Writeup Template

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

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 Points

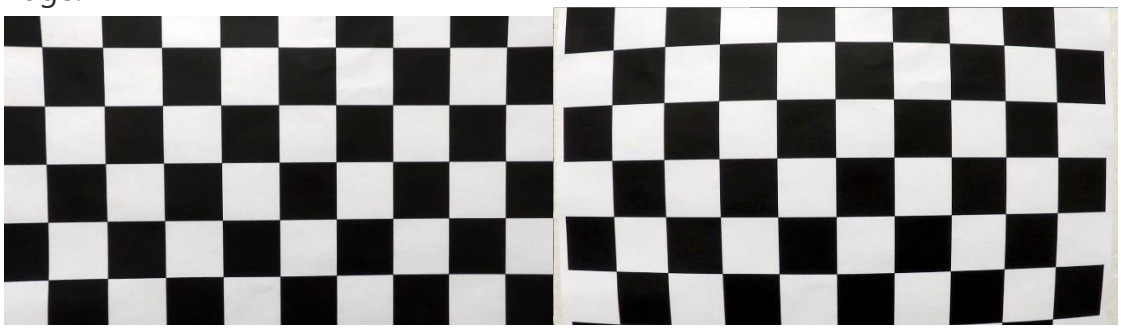
Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Camera Calibration

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

- The code for this step is contained in the first code cell of the IPython notebook.I start by preparing "object points", which will be the (x, y, z) coordinates of the chessboard corners in the
world. Here I am assuming the chessboard is fixed on the (x, y) plane at z=0, such that the object points are
the same for each calibration image. Thus, objp is just a replicated array of coordinates, and objpoint[[0. 0. 0.]
 [1. 0. 0.]
 [2. 0. 0.]
 [3. 0. 0.]
 [4. 0. 0.]
 [5. 0. 0.]
 ,,,
 [3. 5. 0.]
 [4. 5. 0.]
 [5. 5. 0.]
 [6. 5. 0.]
 [7. 5. 0.]
 [8. 5. 0.]]

will be appended with a copy of it every time I successfully detect all chessboard corners in a test image.
imgpoints will be appended with the (x, y) pixel position of each of the corners in the image.

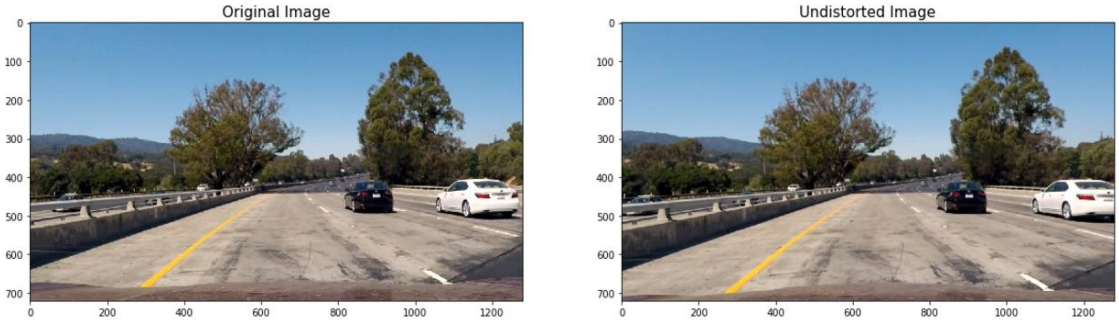


Example of Undistorted and distorted image respectively.

Pipeline (single images)

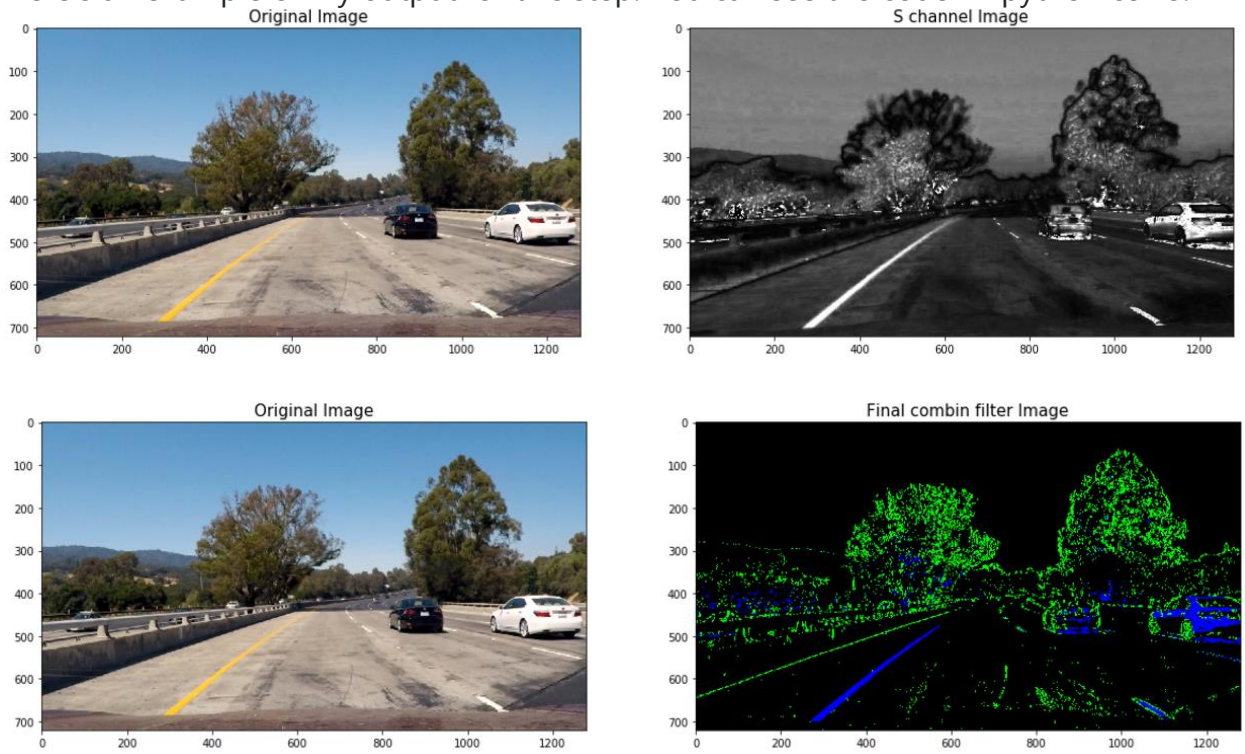
1. Provide an example of a distortion-corrected image.

- I used the cv2.undistort to correct the distortion. And provide the distortion correction matrix in the function. Here is the example of correction:



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 used a combination of color and gradient thresholds to generate a binary image. Here's an example of my output for this step. You can see the code in ipython cell 8.

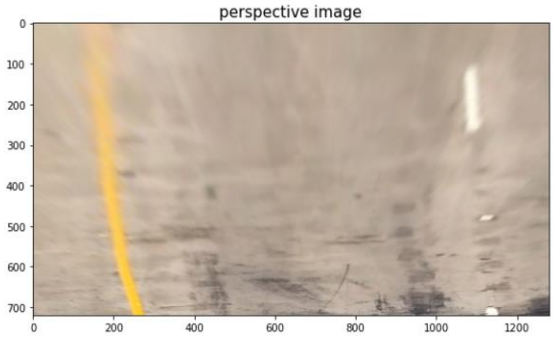


3. Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.

my perspective transform takes raw image (720) and transform matrix M, warp the image to bird view. I choose the fix src and dest point in image

src = np.float32([755,470], [1100,675], [300,680], [600,470]])

dest = np.float32([1160,0], [1160,720], [260,720], [260,0]])




```
def perspecive(img):
    imgshape = (img.shape[1],img.shape[0])
    M = cv2.getPerspectiveTransform(src,dest)
    Minv = cv2.getPerspectiveTransform(dest,src)
    wraped = cv2.warpPerspective(img, M, imgshape, flags=cv2.INTER_LINEAR )
    return M, Minv, wraped
```

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

I sliced the input image into n= 9 bars, each bar, I calculation the histogram and find the two peaks can represent the lane line location. Then save the good location points for left and right lane separely, then feed into a 2nd order polynomial function as showing:

There are two part of work: 1. From last n sliced window, we can collect at least 3 good points for left or right lane, for example `lefty` and `leftx` that can fit into `left_fit = np.polyfit(lefty, leftx, 2)`, the output is the A, B, C values for polinomial function f(y)=Ay2 +By+C. 2. Calculate the left_fitx and right_fitx values according to each row in the picture space.

5. Describe how you calculated the radius of curvature of the lane and the position of the vehicle with respect to center.

Remember, the "bird_view" in this pipeline is not true bird view. Because of the camera is pointing forward and downward at small angle, in the view, the vertical pixel may repersent 100m and further. The horizotal pixel is repersenting the car width closely. There are the factors I used to convert pixel space to true environment.

```
# Define conversions in x and y from pixels space to meters
ym_per_pix = 30/405 # meters per pixel in y dimension
xm_per_pix = 3.7/500 # meteres per pixel in x dimension
```

Pick a y-value where we want radius of curvature. I choose the maximum y-value, corresponding to the bottom of the image y_eval = np.max(yvals). Because it is close to the vehical, the curve can reflect the steering angles.

```
left_fit_cr = np.polyfit(np.array(lefty,dtype=np.float32)*ym_per_pix, \
                         np.array(leftx,dtype=np.float32)*xm_per_pix, 2)
right_fit_cr = np.polyfit(np.array(righty,dtype=np.float32)*ym_per_pix, \
                          np.array(rightx,dtype=np.float32)*xm_per_pix, 2)

def left_curverad(left_fit_cr, y_eval):
    left_curverad = ((1 + (2*left_fit_cr[0]*y_eval + left_fit_cr[1])**2)**1.5) \
                            /np.absolute(2*left_fit_cr[0])
    return left_curverad

def right_curverad(right_fit_cr, y_eval):
    right_curverad = ((1 + (2*right_fit_cr[0]*y_eval + right_fit_cr[1])**2)**1.5) \
                                 /np.absolute(2*right_fit_cr[0])
    return right_curverad
```

Final image output .

[Video Link]

Discussion includes some consideration of problems/issues faced, what could be improved about their algorithm/pipeline, and what hypothetical cases would cause their pipeline to fail.

1 > I notice that my pipeline is not working correctly in shadow condition.

Any good suggestion ?

2 >The method have more complex, can Car speed couse any problem ?

3 > What happen if lane lines not visible due to surrounding vehicles ?