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.

Image References :

Edge Point Calibrated for all the images

https://view5639f7e7.udacity-student-workspaces.com/view/CarND-Advanced-Lane-Lines/output_images/calibration_edge_pts.jpg

Results of various colour channels,
https://view5639f7e7.udacity-student-workspaces.com/view/CarND-Advanced-Lane-Lines/output_images/color_spaces.jpg

Collated all test images in one image - undistored, unwarped, combined L&B channels

https://view5639f7e7.udacity-student-workspaces.com/view/CarND-Advanced-Lane-Lines/output_images/overall_output.jpg

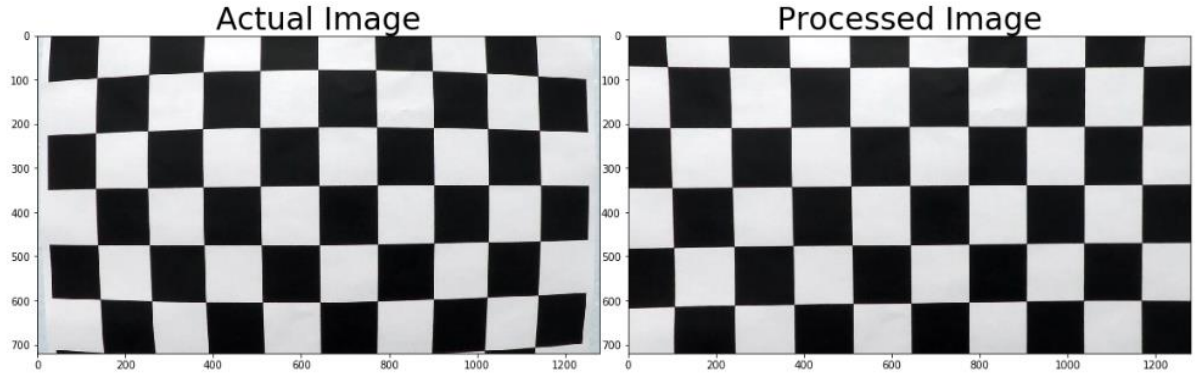
Rubric Points :

Camera Calibration

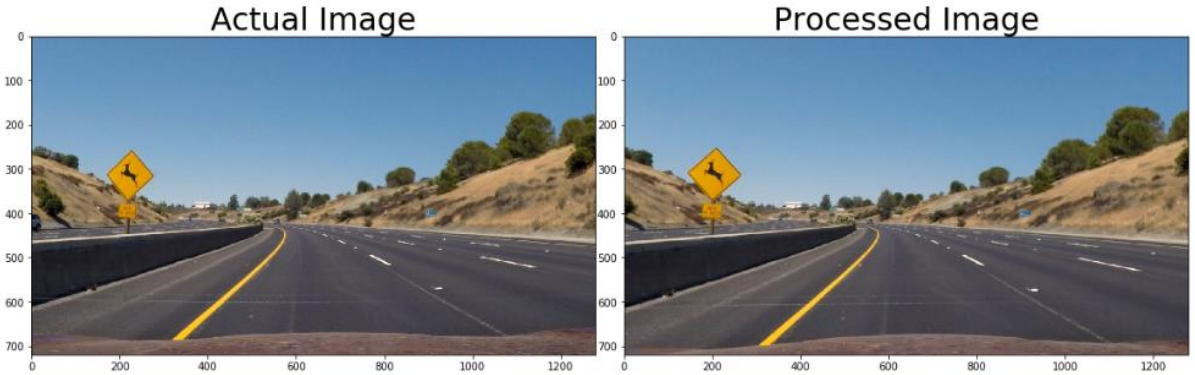
1. Briefly state how you computed the camera matrix and distortion coefficients. Provide an example of a distortion corrected calibration image.

Fixed the size as (9,6) for findChessboardCorners and identified the corners. Based on the criteria, layed the edges over the chess board.

With the edges as image points, using calibrateCamera method, i arrived the distortion coefficients. Based on the distortion coefficients, applied undistort to the image to display the undistorted image.

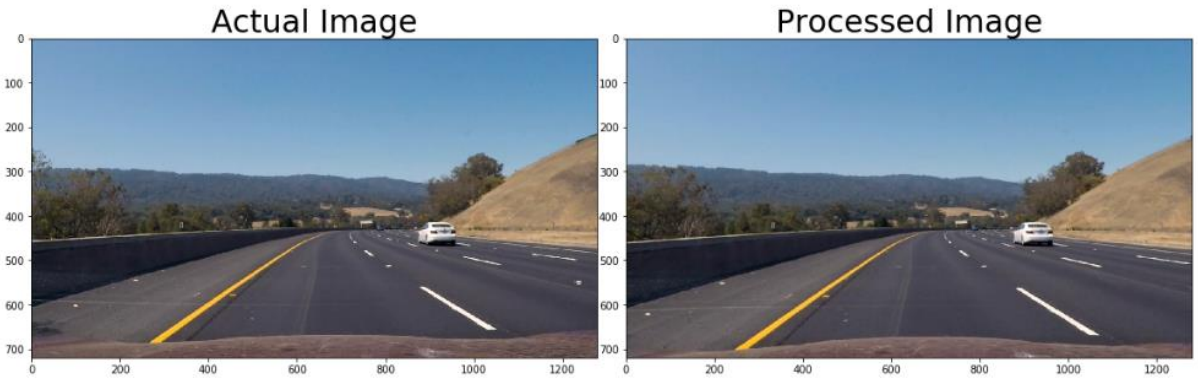


With the coefficients, I was able to undistort the test image, below are the samples

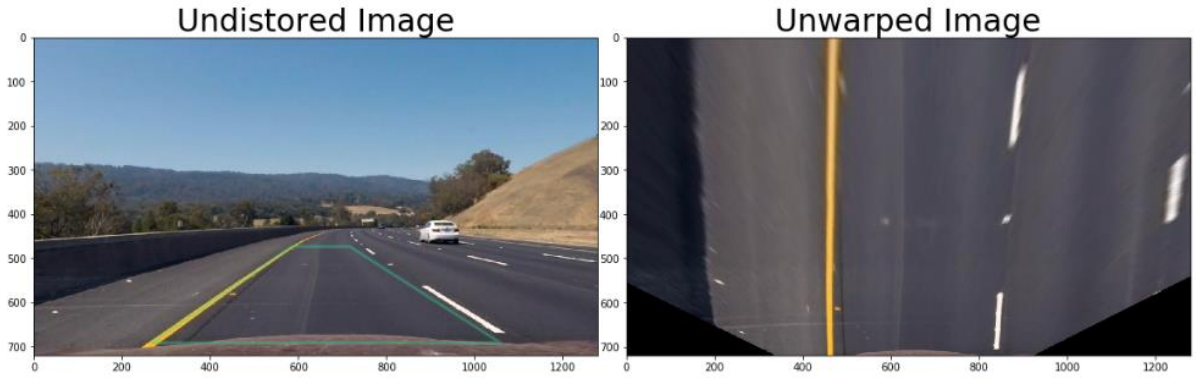


Pipeline

1. Un distort the image



2. Perspective Transformation

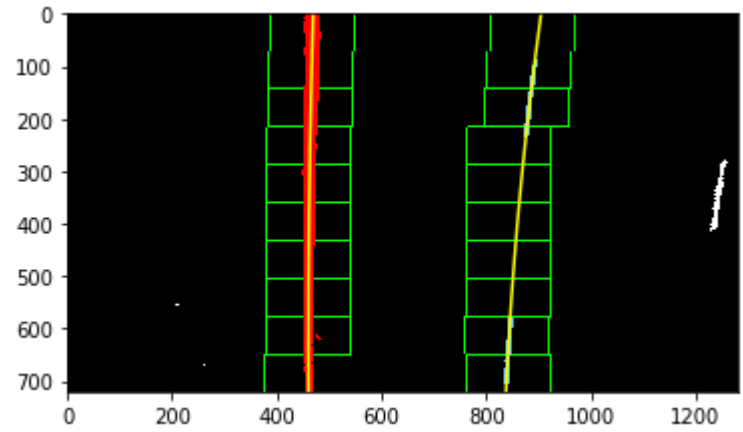


3. Color Space Exploration

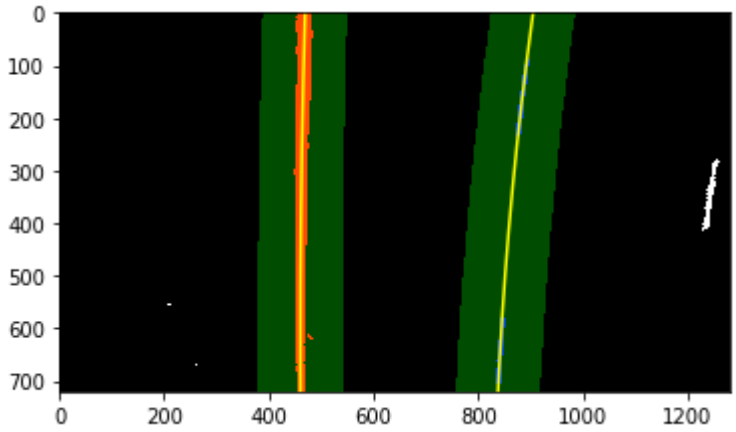
I was exploring variour color spaces like RGB, HLS, HSV & LAB. Identified S-Channel, L-Channel & B-Channel will be a good fit. Based on various combinations of channels & threshold corrections, I decided to combine L-Channel & B-channel to extract the lane lines better.

4. Polynomial Fit for finding lane lines

I was using sliding window function to identify and overlay windows to predict lane line direction like below.



Also used polyfit using previous frame (apply_polyfit) function to predict & lay the lane lines like this,



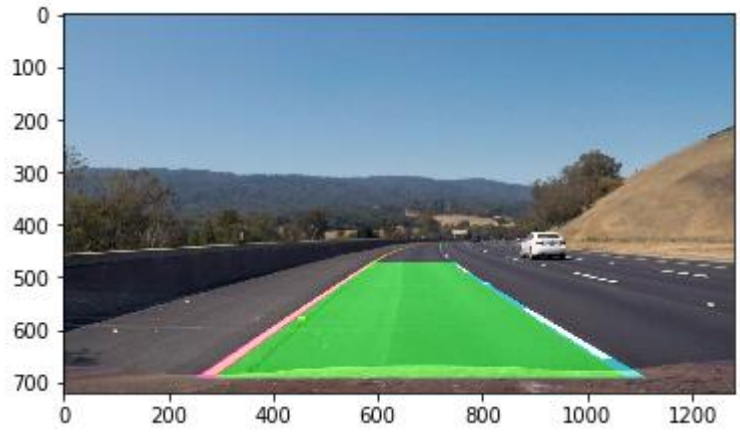
5. Calculate Curvature Radius

By using calculate_curvature_radius function, with left & right fit data and indexes data of the image, I was able to arrive the radius of curvature for the above the test image as,

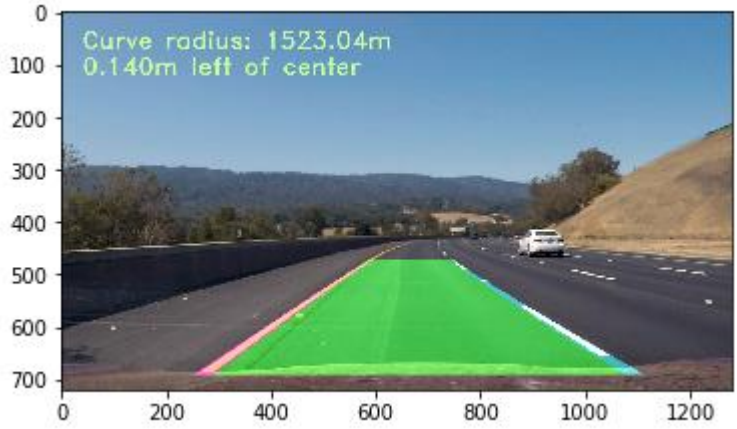
```
Radius of curvature : 796.819852914 m, 2249.25746582 m
Distance from lane center : -0.140387795565 m
```

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

Using the fill_lane_space function, I plotted the lane area on the test image like below,



Added the additional info like below,



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!).

https://view5639f7e7.udacity-student-workspaces.com/view/CarND-Advanced-Lane-Lines/project_video_output.mp4

Discussion

1. Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail?

Problems I faced was with respect to the color variations, shadows, etc, that made the threshold correction a tedious process. Spent a lot of time there. During various color space exploration, extraction & combination of different channels was a challenging one. I was assuming the lane width as a fixed one and deriving at the left & right fits.

What could you do to make it more robust?

If I was able to dynamically threshold the color spaces, the algorithm would have been more robust in various lighting conditions.