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](https://review.udacity.com/#!/rubrics/571/view) Points
---

#Writeup

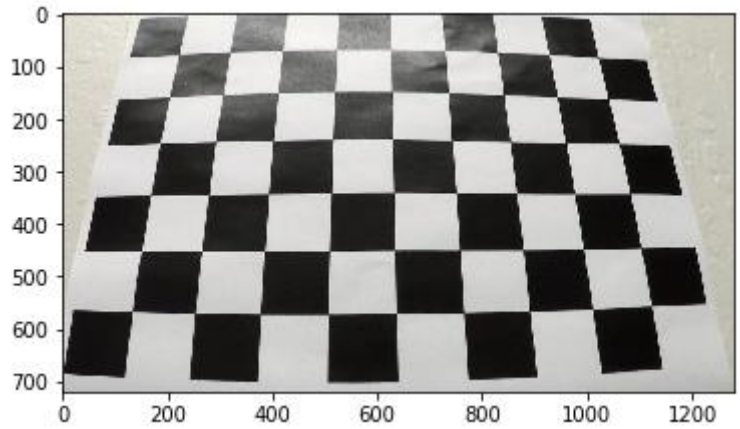
Camera Calibration

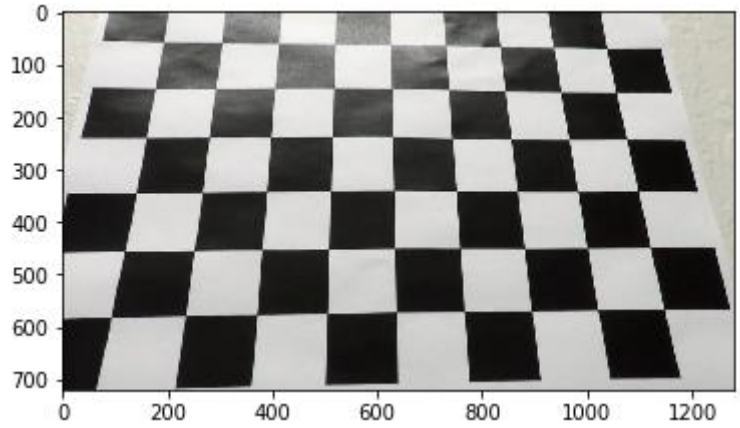
1. Briefly state how you computed the camera matrix and distortion coefficients. Provide an example of a distortion corrected calibration image.

In order to calibrate a camera, I started off with downloading all the chessboard images taken by different angles. By using cv2.findChessboardCorners(gray, (9,6),None), I extracted the corners which are of the size of 9 by 6. Here corners are defined as an intersecting point of 2 different squares locating diagonally to each other.

Once detecting the corners, assigning the addresses to ‘imgpoints’ which is distorted, and to ‘objpoints’ which are not distorted. Given two different arrays, I used cv.2calibrateCamera() function to grab a matrix of reducing the camera distortion as well as the distortion coefficients inherent within the camera. Using these values allowed me to undistort any image that is taken by the same camera. Here is the result of the original image, and the undistorted image.

Before calibration, and after.





Pipeline (single images)

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

This is one of my images that I got rid of the distortion, by applying the matrix I got above.

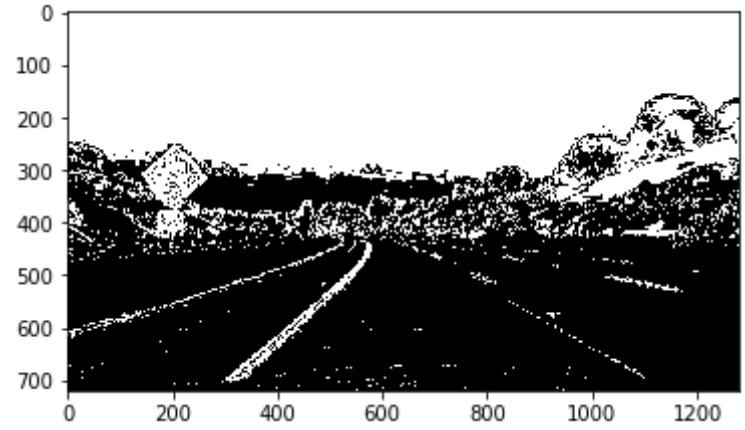
Before calibration, and after.

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.

The functions I generated are in code4: Color and other thresholds.

I used a couple of different filters to generate a binary image; sx threshold, S threshold, mag threshold, and dir threshold. First, I ended up choosing the combination of these filters after many attempts with different filters, kernel sizes and sets of thresholds.
I chose Sx threshold to pick up vertical lines in the image. I didn't want to pick up horizontal lines, so I didn't use Sy. S threshold is robust for the change of brightness. Mag threshold worked to pick up far right line from the front. Dir threshold takes general features out of the image, so this works to pick up any specific features that other 3 thresholds detect. This makes my combined filter robust.

The result is as follows. The top is the original image and the bottom is its binary image.

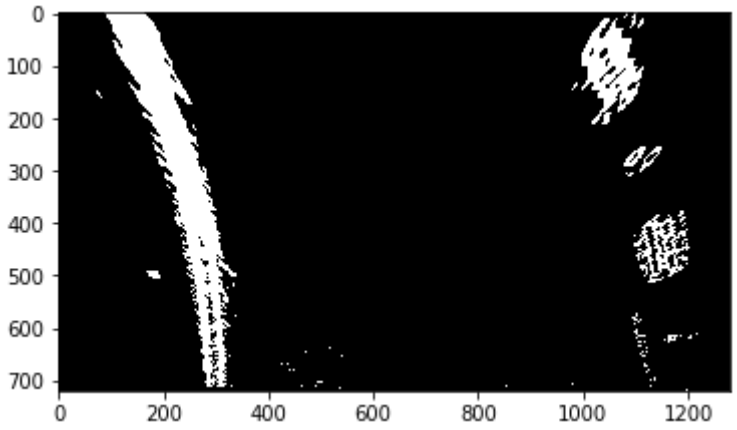


3. Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.

This code is written under the tile; code 3 Preceptive Transformation

I took a set of 4 different points from a front view image, shaping a trapezoid. This is described in 'src' variable. The upper side centers at the half length of xsize, that is, the width of the image. This has a margin for negative and positive about the x axis. This is because when a road has a curvature, the tip of the road in the front view appears oscillating to the direction it's curving to. Therefore, my setting can accept such cases.
The bird view should capture the road from above, so the bottom points of the polygon remain fixed vertically and move apart horizontally to each other depending on the nearest width between the lines. Only the top left and right points of the polygon change in the transformation. I'm going to display the details of parameters below.

Here is an example from one of my images I applied transformation.




```
# define the xsize and ysize
xsize = undst.shape[1]
ysize = undst.shape[0]

# crop an interested region
# a grid in a front view
offset_x_top = 80
offset_y = 0.64
offset_x_bottom = 50

src = np.float32([[0.5*xsize-offset_x_top,ysize*offset_y],
                  [0.5*xsize+offset_x_top,ysize*offset_y],
                  [xsize-offset_x_bottom,1.0*ysize],
                  [offset_x_bottom,1.0*ysize]])


# a grid in a bird view
bird_undst = np.float32([[0.05*xsize,0.05*ysize],
                  [0.95*xsize,0.05*ysize],
                  [0.95*xsize,1.0*ysize],
                  [0.05*xsize,1.0*ysize]])
```

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

The code is written under the tile; code 5 Find lane pixels with windows.

I created a histogram with a searching window which is of the size; the height is the 1/9th of the height of the picture, and the width is a twice of the *margin variable* (200) . The histogram has the same width as the width of the picture, and the height is half the size of the height of the picture.

Looking at the highest values in the left side of the histogram and right side respectively, I retuned the x addresses of these values. I set them as the starting points of the first 2 windows.

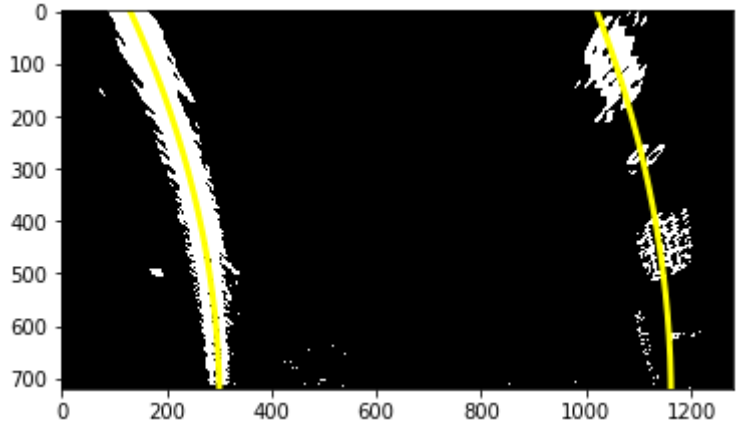
In the windows, I took coordinates of nonzero pixels from the warped binary image and assign them to empty lists; *good_left_inds and good_right_inds*. If the amount of the pixels in the window exceeds a certain amount (*minpix variable*), taking the average values of x addresses, both for left and right line-like pixels, and set them as next starting points. Otherwise, starting points remain same.

Once searched for all the whole image, take the x coordinates from the *good_left_inds and good_right_inds* and get the x and y coordinates for both right and left sides.

The code for fitting a polynomial is written under the tile; code 6 Fit polynomial.

Given the coordinates above, I applied the values to polyfit function to get the coefficients out of them. Using these numbers with the sliced y which ranges from 0 to the size of the image height, this yields the x and y addresses of the left and right line.

Here is the result of a 2nd order polynomial, plotted with yellow.



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.

The code is written under the tile; code 10 Calculate Curvature

Firstly, I converted the whole picture in meter unit from pixel unit. Taking the maximum y value from ploty, I calculated the curvature at the point, as it is the touching point of the polynomial and the imaginary circle. The formula is as follows.

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

I applied the same function to the left and right lines and took the mean value of two yields the average radius.

The code for calculating the vehicle position is written under the tile; code 11 Calculate the vehicle position

Taking the mean value of x coordinates from the left and right lines, I can identify the midpoint of the lanes. Differentiating the midpoint of the image with this value yields the distance of the vehicle from the venter of the lanes, provided that the camera centers at the midpoint of the car front.

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

Out of the coordinates of the left and right lanes on each frame, I plot the covered area with green by using the function cv2.fillPoly.

Original image

Its plotted image

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 I posted my result. The link in the jupyter notebook (./output_project_video.mp4)

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?

When facing with edge cases, such as with extreme brightness or the color of the lane lines look no different to the color of the road, the color value doesn't fit within any thresholds. This can cause the algorithm to fail in generating a binary image.

Also, the cropping size – in my project it was a trapezoid with the fixed size – should change depending on how curvy the roads are. When the road overshoots the range of the defined area, this algorithm can't detect nor paint the drivable space.

If this algorithm can refer to a different picture at the same place in a different situation, this can solve the first problem. Also, having a model of the road shape based on an external map can help the algorithm to improve the precision.