Project : Advanced Lane Finding

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.

Project Rubric :

Project Rubric

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](https://github.com/udacity/CarND-Advanced-Lane-Lines/blob/master/writeup_template.md) is a template writeup for this project you can use as a guide and a starting point.

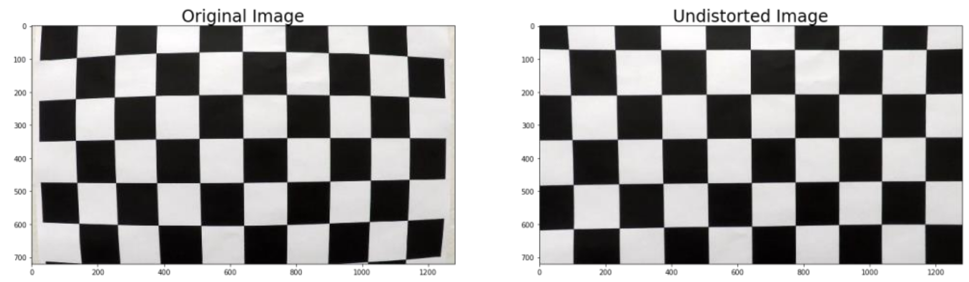
You're reading it!

Camera Calibration

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

The code for this step is contained in the second code cell of the IPython notebook located in "./P2.ipynb "

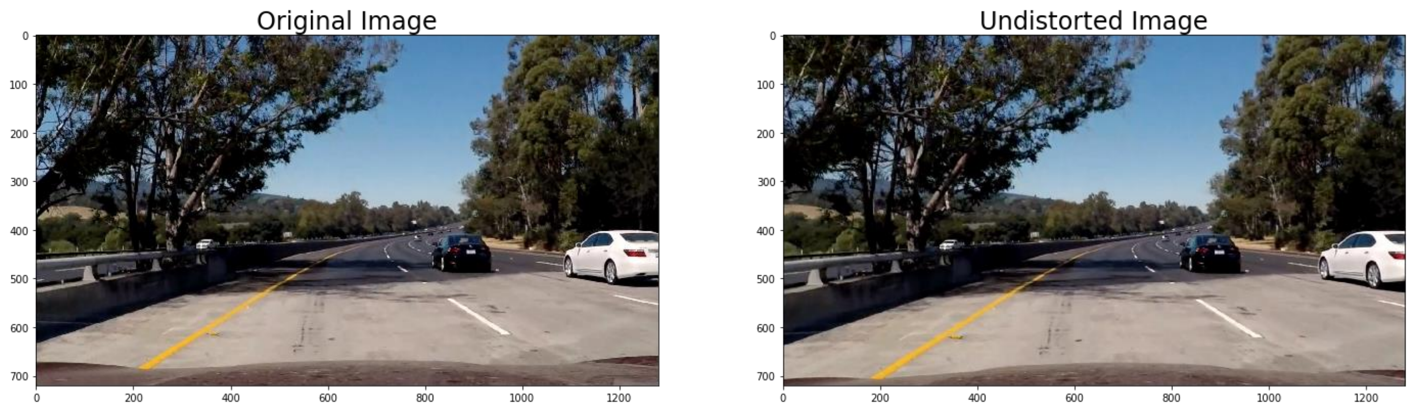
The camera matrix and distortion coefficient are found using the provided images of the chessboards. After finding all the values , any images with same camera can be distorted using **cv2.undistored**



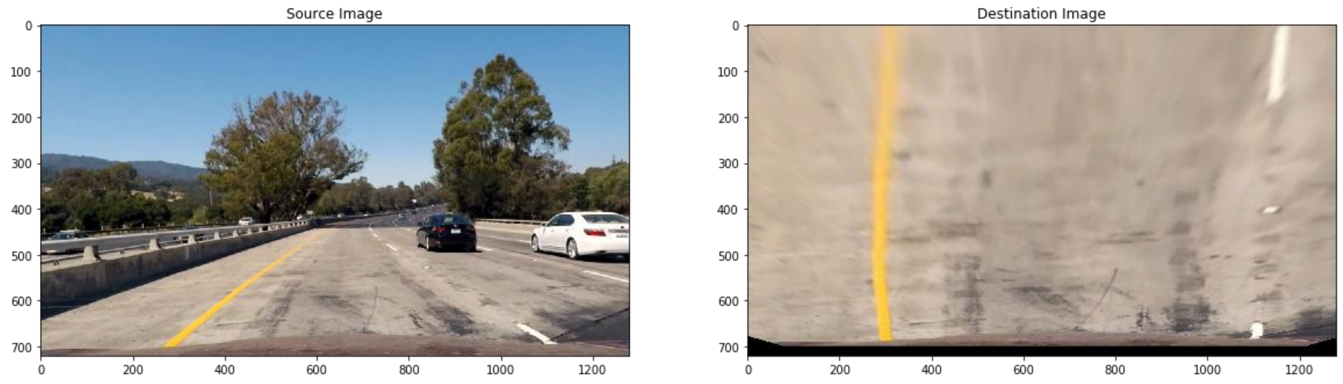
Pipeline (images)

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

To demonstrate this step, I will describe how I apply the distortion correction to one of the test images like this one:



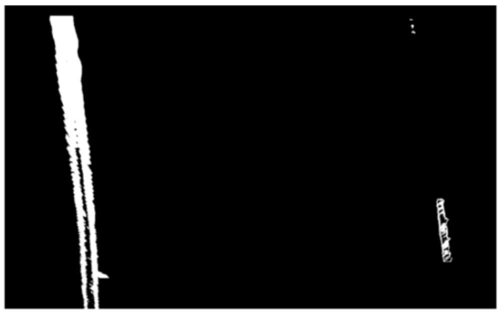
2. Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.



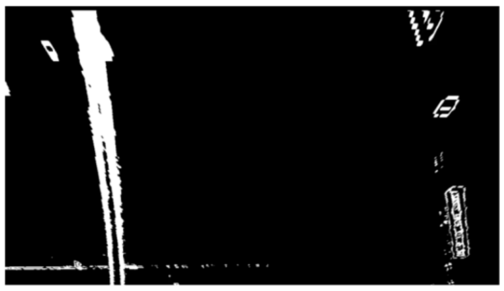
I have used **cv2.getPerspectiveTransform** and **cv2.warpPerspective**

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

Through trial and error, it was found that applying a threshold to the S channel in the HLS color space gives a good indication of where the yellow lane is located:

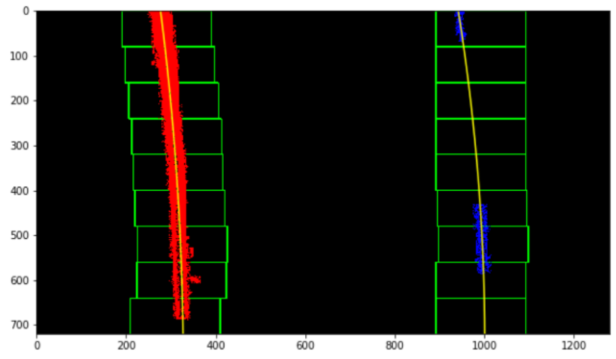


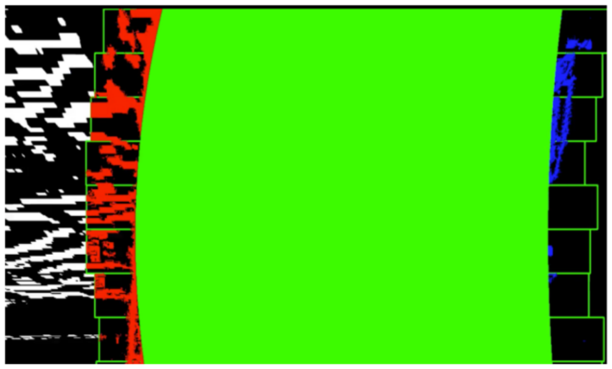
However, combining that threshold with magnitude of Sobel filter in both x and y direction, gave an even better result that highlighted both the yellow and the dashed while lane lines:



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

After obtaining a good binary representation of lane lines, the Sliding Window algorithm was applied to reconstruct the entire lane by fitting quadradic polynomials to the partial lane pixels:





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.

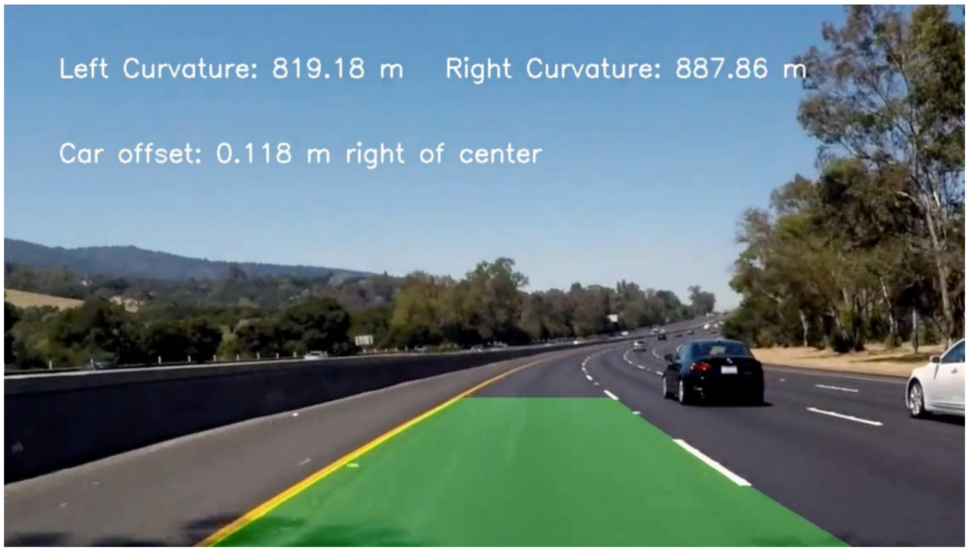
Curvature Calculation and Offset Detection:

Once fitted quadradic functions are found, curvature of the lanes can be computed using the following formula:

$$R_{curve} = \frac{[1+(\frac{dx}{dy})^2]^{3/2}}{|\frac{d^2x}{dy^2}|}$$

The radius of Curvature can be found in **measure_curvature** function in P2.ipynb in project jupyter notebook. Some of the useful concepts and formula are mentioned here

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


Left Curvature: 819.18 m
Right Curvature: 887.86 m
Car offset: 0.118 m right of center

the Project videos is saved in **project_video_output5.mp4**

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 Project video

Advanced Lane Lines

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 issue that I faced was while applying the thresholds. It was bit difficult to get the correct threshold values for each channel and combining them. I am pretty sure there's a better combination than what I am currently using.

Processing the video is taking a lot of time.

Also, I faced some issue with Jupyter notebook. My method was taking the value that was defined in the previous code block and it took a lot of time to figure out what actually was going wrong but I started from the beginning and completed the project step by step.