Readme for the Self-Driving Cars Nanodegree Program 2nd Project Advanced Lane Finding

Goals / Steps of this project:

- 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 threshholded 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 positon 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 vehicle position.

Rubric Points

Camera Calibration

- Script used: "Camera Calibration.py"

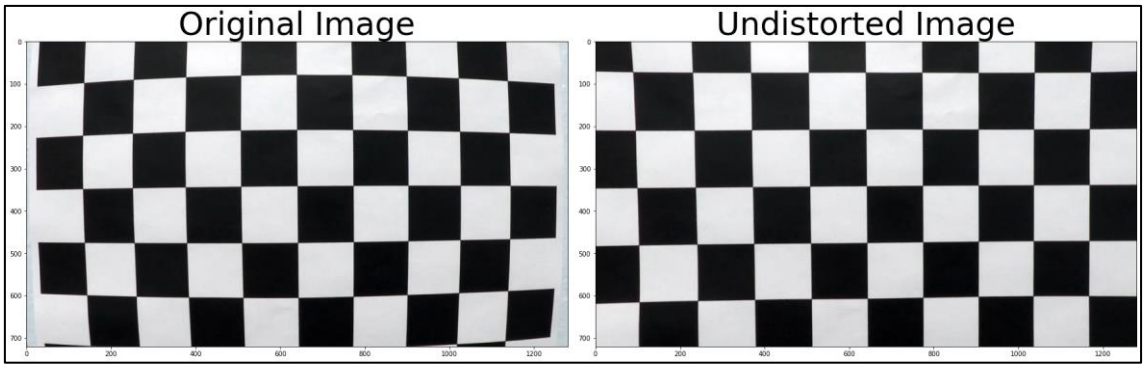
To compute the Camera Matrix and the Distortion Coefficients I created a python script, prepared the object points in the 3 coordenates (X,Y,Z) and assuming the Z=0, set the number of corners in x and y to search in the 20 chessboards (cal_images) and get the image points.

With the object points and the image points I computed the camera calibration with the cv2.CalibrateCamera() function and to undistort the images I used cv2.undistort().

We read a folder with 20 calibration images, finally we get the following camera matrix and Coefficients:

- ret = 0.9230912642921472
- mtx = [[1.15660712e+03 0.00000000e+00 6.68960302e+02]
  [0.00000000e+00 1.15164235e+03 3.88057002e+02]
  [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
- Dist = [[-0.23185386 -0.11832054 -0.00116561 0.00023902 0.15356159]]

Website consulted:
http://csundergrad.science.uoit.ca/courses/cv-notes/notebooks/02-camera-calibration.html



Pipeline (test images)

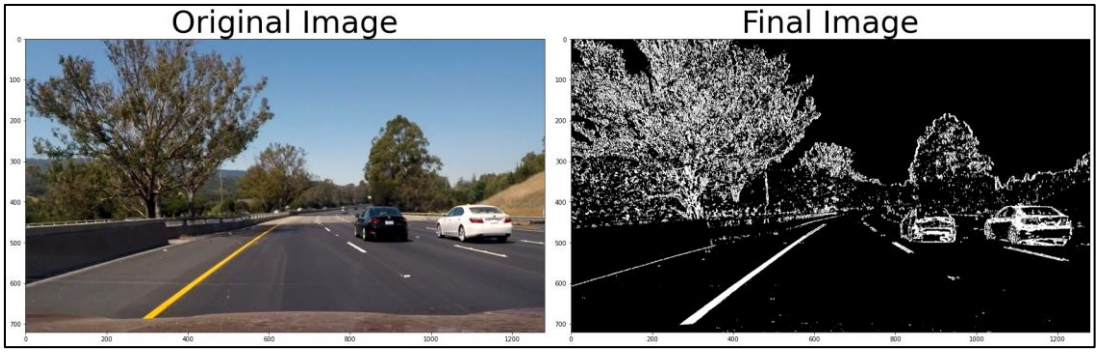
- Provide an example of a distortion-corrected image.

To Undistort the image, I needed the Camera Matrix and Distortion Coefficients got with the calibration of the camera previously done.

- 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.

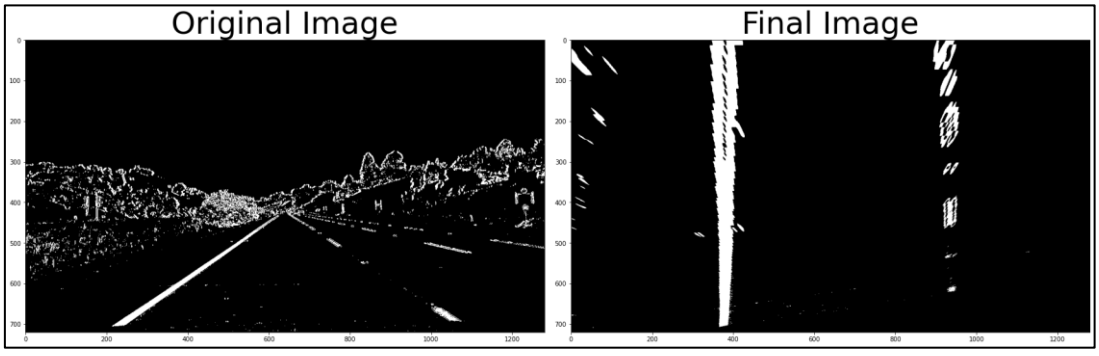
Function used in code: “binary_image”**.** Function created to get a binary image from an RGB undistorted image input.

The function consists in a conversion from RGB to HLS to compute the gradient in the x axis on the Channel L Lightness component and perform a threshold in the color channel S (Saturation).



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

Function used: warp_image. Inputs a binary image and outputs a “birds-eye” image, a Transform matrix “M” and the inverse Transform matrix “Minv”.

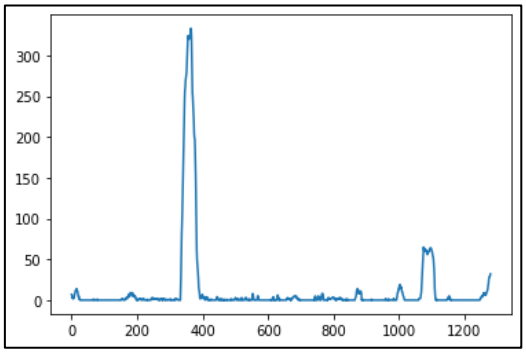


- Describe how (and identify where in your code) you identified lane-line pixels and fit their positions with a polynomial.

The Following Steps are followed:

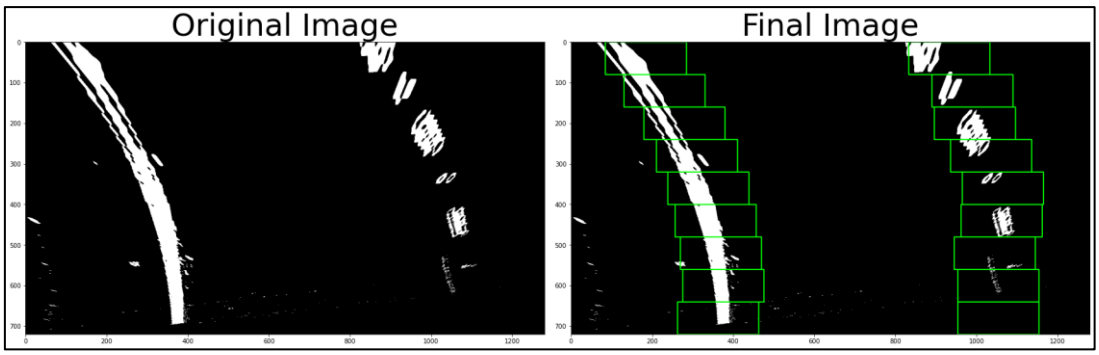
Histogram Peaks

Functions used: histogram & get_peaks. Inputs a warped image and outputs an histogram from the bottom half of the image on the y axis. “get_peaks” outputs the left_x base amd righ_x base of the lines.



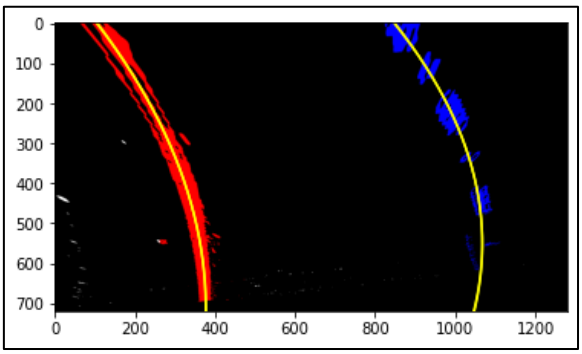
Sliding Windows

Functions used: find_lane_pixels**.** Inputs a warped image, left and right x base, number of windows, margin and min pixels. Outputs the position of the pixels in x and y of both lines.



Fit Polynomial

Functions used: fit_polynomial**.** Inputs the position pixels of both lines and outputs the left and right curves.



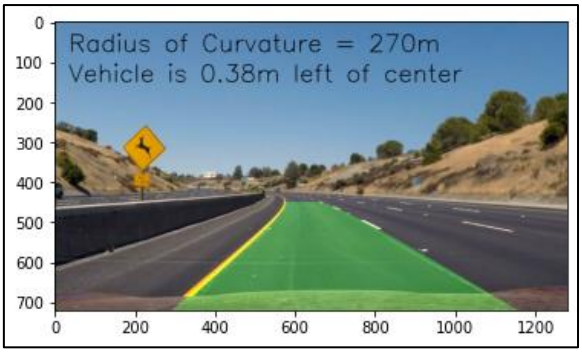
- Describe how (and identify where in your code) you calculated the radius of curvature of the lane and the position of the vehicle with respect to centre.

Functions used: measure_curvature_real & car_offset**.** Inputs the points in the left and right lines, left and right fit to perform a conversion from pixels to meters. Outputs the curvature in meters.

The offset of the car from the centre of the lane is gotten by getting the midpoint of the image (centre of the car) and computing the midpoint of the lane, having those points the rest is to perform a subtraction.

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

The following image shows the previous two rubric points.



Project Video

- 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!)

The output video could be found in the folder of the project with the name Project_video_Output.mp4

Discussion

- 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?

1. During the calibration there are 2 of the 20 Cal_images that I could not use properly, not all the corners are detected with the FindChessboardCorners Function.
2. As an improvement to my pipeline, I can Define a Class to keep tracking of the images of the video.
3. Currently my implementation does not perform a Sanity Check, the lines detections in the Project video are good and does not seem to lose track but Sanity Check would be very helpful for more complex videos.
4. Close Curves would be a problem for this implementation as well as roads with many shadow and Lights.