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

1.Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.

2. Apply a distortion correction to raw images.

3. Use color transforms, gradients, etc., to create a thresholder binary image.

4. Apply a perspective transform to rectify binary image ("birds-eye view").

5. Detect lane pixels and fit to find the lane boundary.

6. Determine the curvature of the lane and vehicle position with respect to center.

7. Warp the detected lane boundaries back onto the original image.

8. Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

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

The Distortion coefficients are obtained from the chess images by mapping 3D points to 2D points and identifying the coefficients, which can be used to calibrate the camera.

Here we 9*6 corners chessboard for calibration.

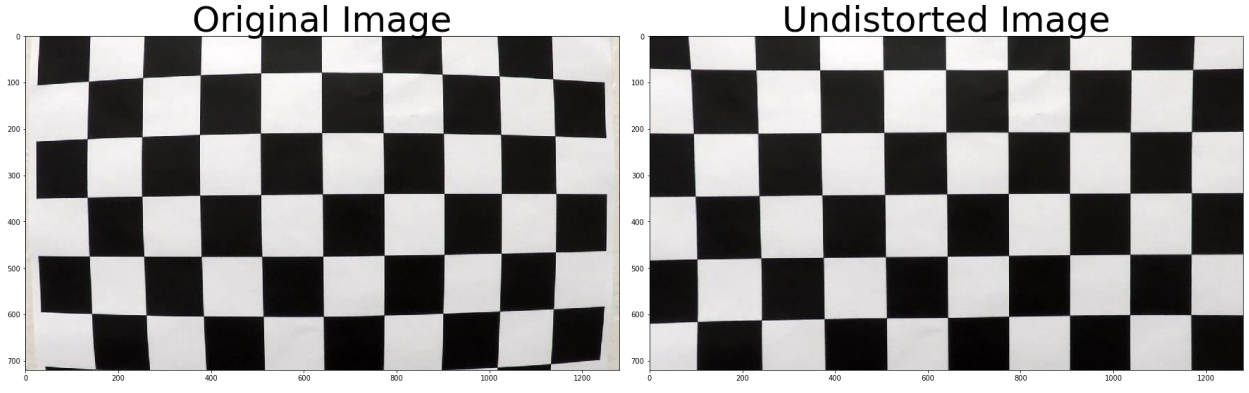
The corners are obtained using cv2.findChessboardCorners function, the image is passed into this function after being converted into gray image.

The corners of the chessboard are mapped to image points.

Using cv2.calibrateCamera function we get the camera matrix and distortion coefficient.

After applying this to an image using cv2.undistort we get the undistorted image.

Example Images:



2. **Provide an example of a distortion-corrected image.**

Below is an example a test image to which distortion correction has been applied.

Original Image

Undistorted Image

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

I used a combined sobelx and S channel binary to get the threshold image.

First I converted the image into HLS and obtained the S channel values.

We use S channel since it gives better lane detection for yellow lines as advised in the classroom tutorials.

For Sobel X I used the S channel instead of gray to identify the strong X gradient.

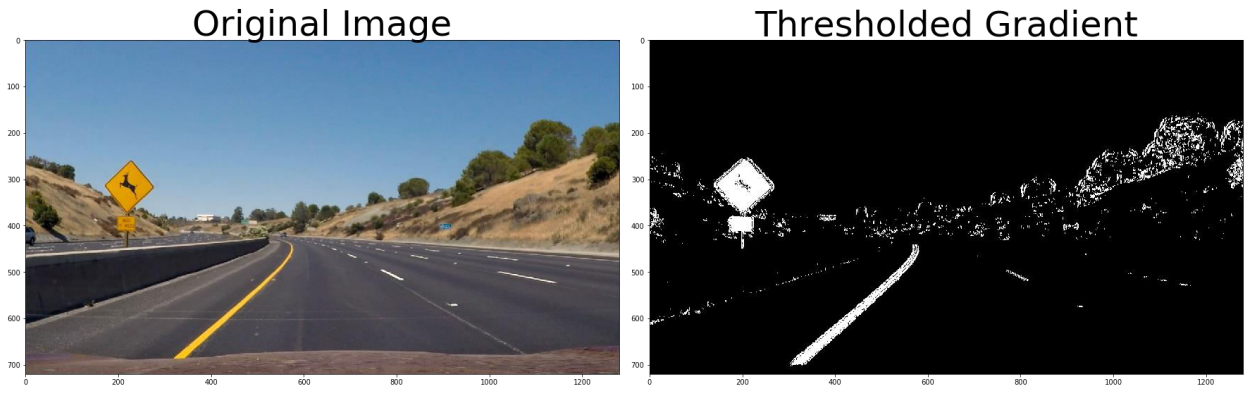
Then Selected the pixels where the S channel has threshold above 170 to identify lanes.

The Choose the final image who value can be 1 if any of the above gradients has 1 as the value.

Thus, I combined the sobelx and S channel with or operation to obtain the **thresholder binary image.**

See code block 7

Example test Image is as below.

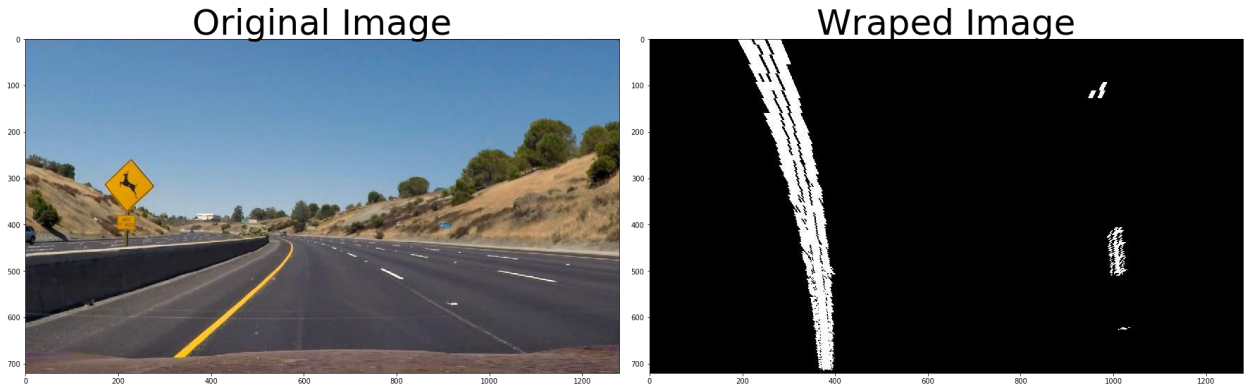


4. **Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.**

I Selected the Source and destination points based on the image size to be more dynamic and used the cv2.getPerspectiveTransform and cv2.warpPerspective to do the perspective transform.

See code block 10.

Example image is as below.



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

Using the histogram, I identified the spike in pixels' values to find the white lines.

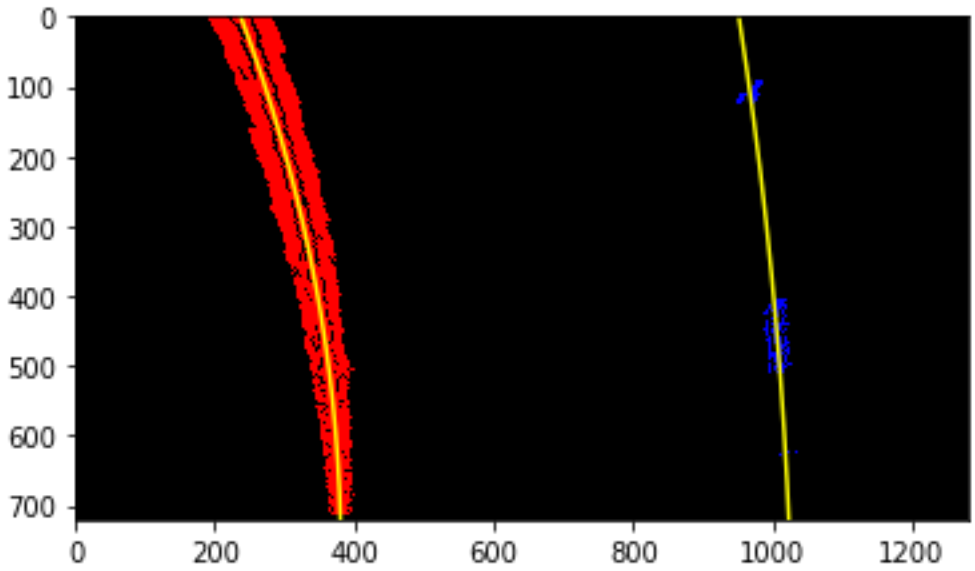
Then obtained non-zero in the plane with function Image.nonzero() for x and y.

I chose the number of windows and its width. I started form the bottom and found the non-zero pixels within this box and appended them.

Then I moved the window based on the mean values of nonzerox.

See code block 12 and 13.

The example image of lane line is as follows.



6. **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.**

See code Block 15.

The blow formulae in image is used for radius of curvature.

$$x_{real} = M_x x_{pix}$$

$$y_{real} = M_y y_{pix}$$

$$x_{pix} = Ay_{pix}^2 + By_{pix} + C$$

$$\frac{dx_{real}}{dy_{real}} = \frac{M_x}{M_y}[2Ay_{pix} + By_{pix}]$$

$$\frac{d^2x_{real}}{dy_{real}^2} = \frac{M_x}{M_y^2}[2A]$$

$$Radius of Curvature = \frac{(1 + \frac{dx_{real}}{dy_{real}}^2)^{1.5}}{|\frac{d^2x_{real}}{dy_{real}^2}|}$$

The formula used in obtained for github ‘https://github.com/sumitbinnani/CarND-Advanced-Lane-Lines/’

The Distance from center is obtained by the formula.

Left=obtained by a polynomial of left_fit[0] which is the bottom one and the same is done for right and mean is used to obtained the position of the car and subtracted for image.size[1]/2 to obtain the distance from center.

Example image with data:

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

The Final stage of test image is show below.

8. **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!)**

The output video is in the project submission folder.

9. **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 video is not performing greatly under shadows. Despite using sobelx, the trees shadows are getting picked. Maybe a better combination of gradients and color may show accurately the lanes.