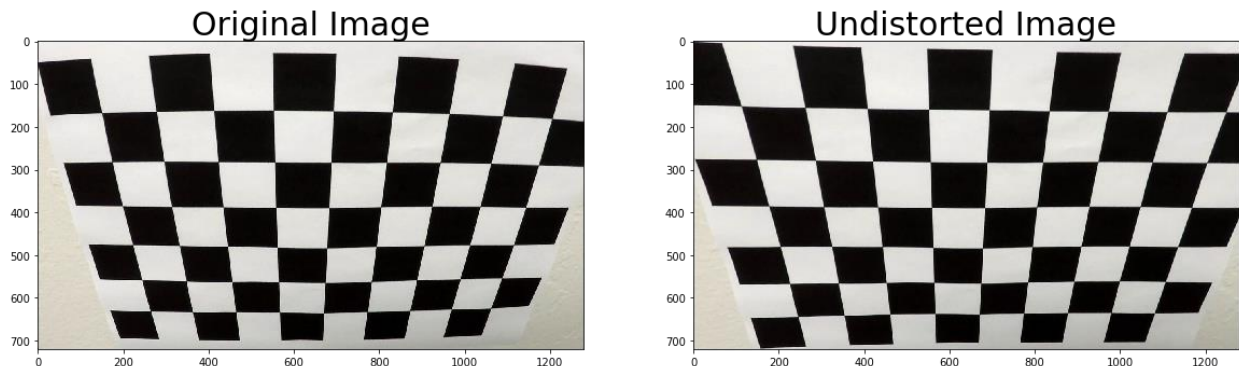


Camera Calibration

I addressed camera calibration in first four code blocks in ipython. I used `cv2.findChessBoardCorners()` to find all the corners in the 20 camera calibration images. All the corner points are saved in an array called `imgpoints`. Then `cv2.calibrationCamera()` is called to calculate the camera matrix and distorted coefficients. At the `cv2.undistort()` is used to get the final undistorted images. Below is an example.



Pipeline (Test Image)

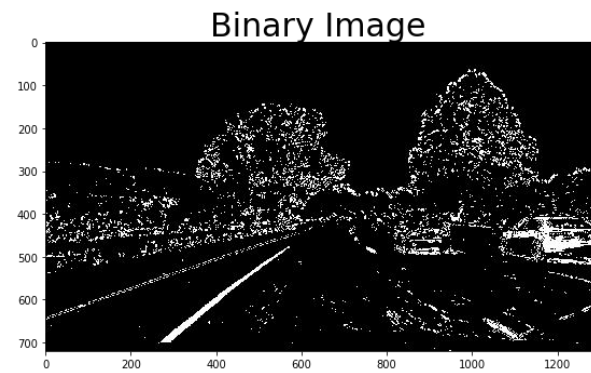
1) Distortion-corrected Image

I undistorted all input test images in the fifth code block. Below is an example.



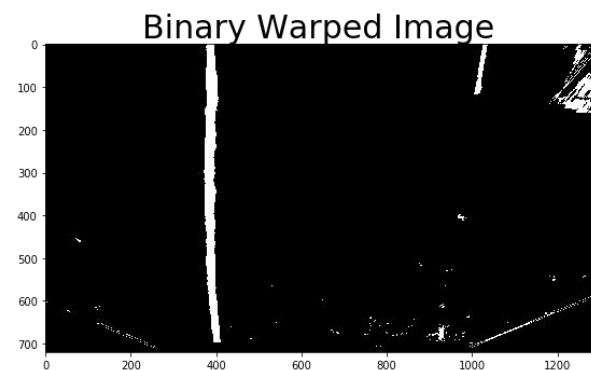
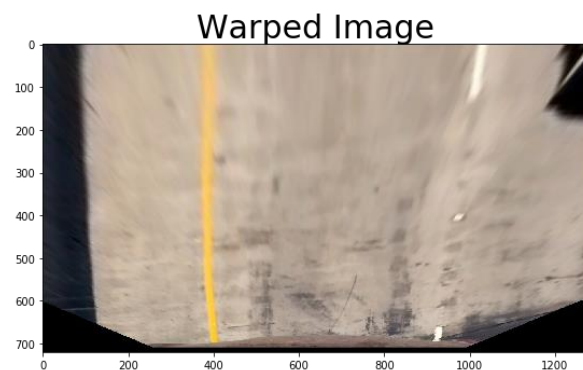
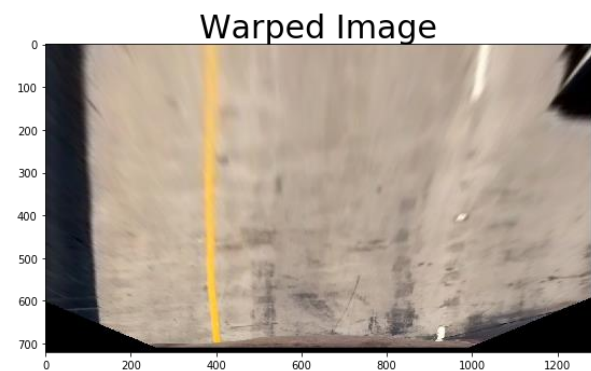
2) Get Binary Image

I used a combination of gradients and color transform to create a binary image containing likely lane pixels. For gradient part, I used Sobel operator to calculate x-gradient, y-gradient, magnitude-gradient and directional gradient and the combined these results to generate the final gradient binary image output. For color transformation, I used HLS color and created a binary image from S channel with a threshold. I combined the two binary images output to get the final binary image. The related code is in the sixth code block. Below is an example of the output.



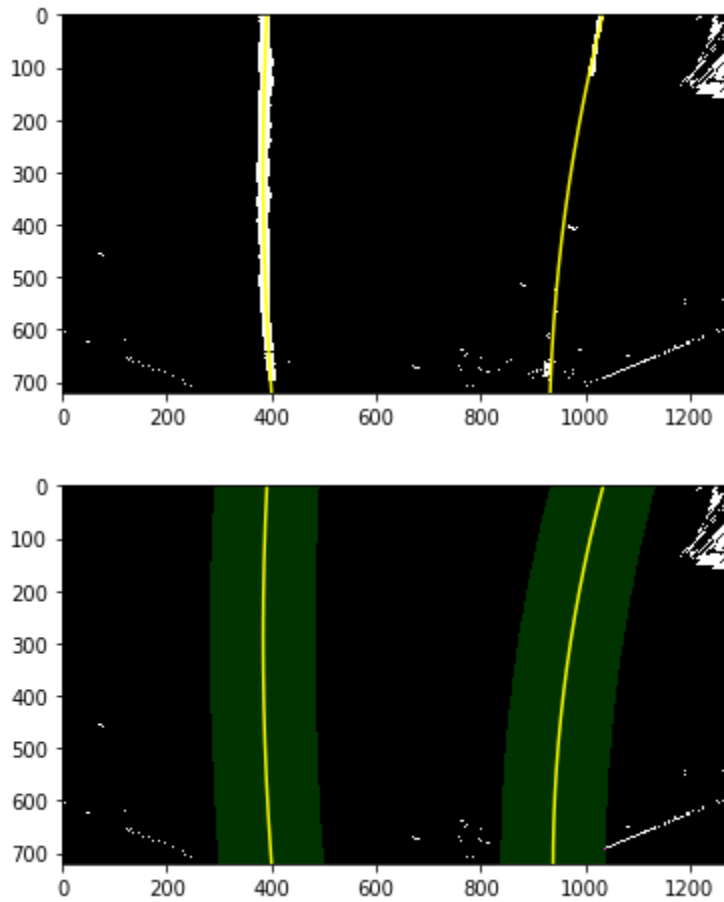
3) Perspective Transform

I performed perspective transform in the seventh to ninth code block. I chose four point in one of the original undistorted input image, and matched these points with another four points which form a rectangular. With these two pairs of four points, I called `cv2.getPerspectiveTransform` to get the transform matrix and inverse transform matrix. With I transform matrix, perspective transform was taken up by calling `cv2.warpPerspective`. I used the approach to get the binary image of the warped image. Below is an example of the result.



4) Fit the Lane line with Polynomial

I used the function `fit_lane_line` to get polynomial fit for the lane line detected. I used the slide window based method to locate all the points along the lane line. With these points, a two order polynomial function is used to fit these points. Below are the results.

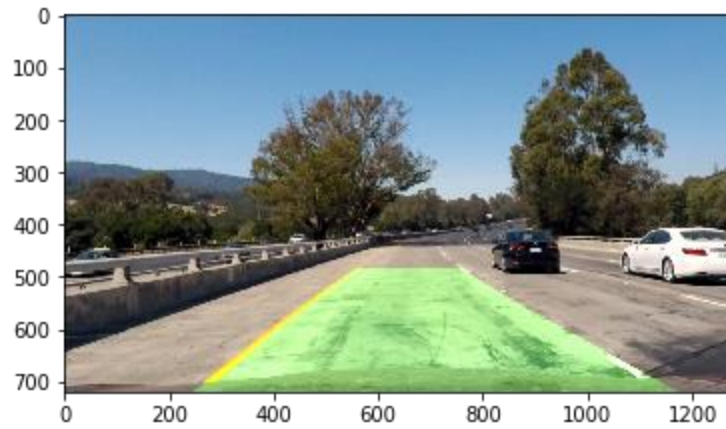


5) Curvature Calculation

I calculated curvature by using function `get_curvature`. I used the points gotten in lane line fit process and used the formula provided in the lecture to calculate real world curvature. For the above image, the curvatures for left and right lane are 1902.06375246 meter and 1093.87042014 meter, respectively.

6) Plot back down unto the road

Below is an example image.



Pipeline (Video)

Link:

<https://drive.google.com/a/cs.unc.edu/file/d/0B4XeyMSkZqcFX0Z2VlJmR2dkNjA/view?usp=sharing>