Advance Lane Finding Project Report

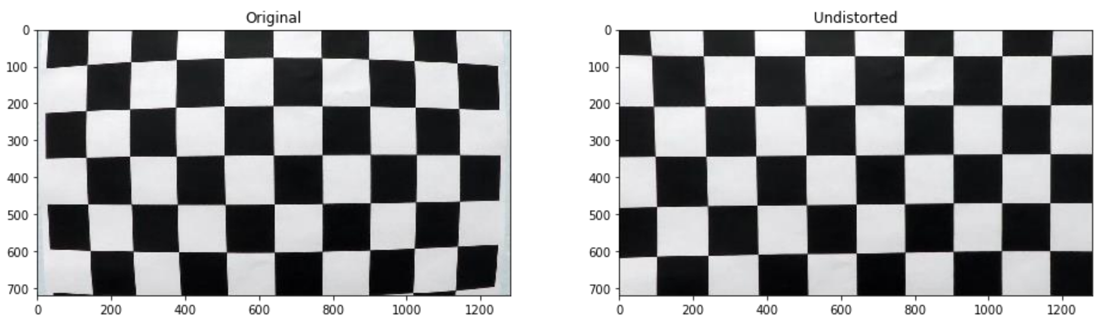
The project is done in a Jupiter notebook. This documentary is just a summary to show that all requirements in the rubric are met. All images can also be find in the notebook main.ipynb or in the folder “output_images.” Final video output can be find in “output_videos”.
An html file exported by jupyter book is also provided, named “main.html”

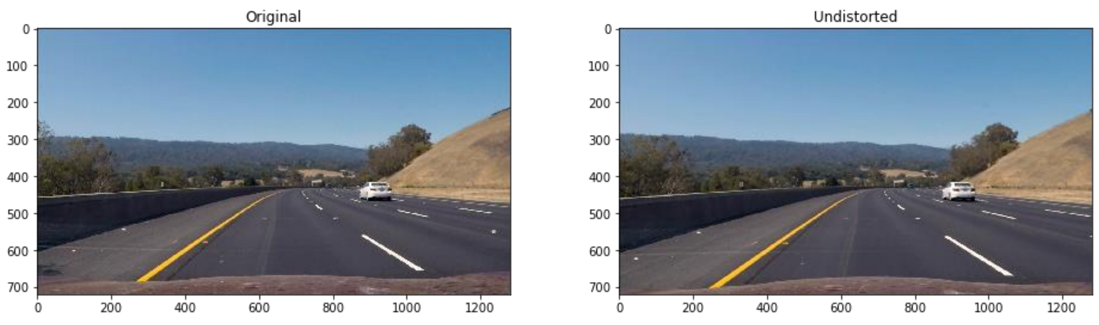
Camera Calibration

1. convert to gray image cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
2. ret, corners = cv2.findChessboardCorners(gray, (nx,ny), None)
3. cv2.calibrateCamera(**args) returns (ret, mtx, dist, rvecs, tvecs)
4. undst = cv2.undistort(img, mtx, dist, None, mtx)○

Here are two examples of the camera calibration result.

1. Chess board
2. A real view on a freeway





Pipeline (images)

Provide an example of a distortion-corrected image.

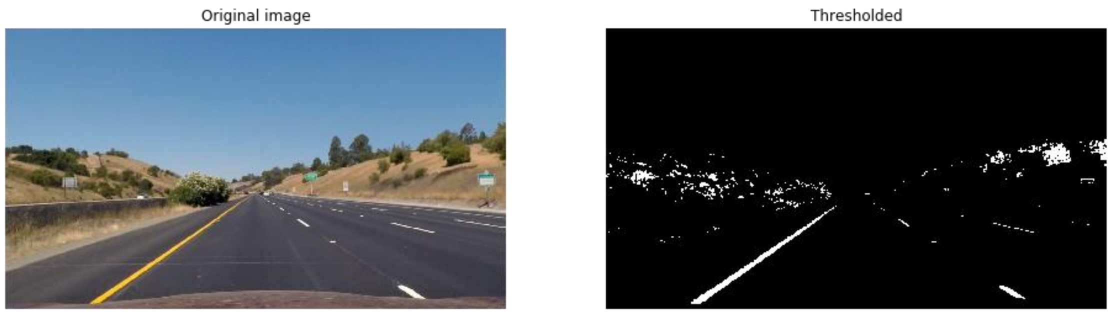
See picture above.

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 defined 4 different functions to pick threshold. However, I didn't use them all in the project. I converted each image into HLS color space to find the threshold, and it worked fine for the project video. A combination of those methods may give a better performance on the challenging videos.

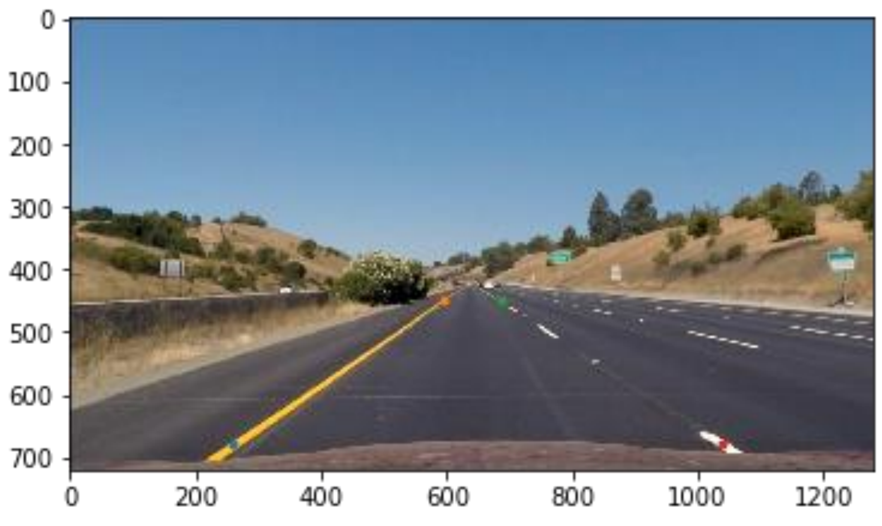
The code is in Cell 7.

Example:

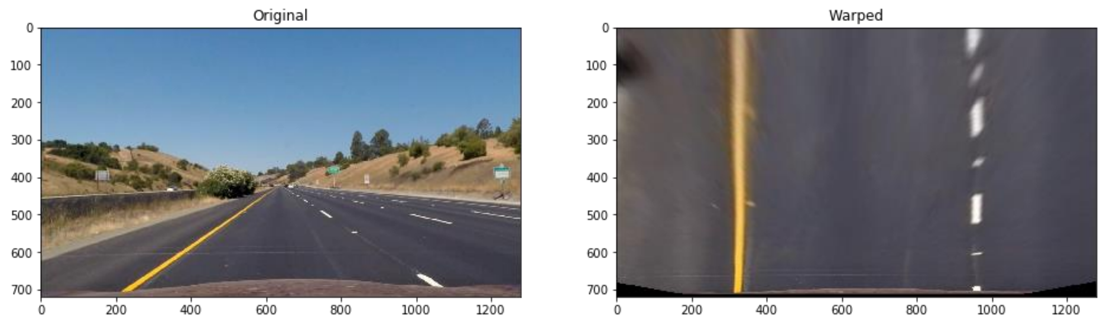


Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.

I chose the "straight_lines1.jpg" to be my template picture for the perspective transform. I manually picked 4 points on the picture and use cv2.warpPerspective(**args) to get a warped picture. The 4 points I picked were [260, 680], [595, 450], [685, 450], [1040, 680], and they mapped on a rectangle [320, 700], [320, 0], [960, 0], [960, 700] with a canvas size (1280,720). The 2 points are colored in the picture below



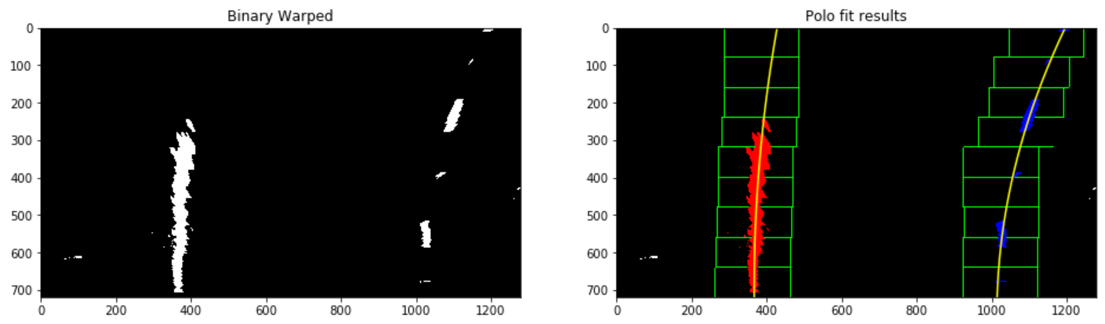
Code for perspective transforming is in Cell 6.
Here is a result example.



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

I used the help code in the course lecture. Adding up the pixels' value in each column, used histogram and find the peaks to find which vertical part has the highest, they got be the pipelines. After finding the estimate position, I started search for lines with 9 blocks.

Here is a visualization, and the code is in Cell 10.



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.

I mapped each pixel in y direction to be 25/720 m. Since in the perspective method I didn't use the whole line, I use 25m instead of 30m to be the length of the road.

I mapped each pixel in x direction to be 3.7/640 m because I used the U.S. regulations requirement of a minimum lane width of 3.7 meters. I mapped the pipelines at 320 and 960 on the x-axis, so the pixels are 640.

I calculated the curvature using the formula below:

$$R_{curve} = \frac{(1+(2Ay+B)^2)^{3/2}}{|2A|}$$

By using the polynomial coefficients found before, we can easily substitute in the equation to find the curvature.

The code is in Cell 11-13, or in Cell 14 that I organized all codes in one function.

Pipeline (images)

> 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 video is in the folder "output_videos."

Code for generating the video is in cell 15.

Discussion

When I tried the generate the videos on Challenging videos, the result didn't perform well. This means my code isn't robust. I am thinking about improving by these methods and will play with them.

1. Use a combination of different threshold. In this project, I only used the HLS color space to detect the line pixels.
2. Fit a higher order polynomial. In the project I used only 2. A 3 degrees polynomial may give a better performance, but meanwhile, it slows down the code.