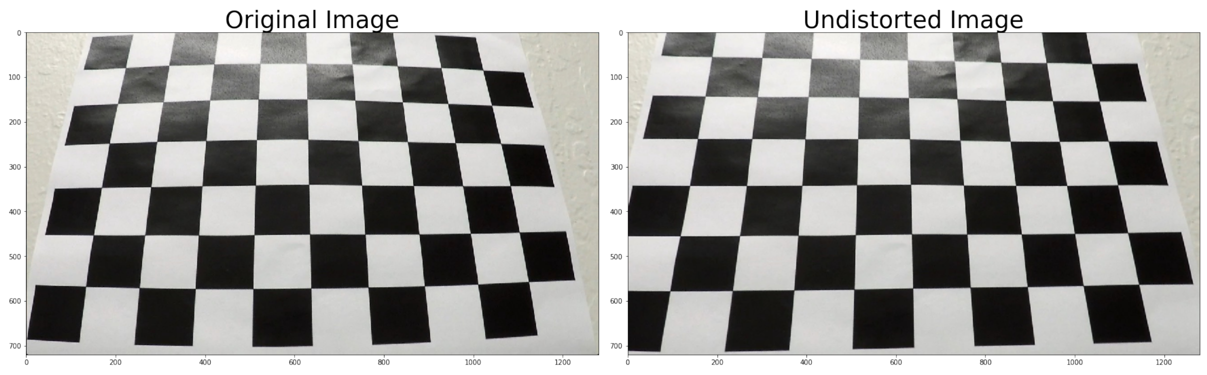
Project 2: Advanced Lane Finding

Overview

The project aims to build a pipeline to achieve advanced lane line detection, including lane lines tracking, at the same time calculating radius of curvature and providing the position of the car. The main tools I use in this project are python and OpenCV.

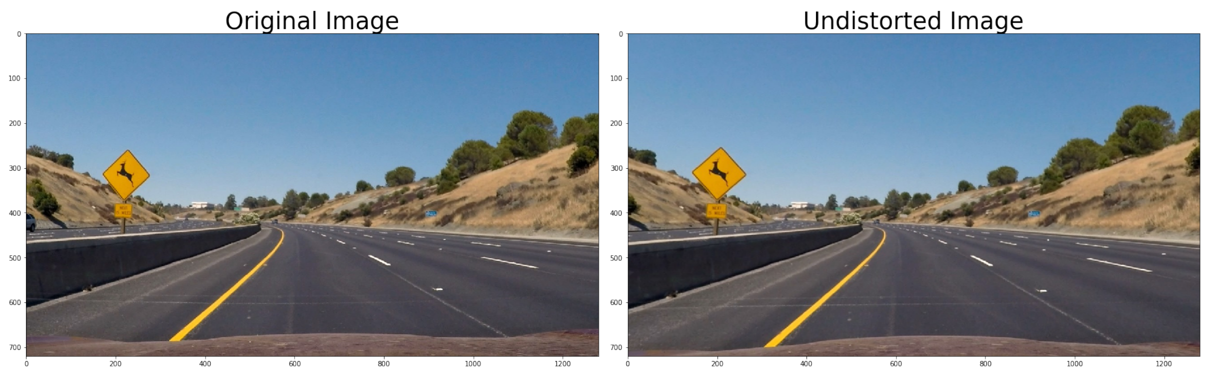
Camera Calibration

OpenCV function calibrateCamera() was used to calculate the correct camera matrix and distortion coefficients using the calibration chessboard images provided in the repository.

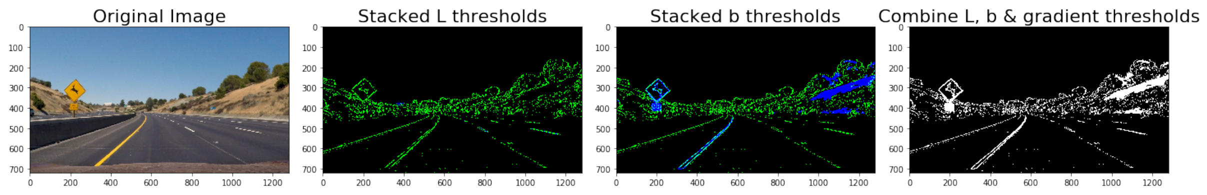


Pipeline (test images)

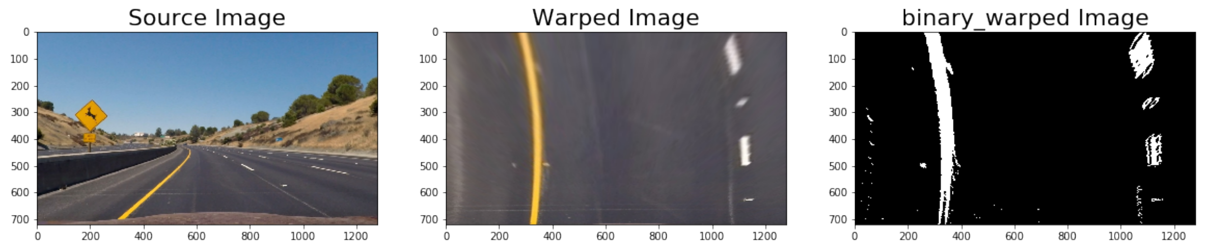
Distortion correction that was calculated via camera calibration has been correctly applied to each image.



Combination of color transforms and gradients has been used to create a thresholded binary image containing likely lane pixels.



OpenCV function warpPerspective() has been used to correctly rectify each image to a "birds-eye view".

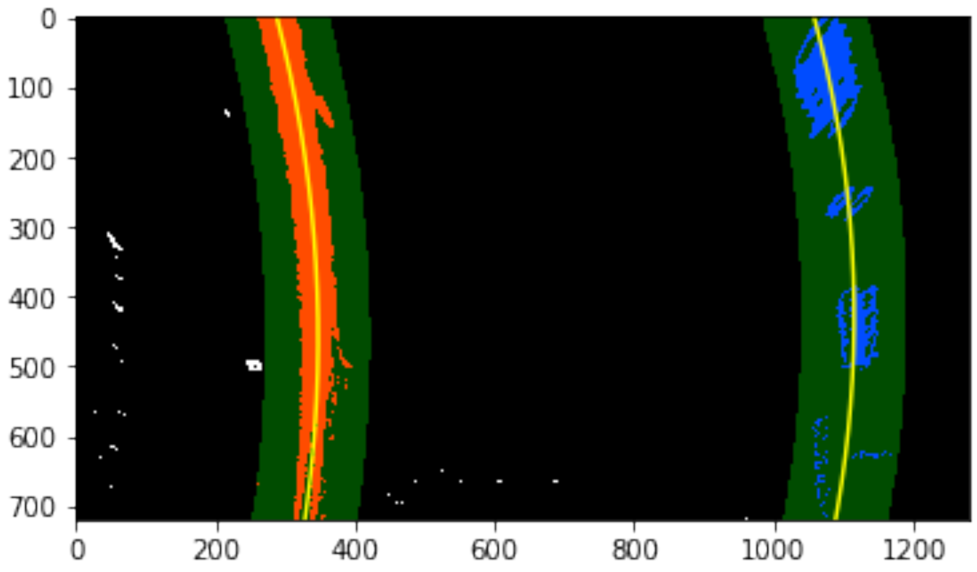


Identify lane-line pixels and fit their positions with a polynomial

After applying calibration, thresholding, and a perspective transform to a road image, I got a binary image where the lane lines stand out clearly. Then I create a histogram of where the binary activations occur across the image to decide explicitly which pixels are part of the lines and which belong to the left line and which belong to the right line. With the histogram I'm adding up the pixel values along each column in the image, so I can use the two highest peaks from our histogram as a starting point for determining where the lane lines are, and then use sliding windows moving upward in the image to determine where the lane lines go. Four steps are taken to implement sliding windows and fit a polynomial:

1. Split the histogram into two sides, one for each lane line.
2. Set up windows and window hyperparameters. set a few hyperparameters related to sliding windows, and set them up to iterate across the binary activations in the image.
3. Iterate through nwindows to track curvature.
4. Fit a polynomial

   And here is what I got



In processing video streams, I add a smoothing step to smooth the value of the current line using a first order filter response, as coeffs = 0.95*coeff_previous + 0.05*coeff_current, and add a sanity check, the lanes are discarded if the change in coefficient is above 0.0005 (this number is obtained empirically).

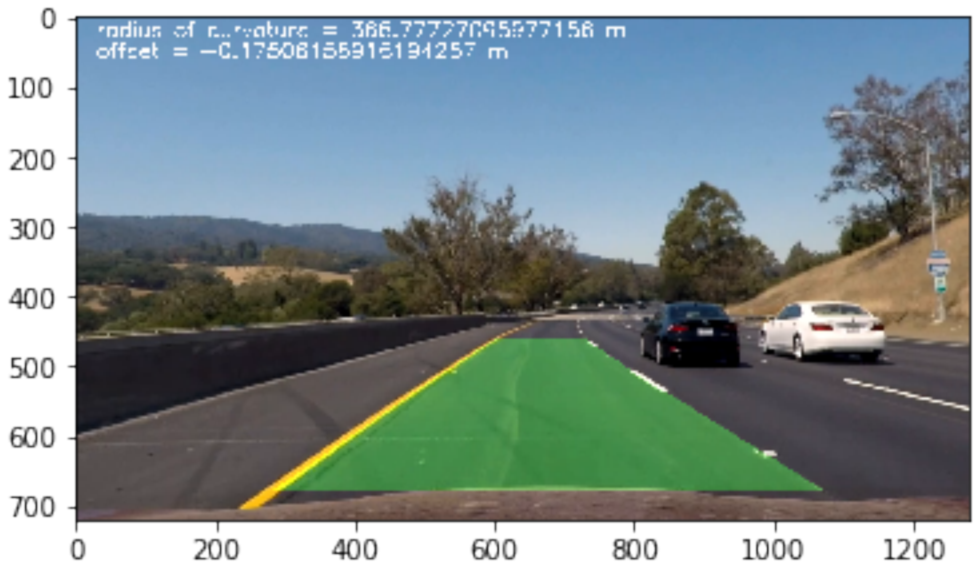
Calculate the radius of curvature of the lane and the position of the vehicle with respect to center.

After a second order polynomial curve has been fitted, f(y) $= Ay^2 + By + C$, to all the relevant pixels I've found in sliding windows in fit_poly(), the radius of curvature can be computed by $R_{curve} = \frac{(1+(2Ay+B)^2)^{\frac{3}{2}}}{|2A|}$. As for the calculation of the deviation of the car from the midpoint of the lane, I choose y value in the bottom of the image and compute corresponding x values of the left line and the right line. So the offset, the deviation of the midpoint of the lane from the center of the image, could be calculated by the following equations:

center = (bottom_left_x + bottom_right_x) / 2

offset = (img.shape[1] / 2 - center) * (3.7 / 700)

An example image of my result plotted back down onto the road


radius of curvature = 366.77727095977156 m
offset = -0.17508155915194257 m
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

Pipeline (video)

The video output is included in the workspace and submitted with the project.

Discussion

Compared with the last version of the pipeline, I

- fix color conversion mistake
- adjust the perspective transform to capture a longer stretch of road
- change ym_per_pix to 15.0/720 to estimate radius of curvature correctly

The wobble of detection seems slighter than the former three versions, but the algorithm still has rooms to improve when detecting dashed white lane lines. The pipeline may also fail when dealing with other complicated conditions (i.e. winding road, raining). The solution, I think, is to apply more robust feature extraction methods, such as convolution neural networks, to make the algorithm able to adapt to different situation.

Reference

1. https://github.com/yashwa7 sdcnd Repository
2. https://github.com/vxy10 P4_AdvancedLaneFinding Repository