The pipeline of find the lane line

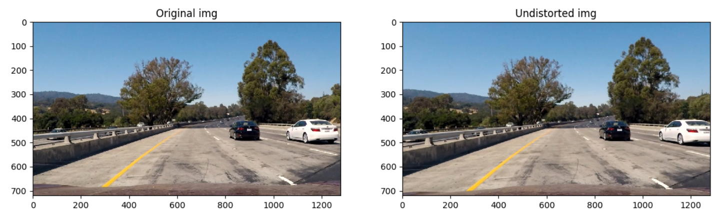
This project is focus on using computer vision to find the lane lines. I used the suggested pipelines to finish this project: The goals / steps of this project are the following:

- 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.

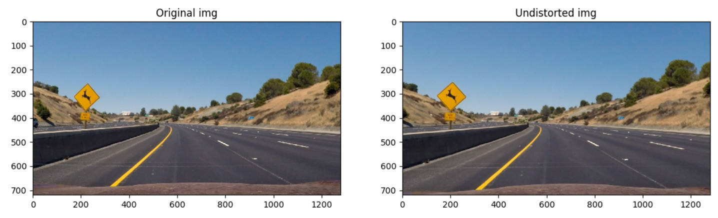
0.1 Provide an example of a distortion-corrected image.

To demonstrate this step, I will describe how I apply the distortion correction to one of the test images like this one:

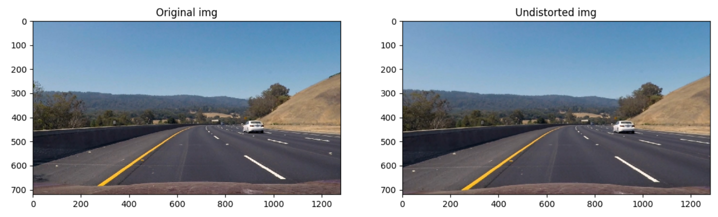
First of call, I compute the camera calibration matrix and distortion coefficients given a set of chessboard images. Then using the opencv function, I apply a distortion correction to original images. The undistorted image can be shown as Fig 1.



(a)



(b)



(c)

Figure 1: Illustration of original images and undistorted images.

0.2 Describe how you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.

In the part, I used a combination of color and gradient thresholds to generate a binary image. In the beginning, I first employ the Gaussian smoothing to relieve the noise problem. I applied a threshold based on the S-channel followed exactly the classroom tutorial.Then convert the combined binary image into a birds-eye image. Fig 2 illustrated a number of examples.

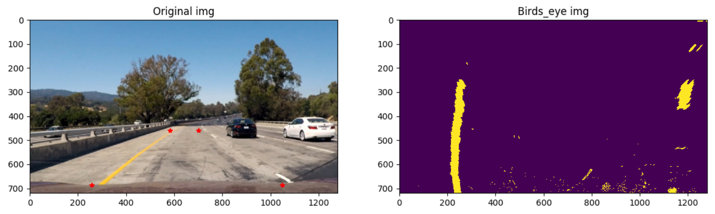
The code for my perspective transform includes a function called birdsEye, which appears step-4 in the file example.ipynb (/examples/example.ipynb). The birdsEye() function takes as inputs an image (img). For the source (src) and destination (dst) points, I used the same choices as the example writeup.

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

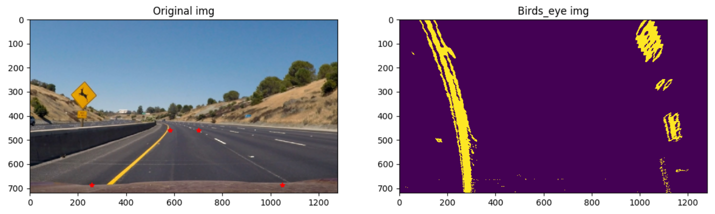
Given the binary warped image, I take a histogram of the bottom half of the image. Then I used numpy function to fit a second order polynomial to each lane.

0.4 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.

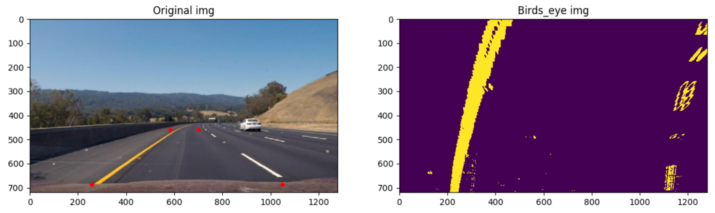
Radius of curvature, corresponding geometrically to the radius of the circle defining the curve, and algebraically, to a simple equation of first and second derivatives of the curve (for a graph).See below



(a)



(b)



(c)

Figure 2: Illustration of original images and warped images.

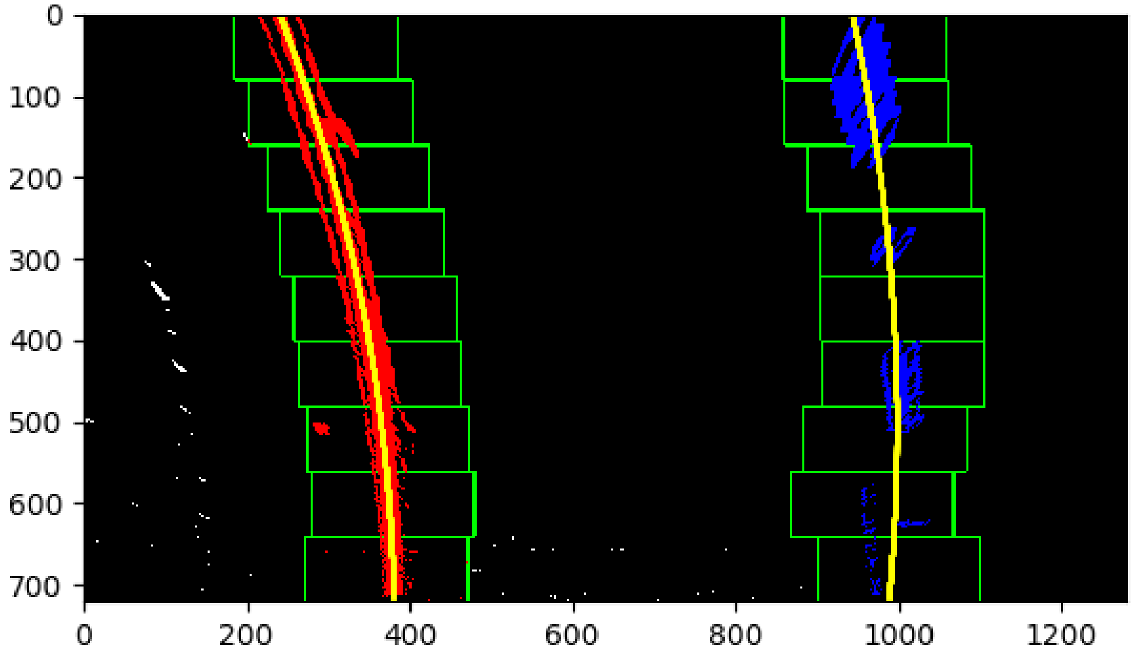


Figure 3

Figure 4: Determine the pixels of lanes