Advanced Lane Finding Project

The goals / steps of this project are the following:

- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- Detect lane pixels and fit to find the lane boundary.
- Determine the curvature of the lane and vehicle position with respect to center.
- Warp the detected lane boundaries back onto the original image.
- Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

Rubric Points:

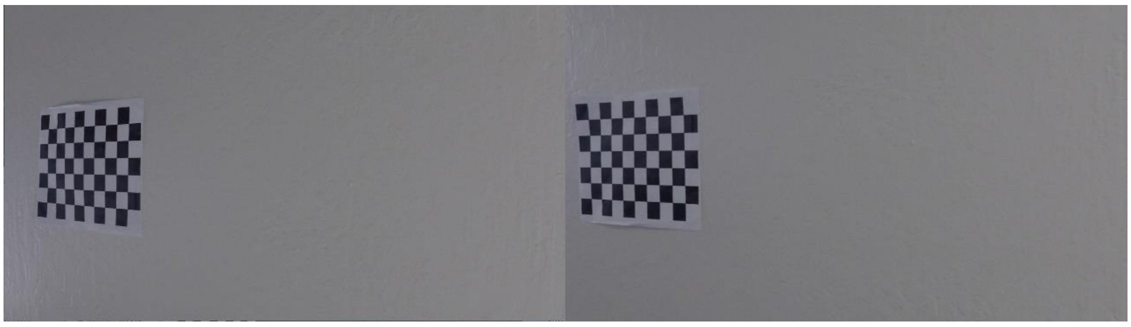
Entire Code is

1. Camera Calibration:

I have done following things using openCV Library for Camera Calibration:

I. Converted Image to gray scale using cv2.cvtColor()
II. Get Corner of the Chessboard Image using cv2.findChessboardCorners()
III. Get imgPoints and objPoints of the images
IV. Draw this corner points using cv2.drawChessboardCorners()
V. Calibrate images using cv2.calibrateCamera() , objpoints and imgpoints and get mtx, dist points
VI. Get undistorted image using cv2.undistort()

Here are the images left image is the original and right image is after distortion

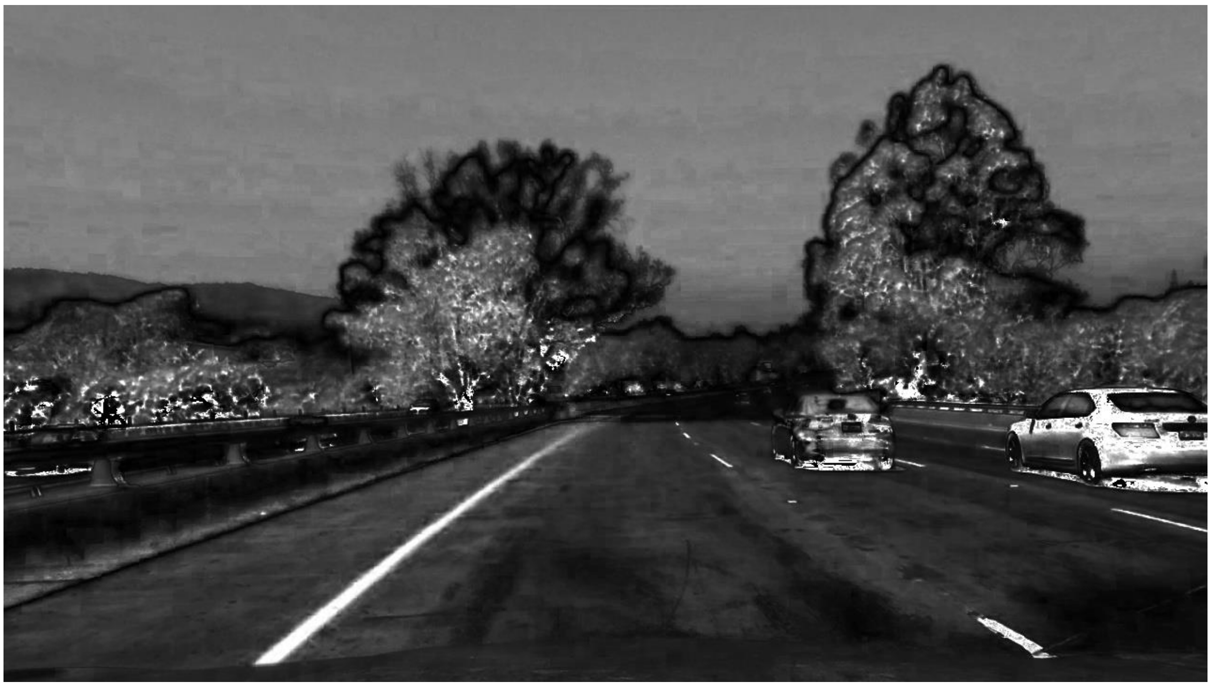


2. Colour Transformation and Gradient:

Following are the steps for Colour Transformation:

I. I have undistorted the image using cv2.undistort()
II. Converted RGB image to HLS using cv2.cvtColor()
III. Use S channel to HLS because it is more affective to detect lane lines

After Color Transformation output will look like this,



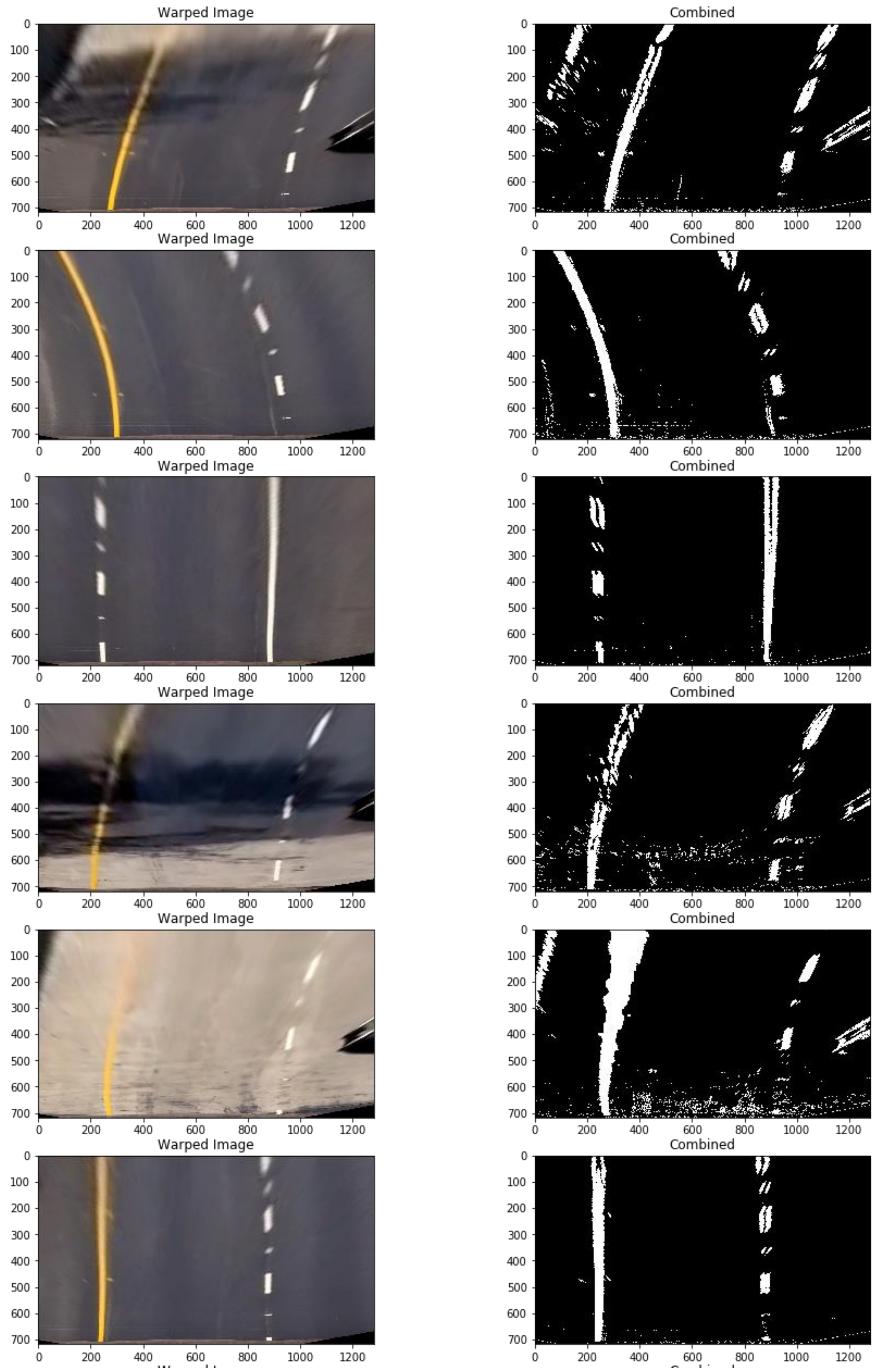
Gradient Transformation,

In this notebook I have implemented 'SobelX', 'SobelY', 'Magnitude', 'Direction' and 'Combined' transformations .

For the Project purpose I have tried combinations of different Transformation to get accurate Lane Detection

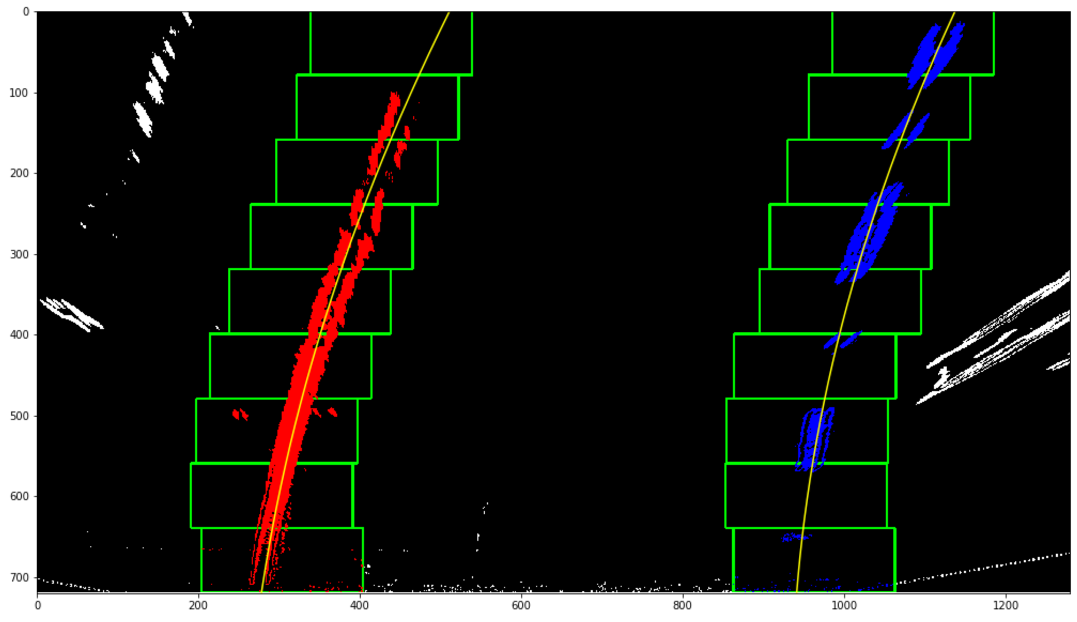
3. Perspectives Transformation:

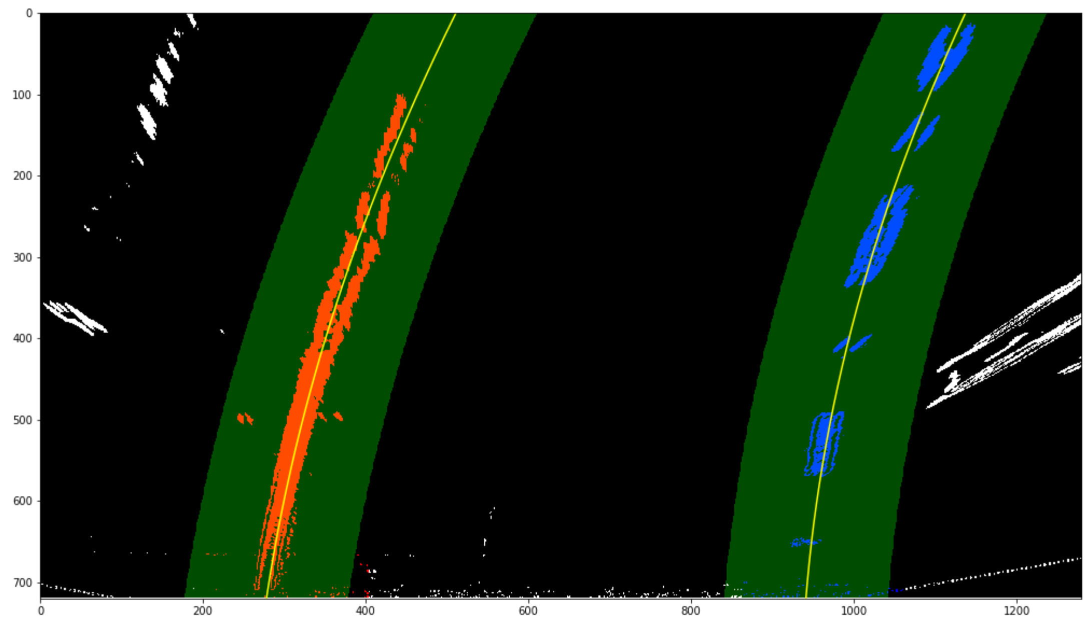
First I got Source and Destination points for perspective transformation. I use cv2.getPerspectiveTransform() to get Perspective Translation. Then I use cv2.warpPerspective() to get warp image. Below I have shown original and warped images.

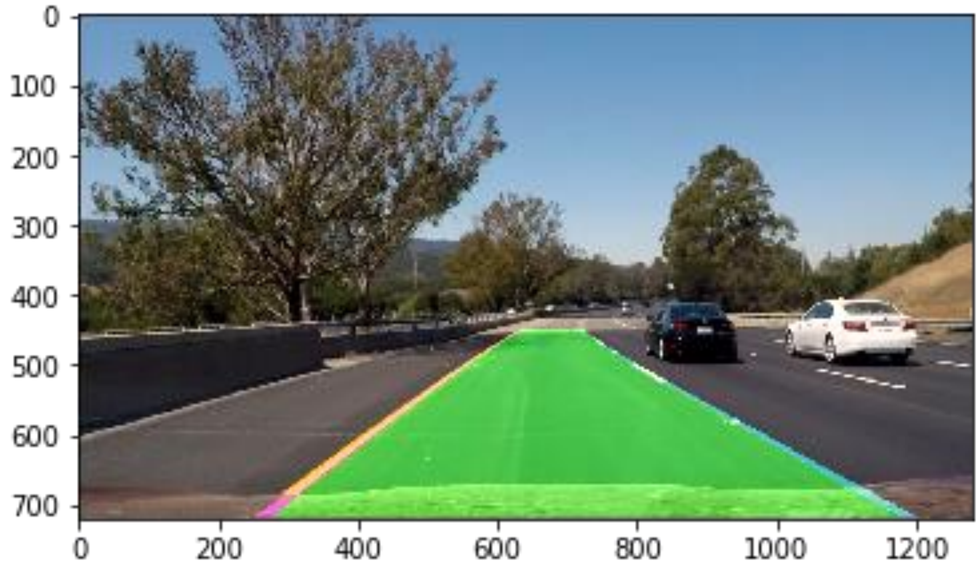


4. Lane Pixel Identification and fixing polynomial to lane: I have followed following steps:

1. Make a binary and transform image
2. Take a histogram of the bottom half of the image
3. Created Window
4. Extract left and right line pixel positions
5. Created Second Order polynomial
6. Draw Lane on Image







5. Radius of curvature of the lane and the position of the vehicle:

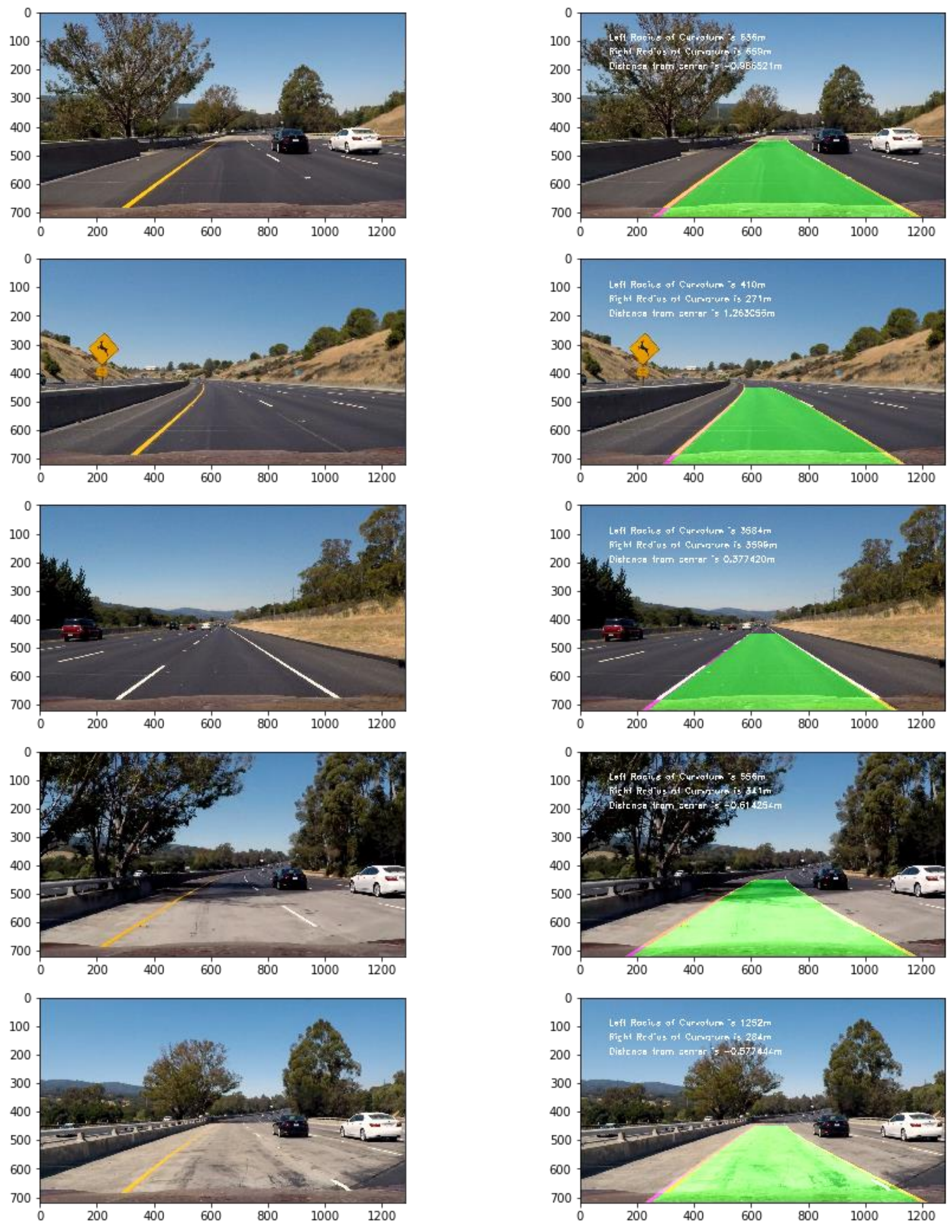
For calculate the curve I have used function shown in the CalculateRadiusOfCurvature () and find curve for left and right lanes using the same function in meter.

To find the vehicle position on the center:

- Calculate the lane center by evaluating the left and right polynomials at the maximum Y and find the middle point.
- Calculate the vehicle center transforming the center of the image from pixels to meters.
- The sign between the distance between the lane center and the vehicle center gives if the vehicle is on to the left or the right

1. **Plot Results on the Image**: I have defined one pipeline containing all the steps for image transformation mentioned in the notebook in the comment. After getting left and right curvatures and vehicle position we can plot it on the image using openCV library.

Here Left Images are original Images, while Right side images are After Result of Transformation using the Pipeline.


Left Radius of Curvature is 636m
Right Radius of Curvature is 659m
Distance from center is -0.986521m
Left Radius of Curvature is 410m
Right Radius of Curvature is 271m
Distance from center is 1.263056m
Left Radius of Curvature is 3584m
Right Radius of Curvature is 3596m
Distance from center is 0.377420m
Left Radius of Curvature is 556m
Right Radius of Curvature is 341m
Distance from center is -0.614254m
Left Radius of Curvature is 1252m
Right Radius of Curvature is 284m
Distance from center is -0.577444m

2. Video Output: I have got the images from video and passed it in to the pipeline and converted back to the video using moviePy lib and saved the video file in the specific location.

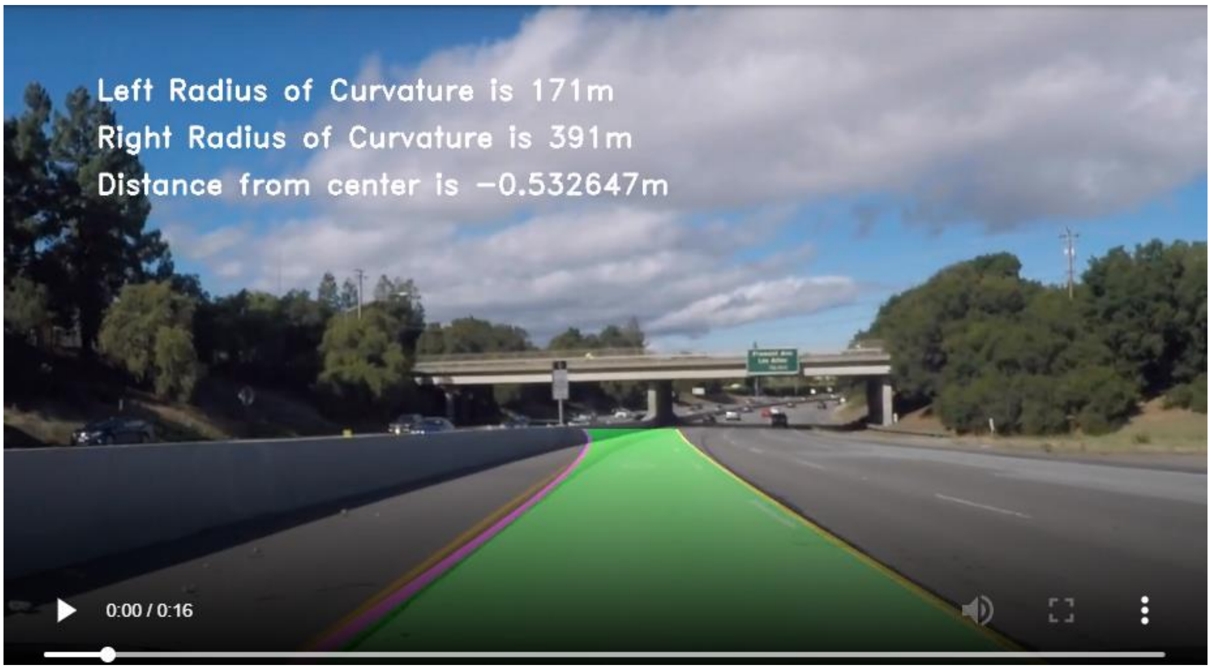
Discussion:

Problem I faced During this Project:

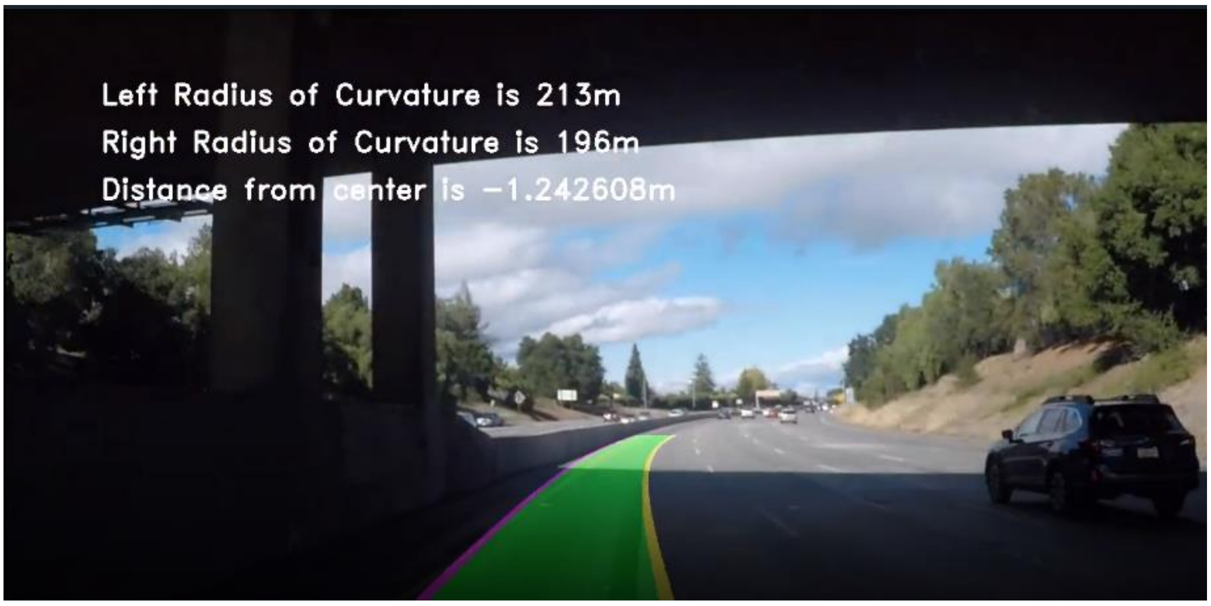
In the video I was not getting clear output earlier, it was not showing lane clearly under different brightness condition. So for solving that problem I tried different colour and Gradient Transformation Techniques but it was not very effective. Then I tried changing different parameters in the Lane Detection section then it was working great.

Where Pipeline might fail:

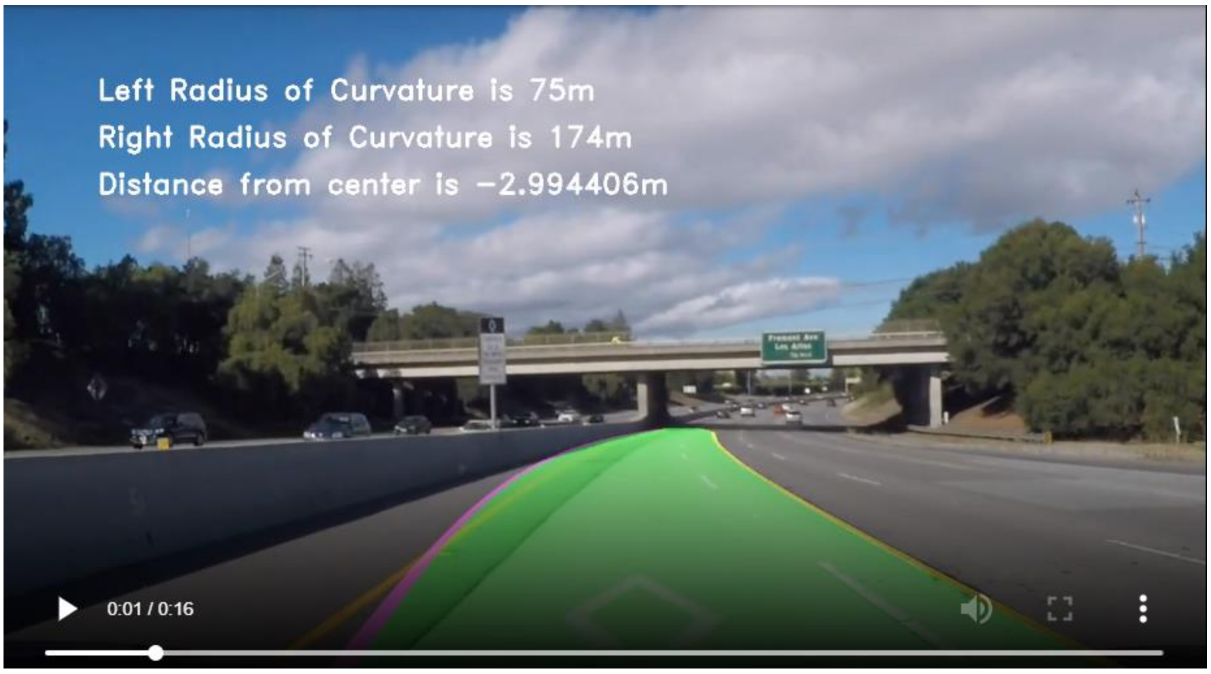
I tried implementing challenge_video, in that video I came to know that my pipeline might fail in detecting lane which is away from car


Left Radius of Curvature is 171m
Right Radius of Curvature is 391m
Distance from center is −0.532647m

I have also detected that under very dark condition (Under the Bridge) my pipeline was not able to identify pipeline


Left Radius of Curvature is 213m
Right Radius of Curvature is 196m
Distance from center is −1.242608m

My pipeline is identifying other lines in the video instead of Lane Line


Left Radius of Curvature is 75m
Right Radius of Curvature is 174m
Distance from center is −2.994406m
0:01 / 0:16

How I will Overcome this Issues:

We have to detecting the Lane lines in any conditions, So I will have to make image Lane lines more readable to Pipeline, I need to apply several Channels and Colour Transformation Techniques, I will Combined it, fine tune the parameters.

I have to consider few validation criteria can be used to remove incorrect lines.