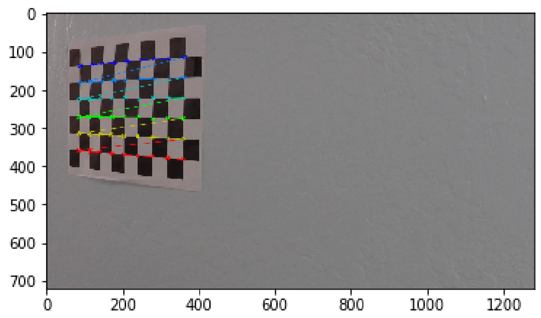
Advanced Lane Finding ProjectThe goals / steps of this project are the following:

- 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.

I will describe rubric points seperately and describe how I process each step.

Draw cornors on Chessboard image

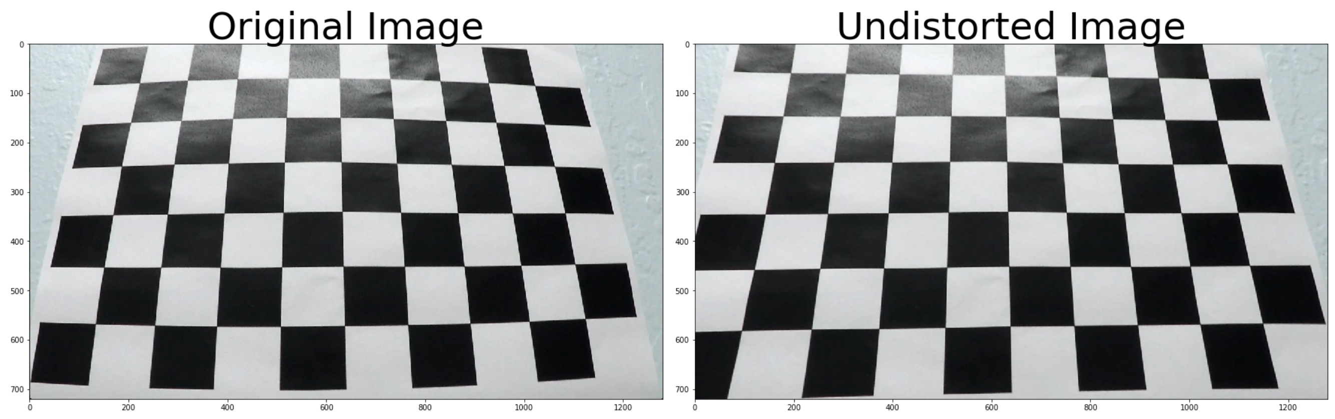
This part is the first part of myProject.ipynb. I started to prepare object points, which will be the (x,y,z) coordinates of the chessboard corners, like (0,0,0), (1,0,0), (2,0,0)… "objp" is a replicated array of coordinate. Through all chessboard images, I read each image in and transfer to gray scale, use "findChessboardCorners" to find the cornors. If corners are found, append each corner points to "imgpoints", "objp" is appended to object points. Then use "drawChessBoardCorners" to draw and display corners on the original image.



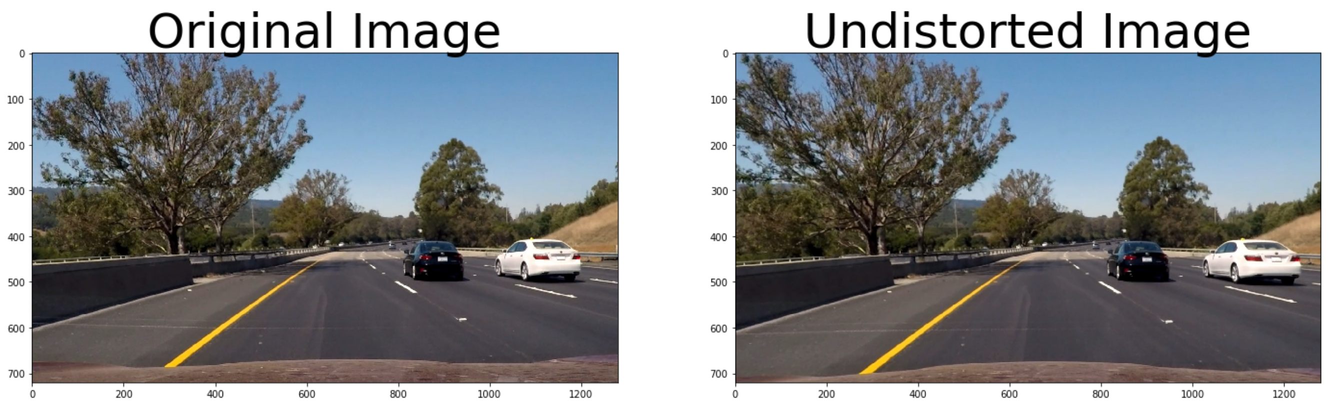
Pipeline (test images)

1. Provide an example of a distortion-corrected image

After read a raw chessboard image, I chose "cv2.calibrateCamera" to perform camera calibration by given image points and object points. Then "cv2.undistort" was used to perform image undistortion. The result is following:



One of the undistorted result of real traffic images is shown below:

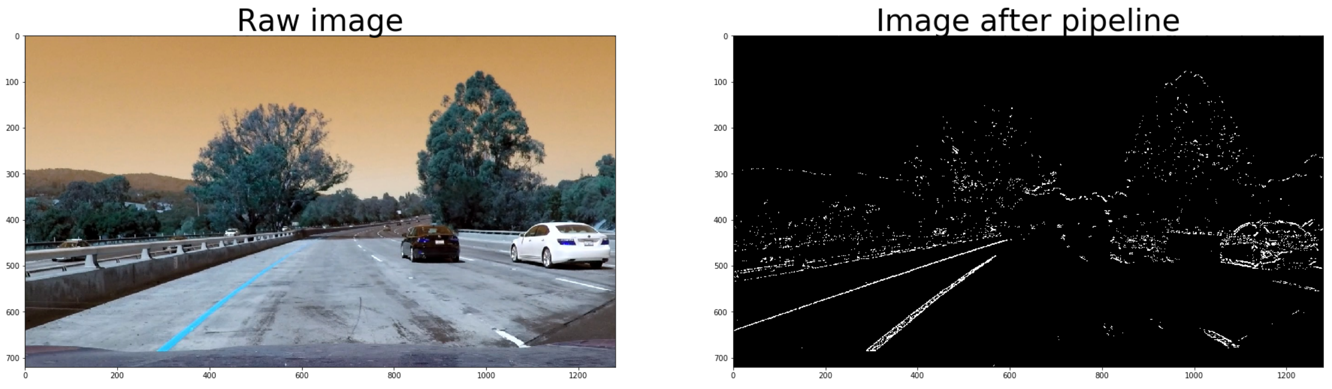


2. Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.

I use combined edge detections method (in [4] and [5] in my codes)to detect the edges from 7 test images, they are listed below:

- HLS color thresholding
- Sober operator x direction
- Sober operator y direction
- Magnitude of Gradient
- Direction of Gradient

The result is shown as:



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

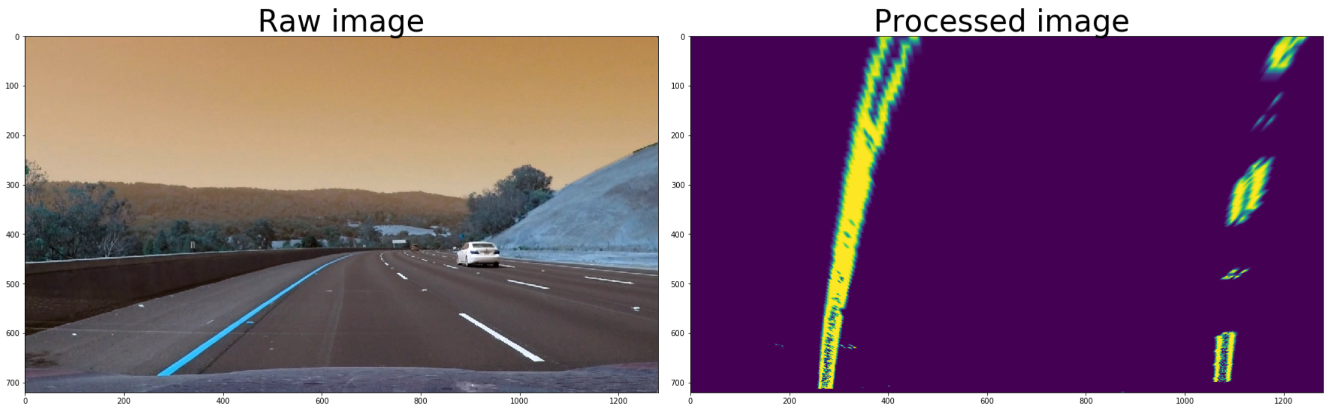
In subroutine corners_unwarp() (in [6] from myProject.ipynb) I apply a perspective transform method to rectify the binary image. This function takes img as input image, as well as the size of image, nx and ny, camera matrix, and the distortion coefficients. I manually choose some destination point to be a good fit for displaying the warped result. The source points are:

[[150+430,460], [1150-440,460], [1150,720], [150,720]]

and the destination points are:

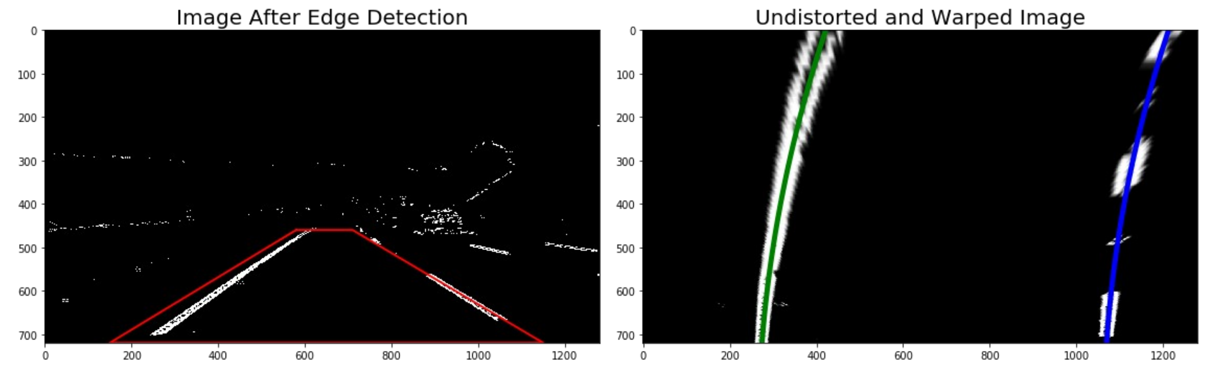
[[200, 0], [img_size[0]-200, 0], [img_size[0]-200, img_size[1]], [200, img_size[1]]]

The result (in [8] from myProject.ipynb) is listed as:

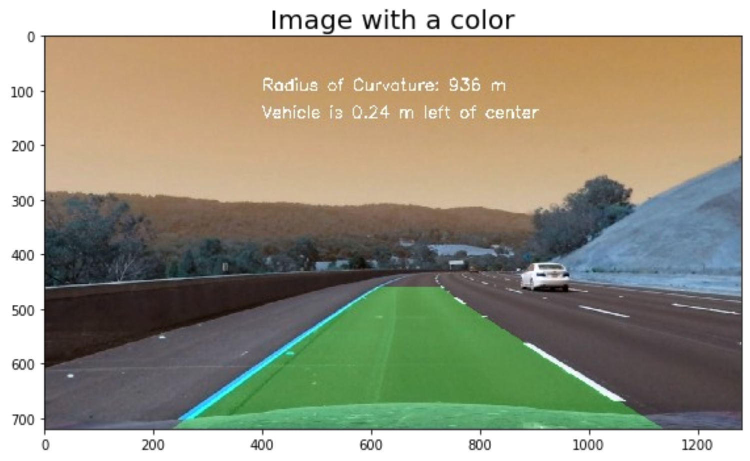


4. Describe how (and identify where in your code) you calculated the radius of curvature of the lane and the position of the vehicle with respect to center.

My function “fit_lanes” will do the work to fit a polynomial line. First, I need to find the peak of the left and right halves of the histogram. Those points are considered as the starting point for the left and right lines. Then setup the window size and the positions. The following step is to go through the windows one by one on images. Use “np.polyfit” to fit left and right lane with 2nd order polynomial fit. The function “find_position” defines the position of the car from the center. It defines conversions in x and y from pixels space to meters.



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



Pipeline (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!)

YouTUbe Link: https://youtu.be/Rmf7Y8GSTrE

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?

The final result in the video shows that it predicts the lanes correctly. It took me lots of time to coding and organizing this process of the pipeline. But gladly I can finish it on time. I use fortran, C and java a lot during work time but not python. Therefore the hardest thing in this project is right code with right grammar. I am highly appropriated the class note which provides much useful coding information so I can keep going forward.

The different combinations to create threshold image show different results. The one I choose in this project may not bring the best result but it is the best I can do. The pipeline may fail if I choose unproper binary image or the area of interest is not drawn correctly. The model can be improved if I can improve edge detection function by other methods to reduce noise effect. The model may fail if it faces some difficult circumstance, such as severe weather condition. The road lanes may not be clearly visible. How to detect lanes from snow day may be next issue to solve.