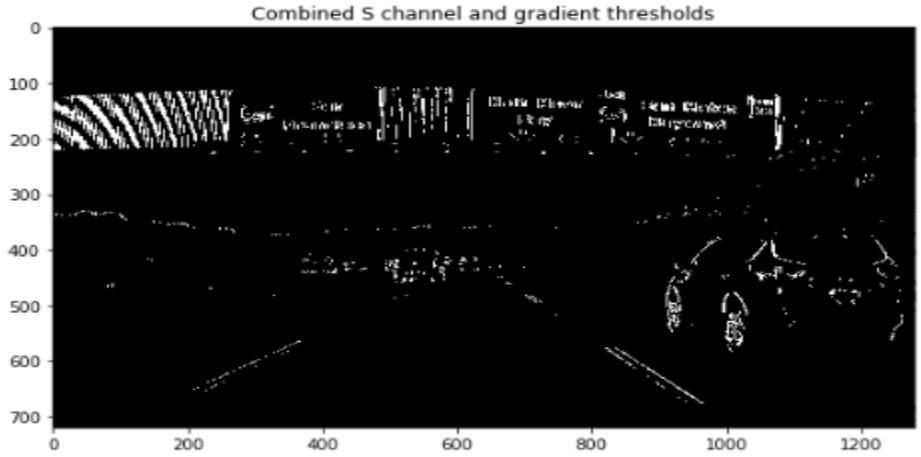


Image processing Steps

1. Correct Image
2. Corp region of interest
3. Do Perspective transformation
4. Apply color filter (white and yellow) (on Perspective transformation)
5. Apply Sobel filter (on Perspective transformation)
6. Combined filter (on color and sobel filtered image Step 4 & 5)

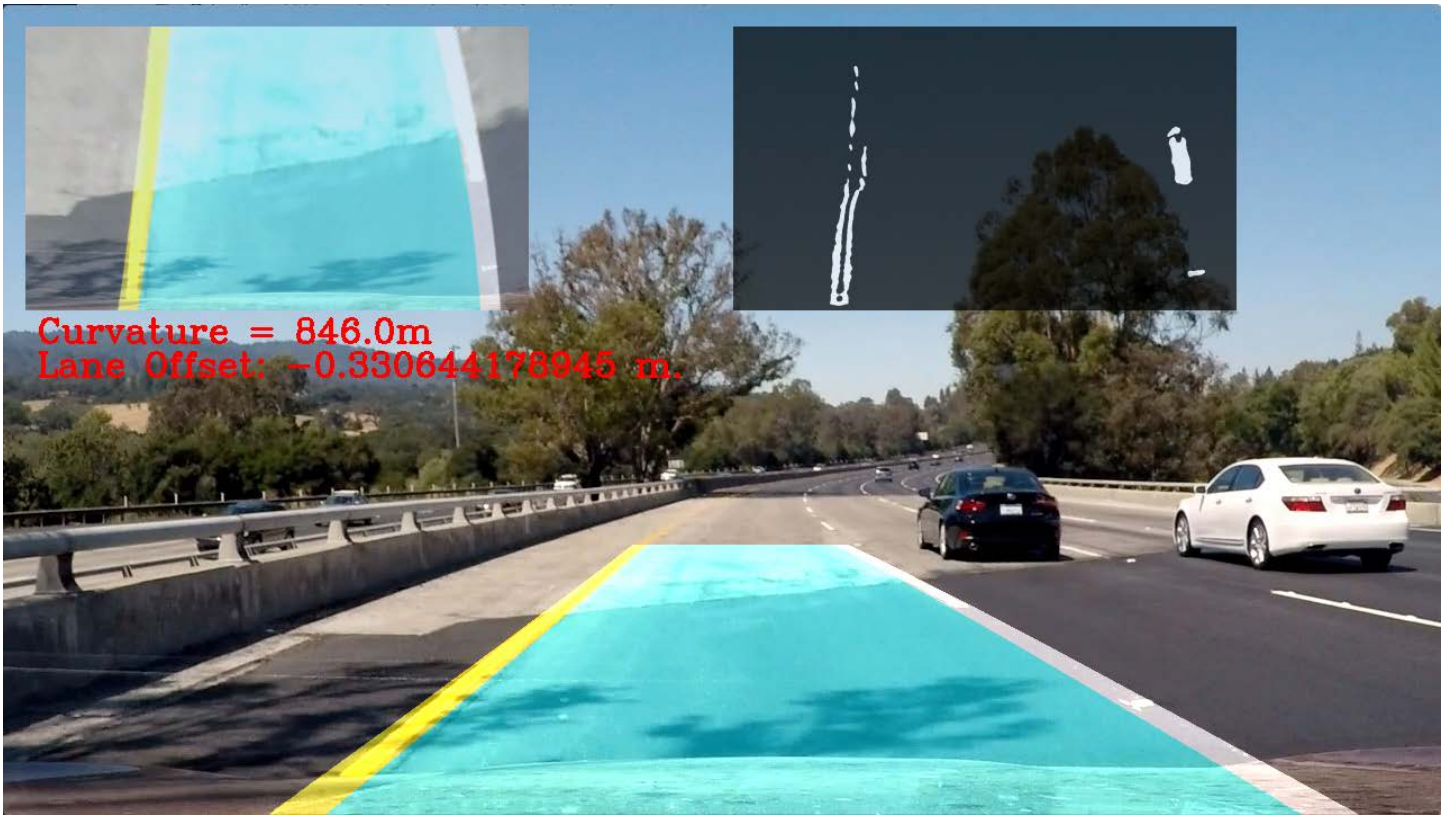


For Sobel filter we used L channel threshold , not S channel as L Channel performs better for brightness contrasts. Direction and magnitude was also applied for sobel filter. See below for Sobel filter result of picture of step 1 above

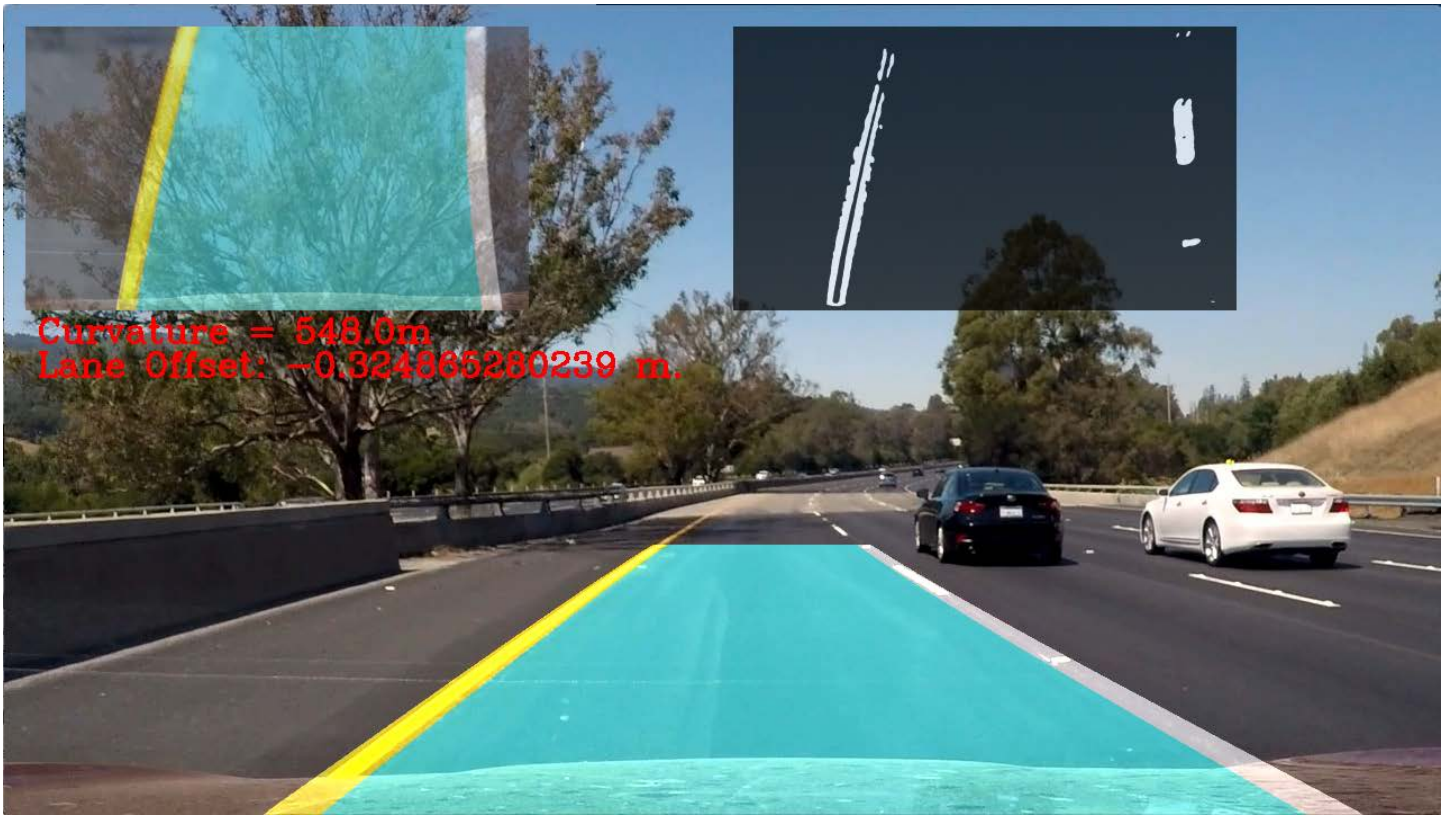


Sample Images (with bird eye view & edge detection)


Curvature = 730.0m
Lane Offset: -0.195925275769 m.


Curvature = 846.0m
Lane Offset: -0.330644178945 m.


Curvature = 1212.0m
Lane Offset: -0.0519170125159 m.


Curvature = 548.0m
Lane Offset: -0.324866280239 m.