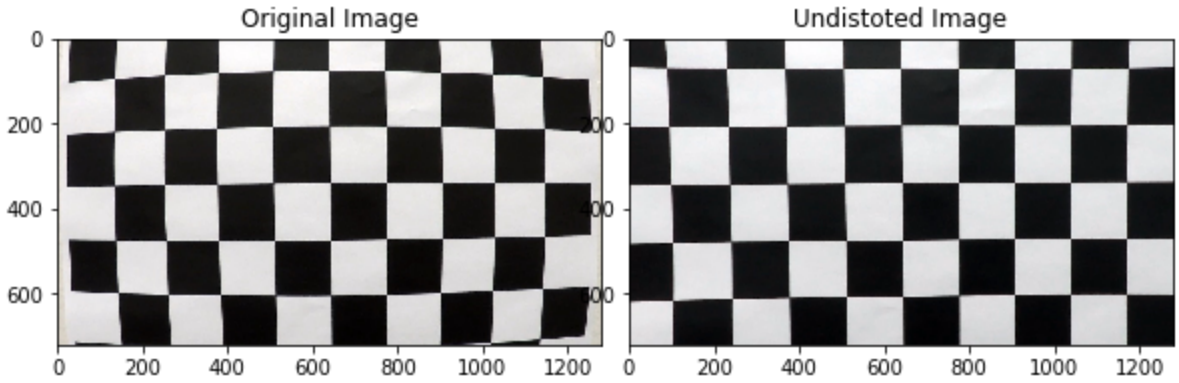
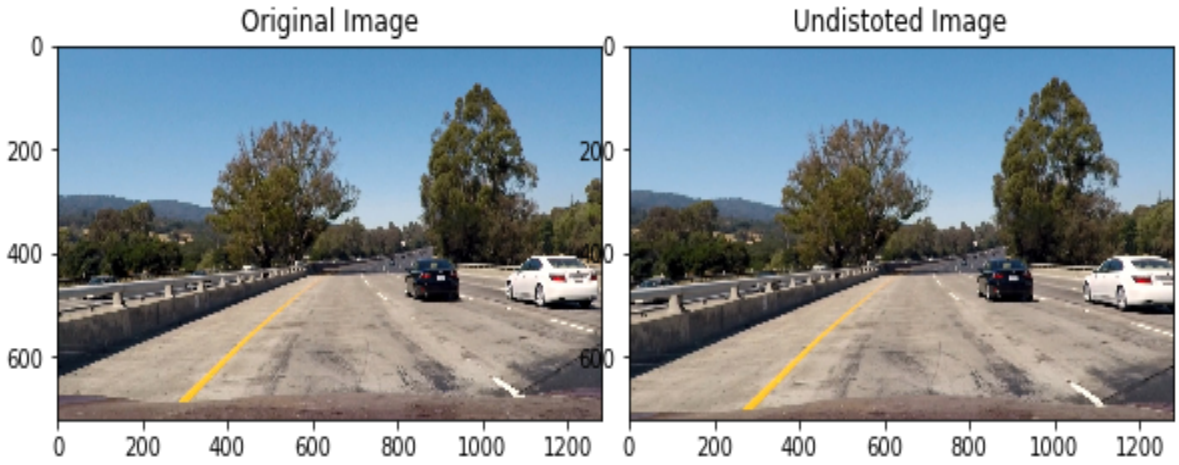
Camera Calibration

In order to undistort the image we firstly need to identify the corners of the chessboard images by using the function cv2.findChessboardCorners. Then we need to create two array(image_point and object_point) to store the corners and a float32 type image which we created to store all object points of the image. Secondly, we use the cv2.calibrateCamera function to get the matrix and distortion coefficients. Finally we can undistort image by using cv2.undistort() with the matrix and distortion coefficients. For example:

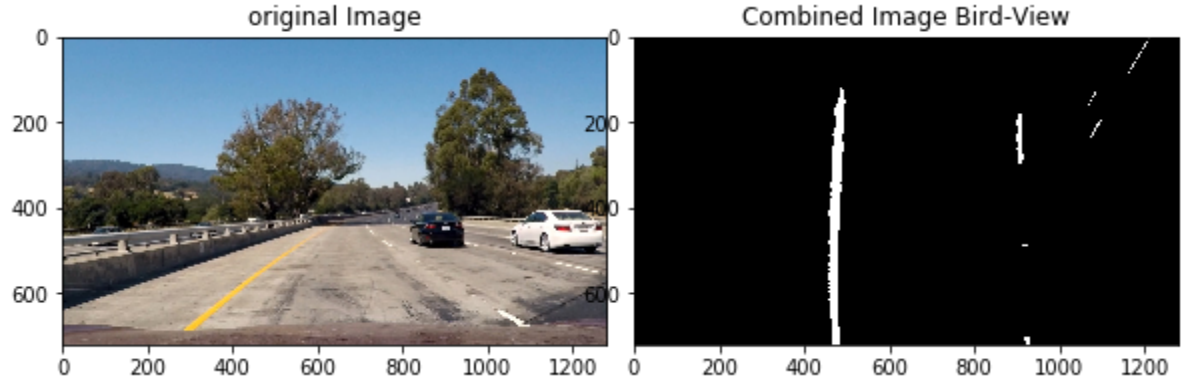


Pipeline(test images)

1. We can undistort an image using the matrix and distortion coefficients of the cv2.calibrateCamera(). Example image where undistortion is applied:



2.From In[8] to In[32] of the code, the image is converted into a binary image. The thresholding() is used to identity the white pixels of the image. The color_thresholding() is used to identity the other light colors in the image. The light colors can be yellow, blue etc… The both the result from the functions will be converted into a single image and then the perspective transform is applied to transform them to a bird-eye view image. For example:



This is the binary form of the original image.

3. We can transform an image to bird-eye view by using two function:

- cv2.getPerspectiveTransform()
- cv2.warpPerspective()

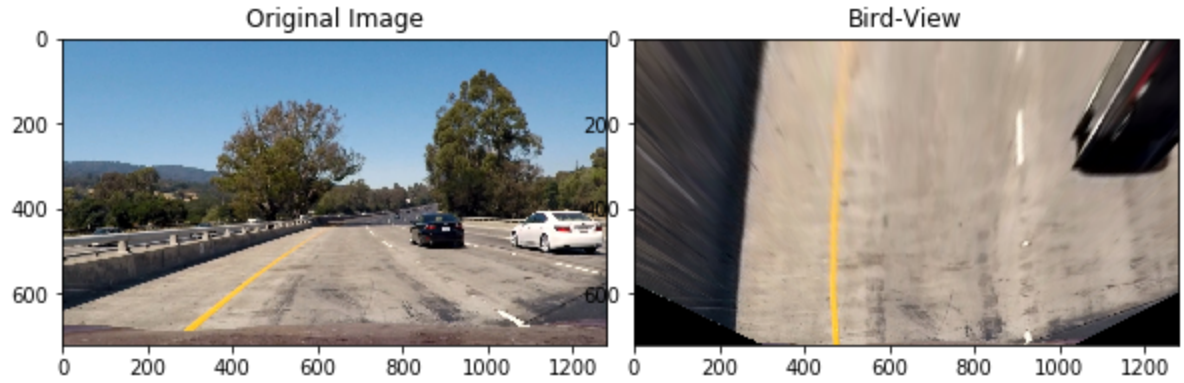
The getPerspectiveTransform() took two parameter.

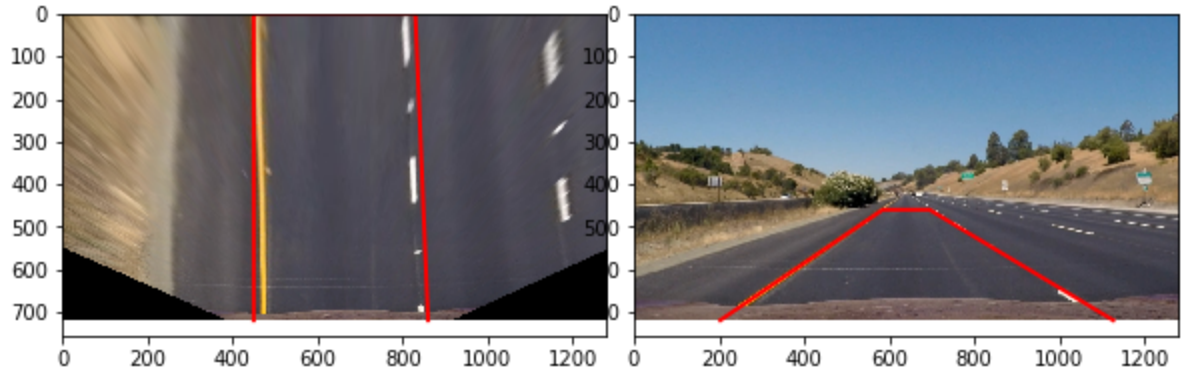
- Src- source_points
- Dst- destination points

One is the 4 coordinator point of part of image to be transformed. Second is the 4 coordinator point of the image to be plotted according to the coordinate values. This functions returns a matrix value of the transformed image.

The warpPerspective takes fours arguments. First one is the original image. Second is the matrix value returned from the getPerspectiveTransform(). Third one is the shape of the image. Finally a flag. This transform image to bird-eye-view image.

The code lies on In[31] of the file.

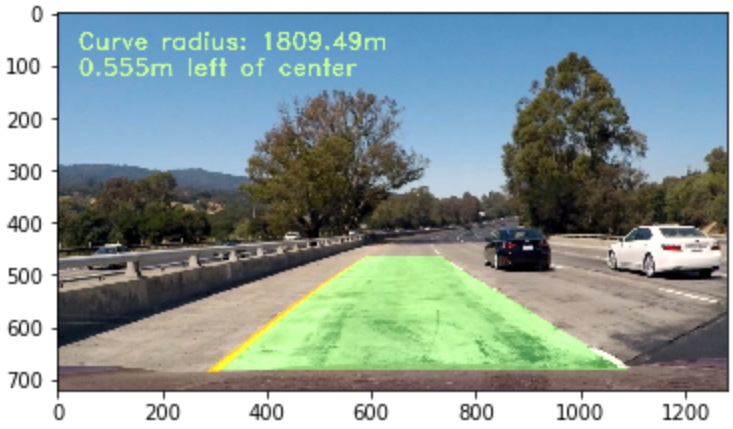




4. According to the lessons and the project prepared, in order to identify the lane line we must convert the image into histogram. Then we take only the half portion of the image. Calculate the midpoint, left and right portion of the image. Using nonzero() seperate the x and y pixels along the the image into an array nonzeroy and nonzerox respectively. Then identify the good pixels in the image and append to left_lane_inds and right_lane_inds array. Extract them with the nonzerox and nonzeroy arrays which will result pixels position of left and right lane line along x and y directions. This results can be fitted with a second degree polynomial which will find a best lane line along the pixel points.
Code lies on In[36] of the file.

5.By fitting a second degree polynomial, we will get the left and right lane line in x and y direction of the image. In order to find the curvature of the line we use the equation (Ax**2+Bx+C) . This will give the left and right curvature of the lane line. Center lane position of the vehicle is the half of left fit and right fit of the image.
Code lies on In[40] of the file.

6.



Discussion

There could be more explanation on finding the lane line. I felt a little bit confused in the part of lane finding. It takes some time to understand what was meant about the sliding window and skipping the sliding window. I am very amazed by the use of histogram because each simple things make sense in some way or the another. The things which i learned during under-graduation period like the second polynomial fit, line equation(first project) was used throughout the project and i felt surprised and happy by seeing a practical use of them.