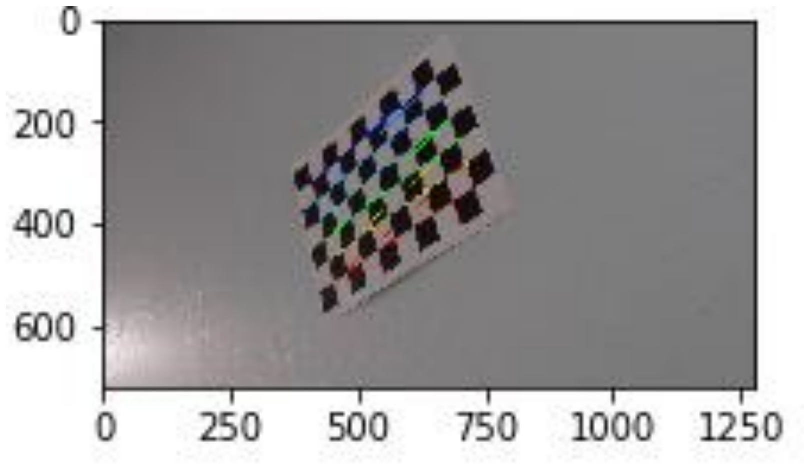
Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

The curvature captured is leaning towards the left until I figured out corrected midpoint and the corresponding fit. At the first take, after following the course and referring some online tutorials, the fit doesn't seem to be working well with default number. There must be a robust way to come up with thresholds and lane metrics. I will have to explore those options in detail to pick more smart numbers

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

Followed the class lessons to comeup with obj_points and the img_points using 9*6 checker board.

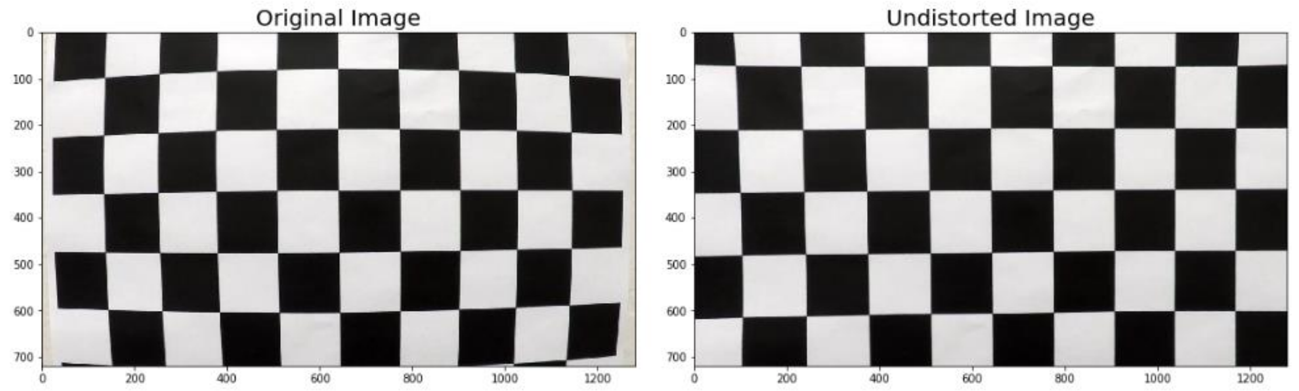
A sample calibration image look like this:



Distortion is perfomed by

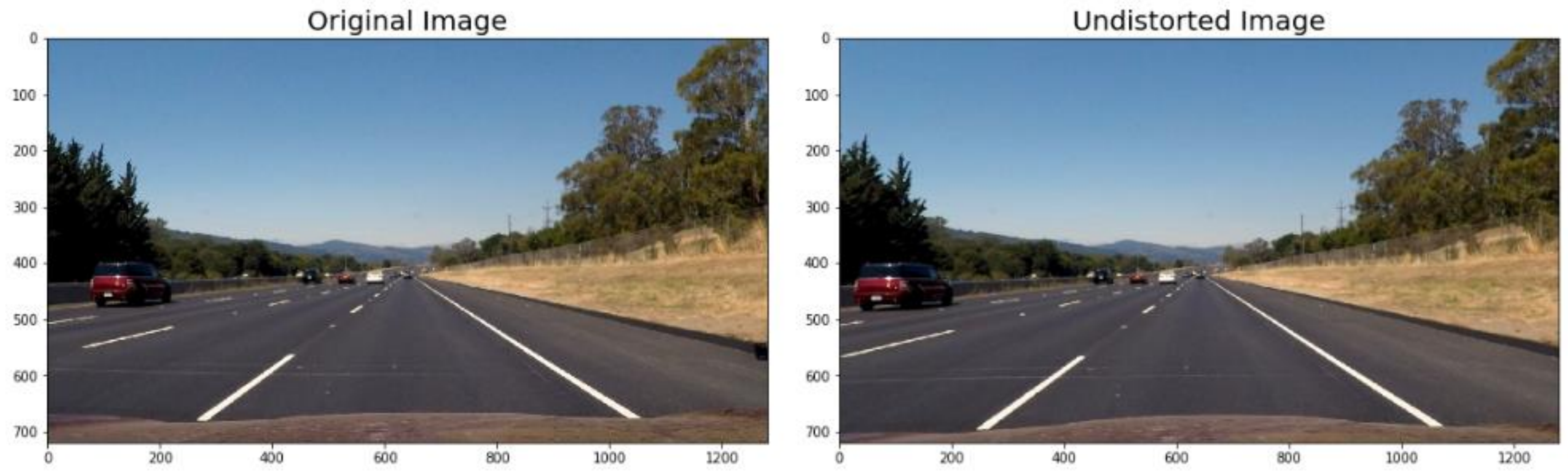
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1], None, None)

The following is a comparison of original image and distorted image


Original Image
Undistorted Image

Distortion as applied to a test image shown below:

Since the original image already flat, not much difference is noticed except for the car dashboard/front bonnet – little distortion happened there.



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.

As part of the perspective transform the source and destination points utilized are:

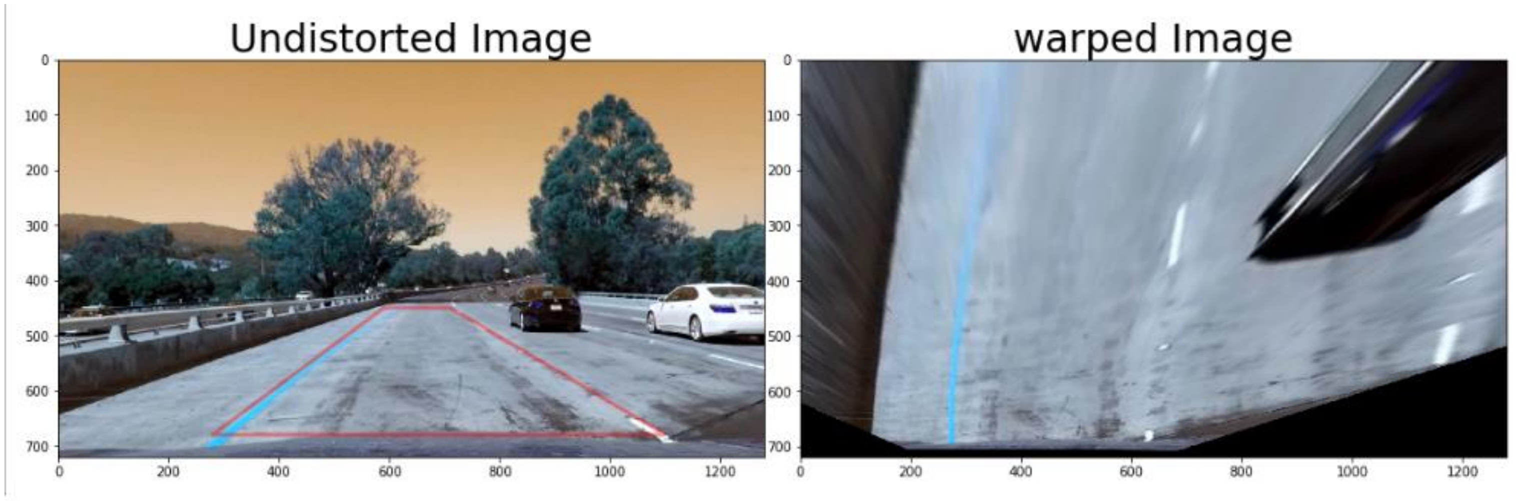
Source points : [800,510],[1150,700],[270,700],[510,510]

Destination points are picked just by glancing at the points after lot of trial and error: [650,470],[640,700],[270,700],[270,510]]

Once the points are fixed, I did the perspective transform using the cv2 function and the warpperspective to get the *unwarped* image.

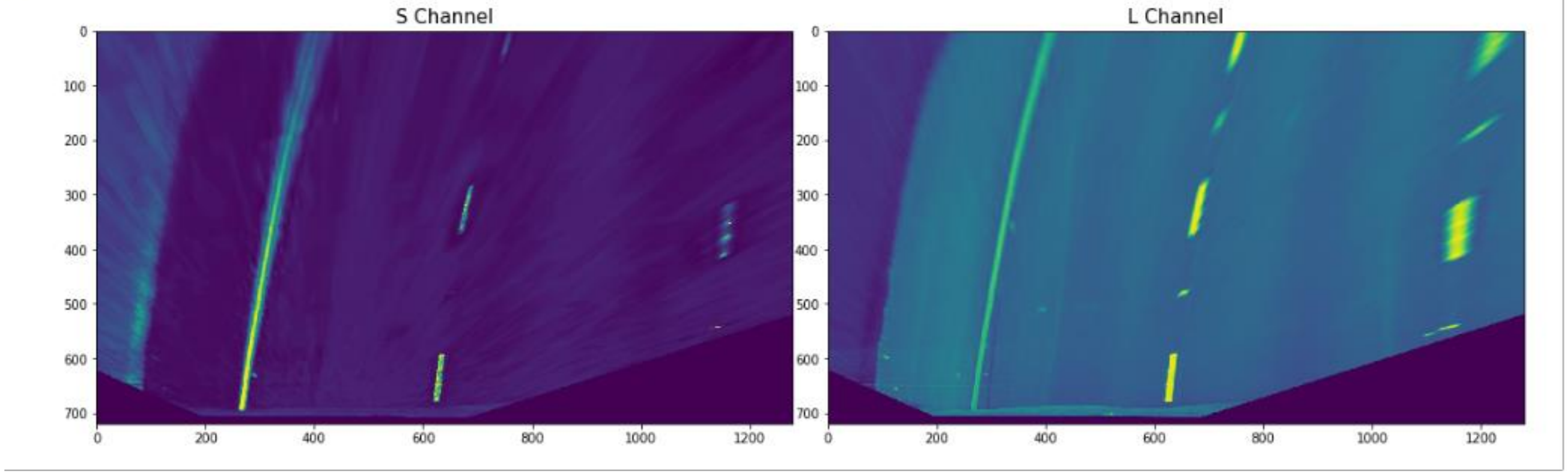
Example of warped binary output is below:

The Sobel thresholds, magnitude threshold and directional threshold, color thresholds are shown in cells 9 and 10 respectively. After transformation, the S channel looks lot better than the other colour channels, with more focus on two lanes.



Binary Output:

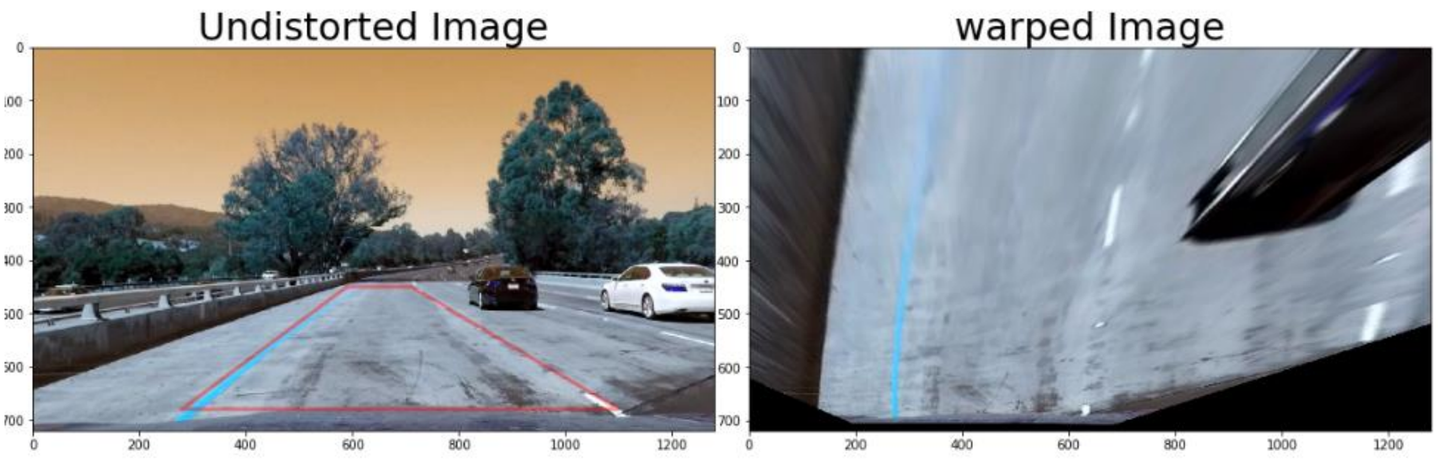
For binary I considered stacking Sobel threshold and HLS (S) threshold since Sobel gave better edges than other thresholds. Also directional threshold is as good as color threshold. I went ahead and used HLS threshold:



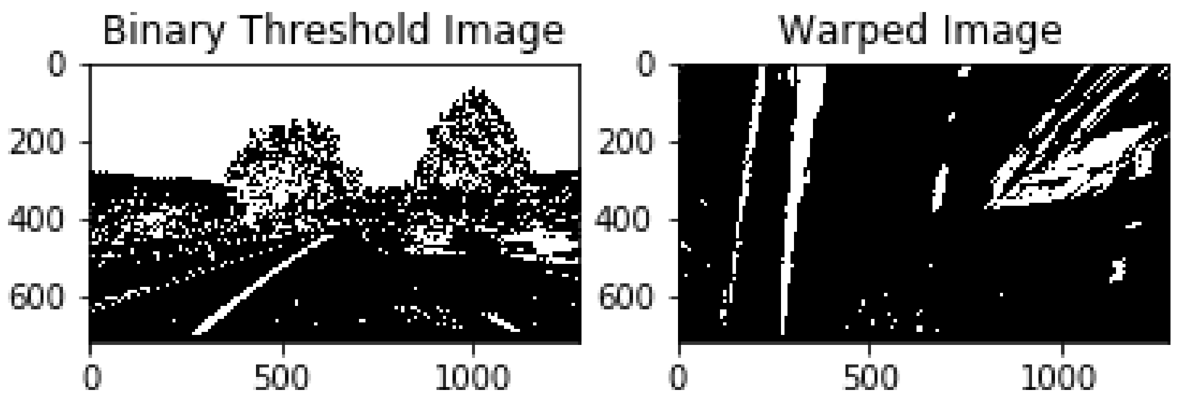
Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.

I used perspective transform to unwarp the image in cell 6:

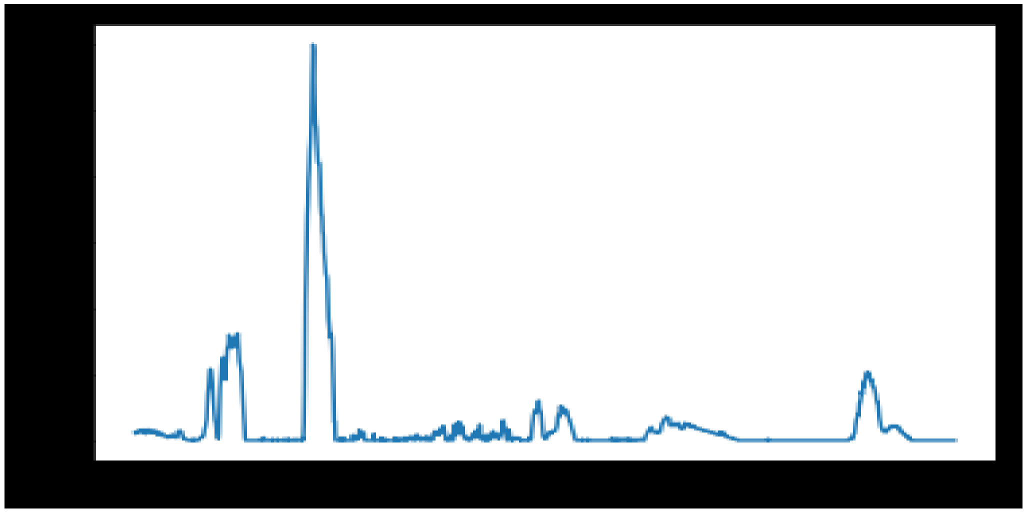
M = cv2.getPerspectiveTransform(src, dst)



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



Histogram



Identifying lane pixels started with histogram as shown above to see where the peaks are.

And then the window shape is defined to find the line to fit a polynomial. Left base and right base are obtained from the histogram peaks with a margin of 80 from the sides. Step through each window and keep finding the indices of the left and right lanes. Left and right lane pixel positions are then extracted from the lane indices. Finally, second order polynomial function is fit into the each window. Refer cell 15 for the code.

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.

Line 19 shows how the radius of curvature is identified.

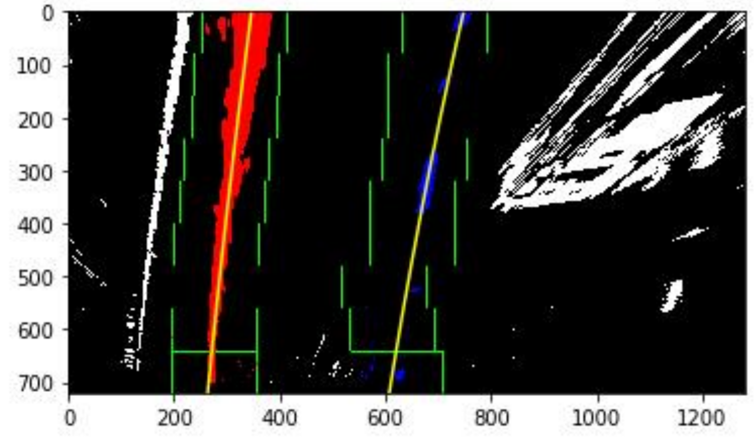
First obtain the polynomial function for left and right lanes. Identify new coefficients in meters (convert pixel to meters) And then R-curve is calculated using the formula for each of the lanes given by left_curve_radian and right_curve_radian

$$\text{Radius of curvature} = \frac{\left[1 + \left(\frac{dy}{dx}\right)^2\right]^{3/2}}{\left|\frac{d^2y}{dx^2}\right|}$$

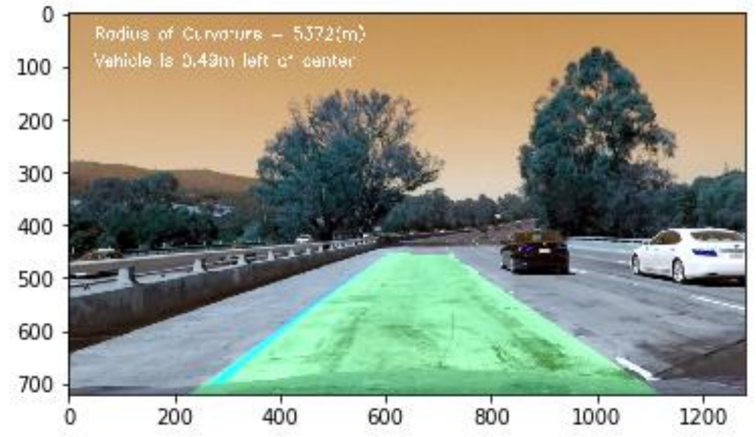
Left Radius = 1403.3

Right Radius = 640.5

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



```
<matplotlib.figure.Figure at 0x7ffa0a567d68>
```


Radius of Curvature = 5372(m)
Vehicle is 0.43m left of center
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

Challenges:

This project over-loaded, and too much to solve in less time compared to project 1. Looks like there is no transition between two projects, just heavily loaded, would be great if it is split into two.

So there were plenty of challenges like the lanes doesn’t capture well with the thresholds and lot of trial error method had to be tested. Also spotting source and destination points were tricky, again had to go through lot of experiments.

My pipeline will likely fail in large curves, in which case I have to develop more robust curvature detection functions or try another sliding window approach.