Self-Driving Car Engineer Nanodegree Program

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

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. Here is a template writeup for this project you can use as a guide and a starting point.

You're reading it!

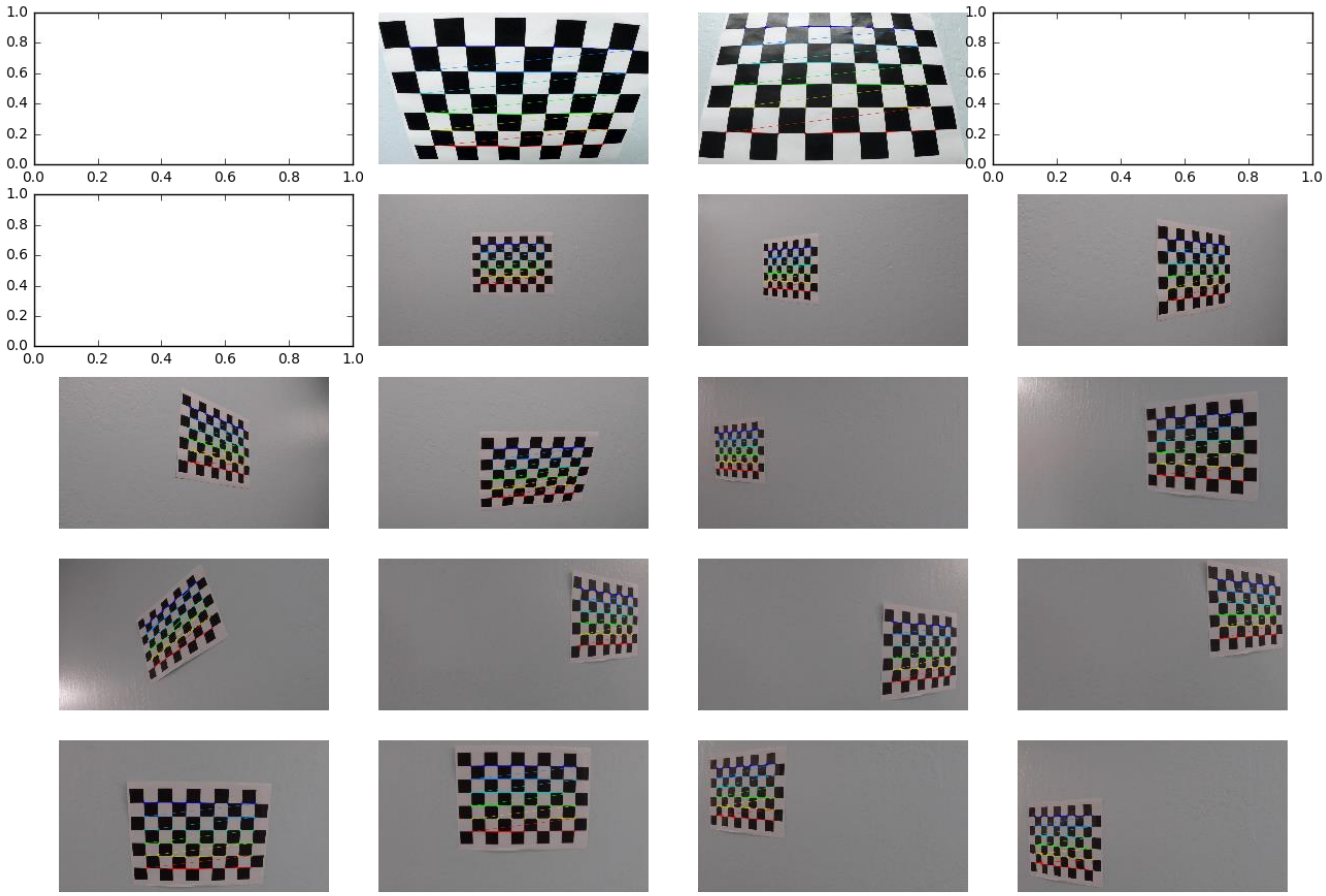
Camera Calibration

1. **Briefly state how you computed the camera matrix and distortion coefficients. Provide an example of a distortion corrected calibration image.**

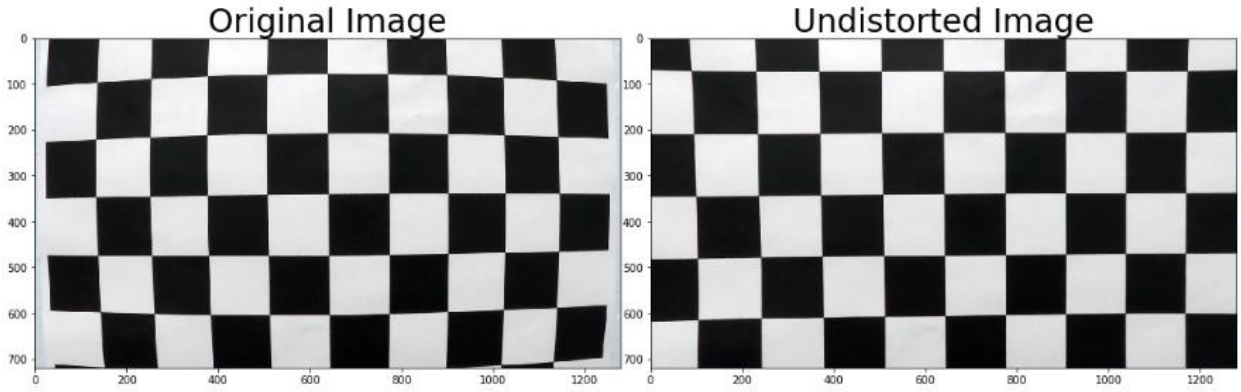
Function to Calibrate Distortion of Camera images caused by the bending light rays with the lens associated with the camera. Here some random camera photos of Chessboard have taken in understanding the light rays bending positions in Image and Calibration has done with the use of OpenCV findChessboardCorners and CalibrateCamera functions.

Function was written to work as generic for all cameras calibration works, with respect to the Calibration images directory. Function will read image by image in the specified folder and extract the image points using `findChessboardCorners`.

Some of the chessboard images don't appear because findChessboardCorners was unable to detect the desired number of internal corners.



The image below depicts the results of applying `undistort`, using the calibration and distortion coefficients, to one of the chessboard images:

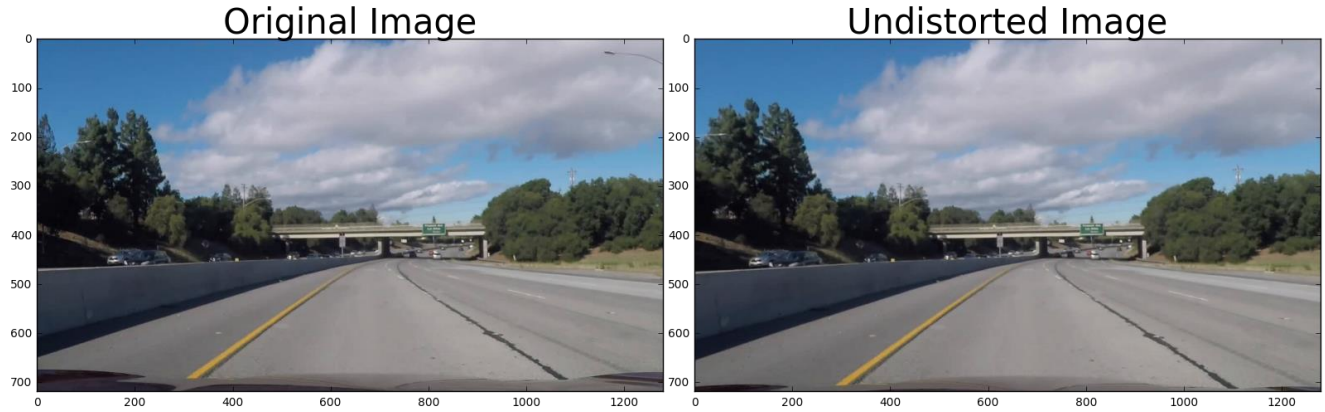


Pipeline (single images)

1. Provide an example of a distortion-corrected image.

Transforming the concentrated Source points image to Destination points Image. Depends on the Source points Coordinates given, Perspective transformation image can be genrated with the use of OpenCV.getPerspectiveTransform

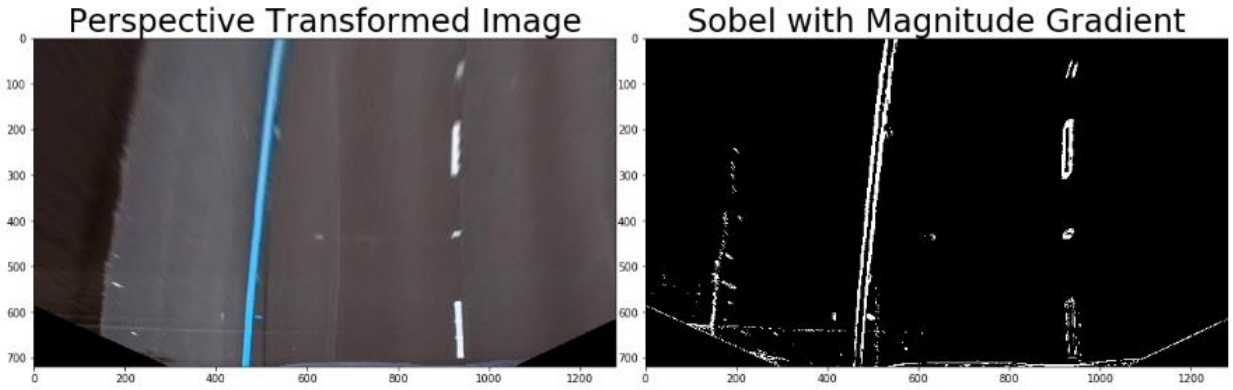
Function.



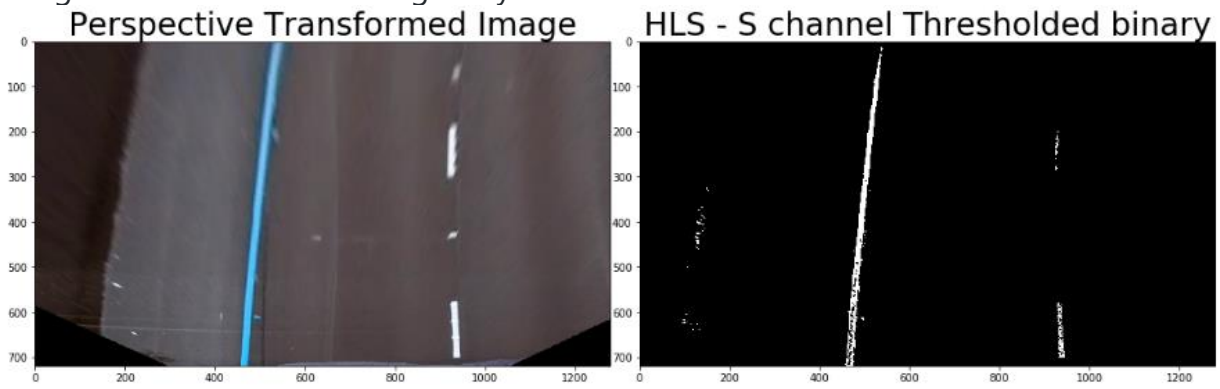
2. Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.

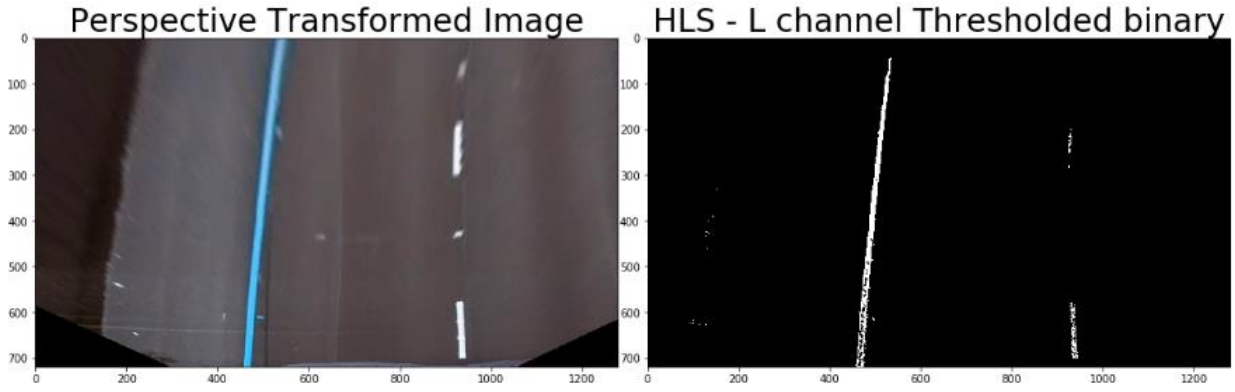
Function applies Sobel x and y, then computes the Absolute values of Soble x direction of the gradient and applies a threshold values to filter the pixels from image.

Threshold ranges have changed manually till getting the proper fit of values to show case the lanes

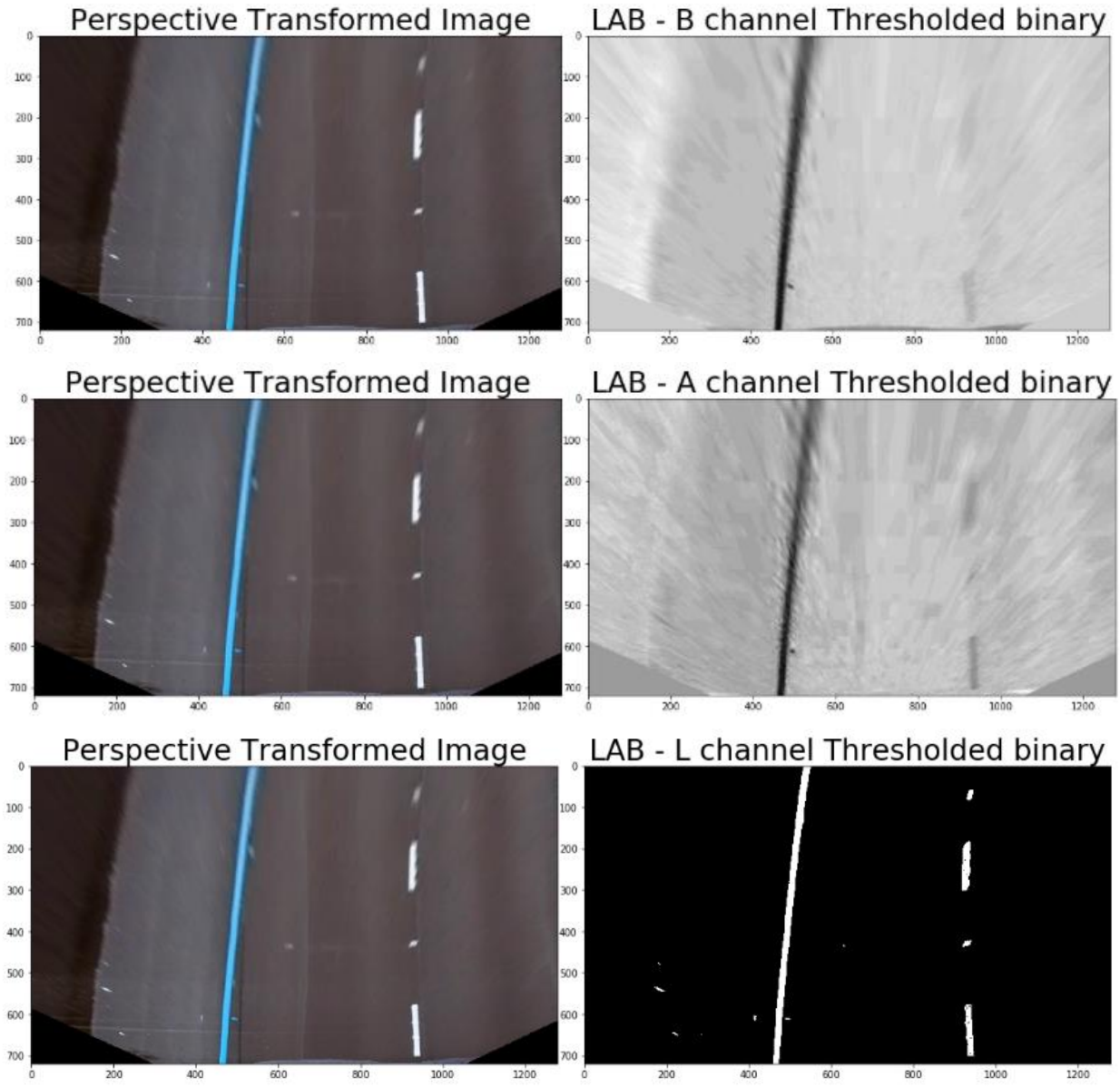


Several combinations of filters thresholds and the channels of HLS and Lab color spaces images have printed and observed manually to check out the best fit of Channel in using it for further lane finding analysis.





After distinguish observations of several channels L channel of the HLS color space has chosen to isolate white lines and the B channel of the LAB colorspace to isolate yellow lines.



Below are the results of applying the binary thresholding pipeline to various sample images:

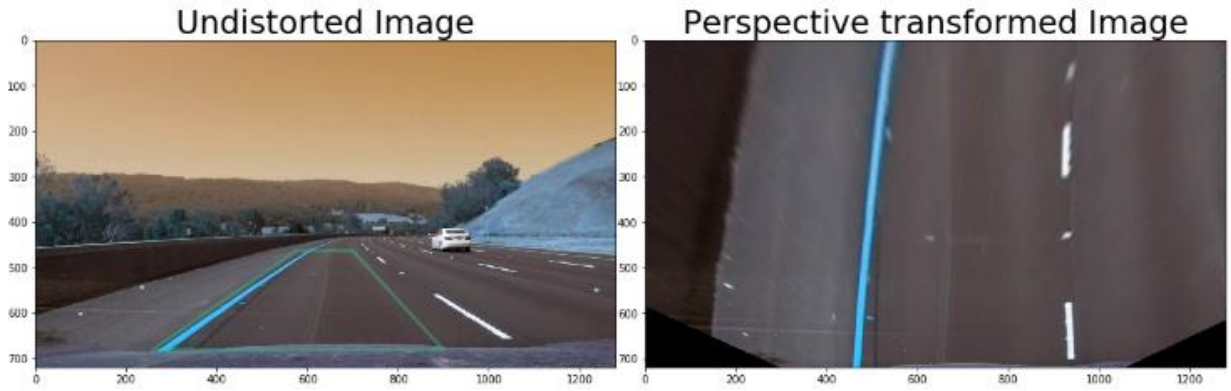
3. Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.

PerspectiveTransform named function has used in applytin perspective transform to Image Transforming the concentrated Source points image to Destination points Image. Depends on the Source points Coordinates given, Perspective transformation image can be genrated with the use of OpenCV.getPerspectiveTransform Function.

The function takes arguments of Image which to be applied with Perspective transoform, source points and destination points how it should be modified.

I have changed the source points in several ways in finding perfect fit of lane and end up with the below mentioned source points.

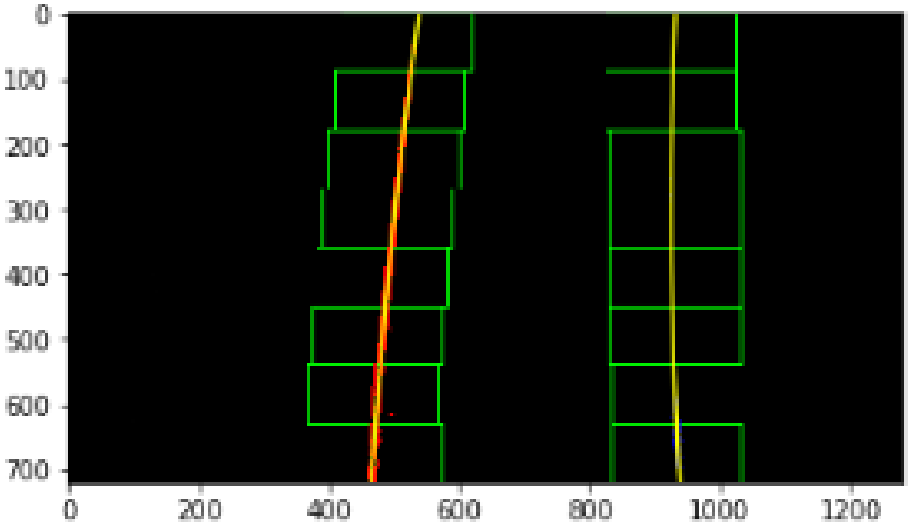
```
# define source points
SourcePoints = np.float32([(575,464),
                           (700,464),
                           (258,682),
                           (900,682)])
# define Destination points
DestinationPoints = np.float32([(450,0),
                                (w-450,0),
                                (450,h),
                                (w-450,h)])
```

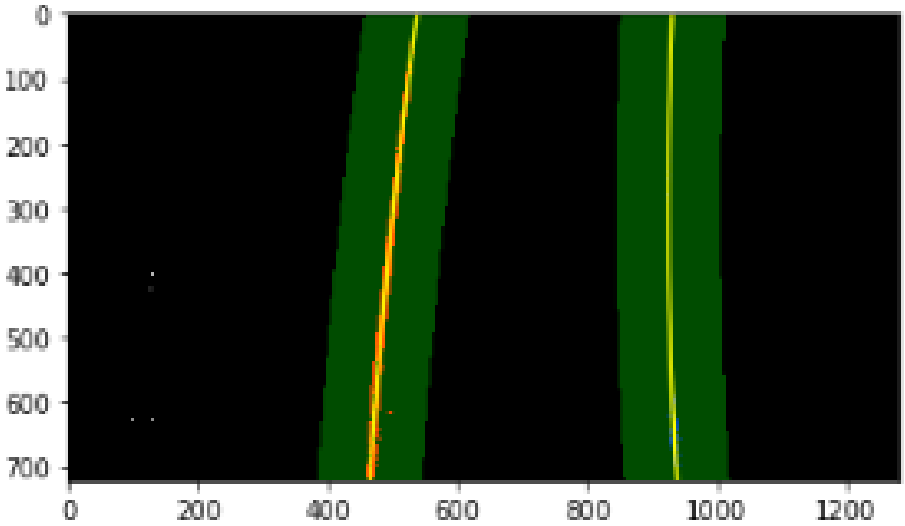


4. Describe how (and identify where in your code) you identified lane-line pixels and fit their positions with a polynomial?

Function in sliding window template across the image from left to right and any overlapping values are summed together creating the convolved signal. The functions `find_lane_lines` and `polyfit_using_prev_fit`, which identify lane lines and fit a second order polynomial to both right and left lane lines. Search_around_poly function takes inputs of Perspective transformed binary image.

Function applies sliding window of Operation in selecting the image window by window in a for loop from bottom to top in recognizing the line pixels concentration from bottom to top based on the selected number of Window size. The selection of pixels is also depends on the threshold considered. Indexes associated with lane lines will be retrieved with the filter conditions chosen.
Once after differentiating the good fit lane lines of left and right lane lines, with the use of numpy poly fit function second order polynomial equation can be applied on the lines. The window by window well fitted lane lines were appended to the variables for further requirement of lane finding.





5. Describe how (and identify where in your code) you calculated the radius of curvature of the lane and the position of the vehicle with respect to center.

CurvatureCalculationDistancefromLane function is used at here in calculating the curvature associated with the lane line.

Function in Calculating the Curvature of lane lines with the use of Second order polynomial Equations coeffiecients. For converting the pixel space to Meters, assumptions have made in calculating the properional pixel to meter space. Once after the pixel space covertion with the use of second order polynomial coeffiecients distance and radius infomration updated

```
left_curverad = ((1 + (2*left_fit_cr[0]*y_eval*ym_per_pix + left_fit_cr[1])**2)**1.5) / np.absolute(2*left_fit_cr[0])

right_curverad = ((1 + (2*right_fit_cr[0]*y_eval*ym_per_pix + right_fit_cr[1])**2)**1.5) / np.absolute(2*right_fit_cr[0])
```

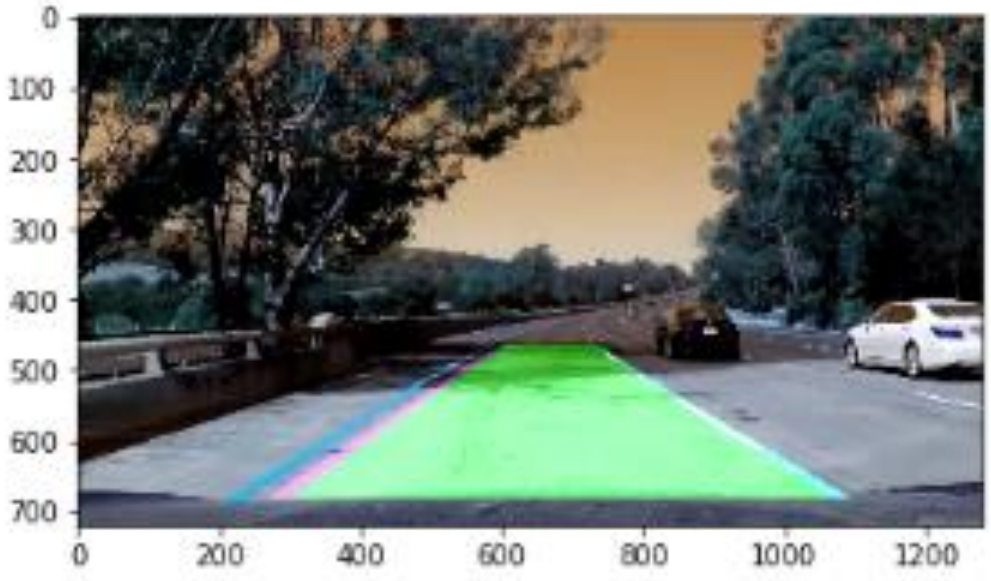
left_curved and right_curved variables are used in storing the curvature information from the second order polynomial equations. With the use of second order polynomial coefficients radius of left curve, radius of right curve and center of the curvature is calculated with the formulaes given.

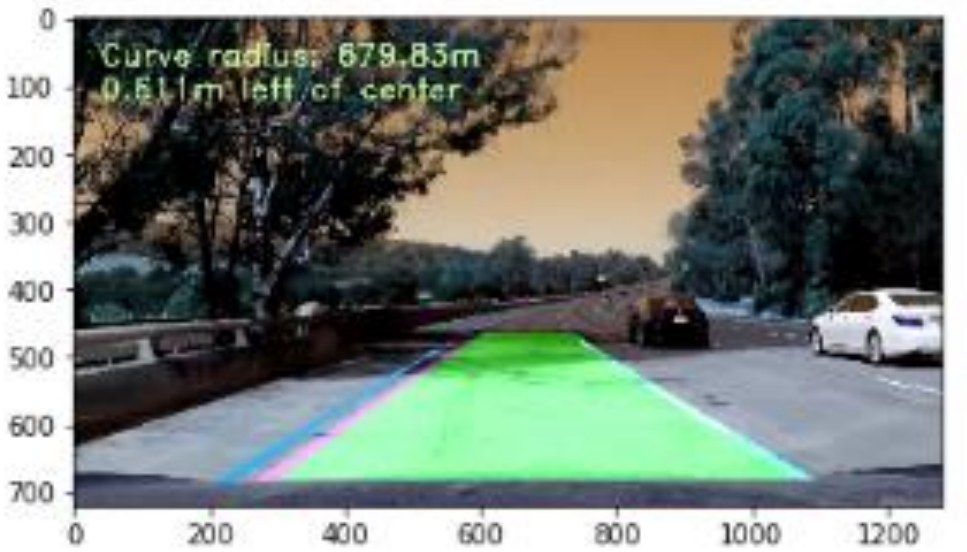
```
left_fit_x_int = left_fit[0]*h**2 + left_fit[1]*h + left_fit[2]

right_fit_x_int = right_fit[0]*h**2 + right_fit[1]*h + right_fit[2]
```

6. Provide an example image of your result plotted back down onto the road such that the lane area is identified clearly.

LaneImpositionOnImage is used in drawing the lane in on the selected image. Lane area is highlighted with the light green color. Code associated with writing lane is referred from the lesson-9 of Advanced Computer vision. A polygon is generated based on plots of the left and right fits, warped back to the perspective of the original image using the inverse perspective matrix `Minv` and overlaid onto the original image. The image below is an example of the results of the `draw_lane` function.





Pipeline (video)

1. Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (wobbly lines are ok but no catastrophic failures that would cause the car to drive off the road!).

Here's a link to my video result

Discussion

1. Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

The main problem that I faced is to prepare well fit of thresholds in extracting Lanes information form image. The lighting conditions and the yellow lanes information in between the road is irrelevant which I was not able to cover well in Extracting the lanes.

While providing test images, please try to provide atleast some of difficult images in fitting the lanes. The sequence of lanes fitting information is getting changed by the image by image in between. I have tried my level best in reducing the lanes irrelevant fitting information.

The pipeline likely to fail on the roads adjacent to road edges low light conditions and adjacent vehicle crossing some times.

I would like to build a adjustable Neural network in finding the correct of fit thresholds accurately , than lot of manual work of observation can be automated by Deep learning neural network and we can expect good results in predictions.

I have faced difficulty in understanding the class initialization variables, I could not find more information in utilizing the variables effectively in building a robust model in finding lanes.