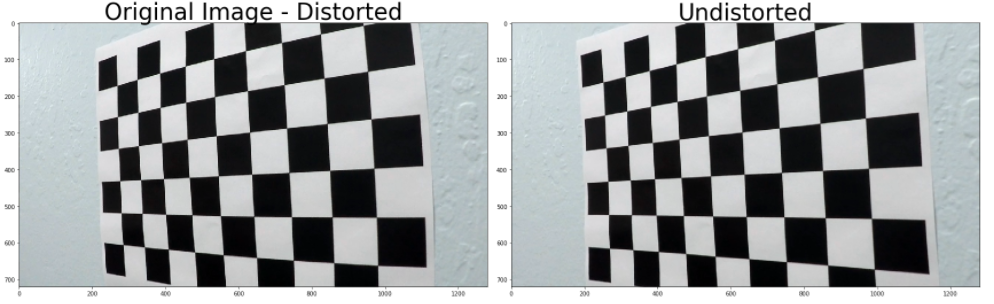
**Project: Advanced Lane Finding**

**Camera Calibration:**

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

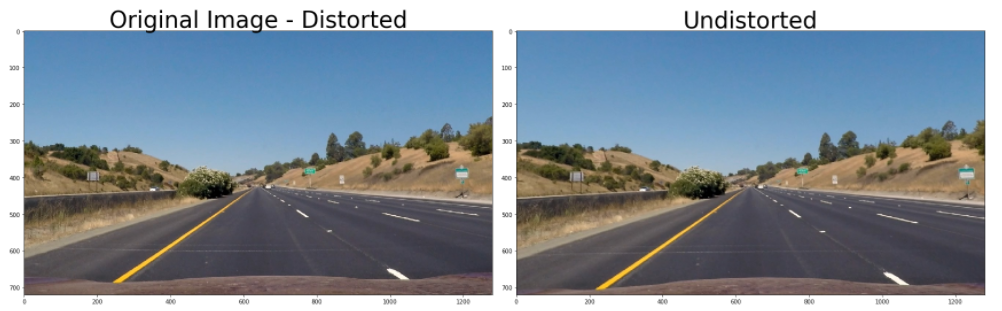
From the fifteenth line of the first code cell in the ipython file, I have uploaded I have done the preparation of the object points, then extracted the image points from the checkboard image patterns using the findChessboardCorners function in OpenCV python. Then using these two set of points I have found the undistortion matrix using the calibrateCamera function in opencv python library.

Then I have applied this distortion matrix to the test image using the undistort function in opencv library.

The following picture show the original distorted and undistorted images.

**Pipeline (single images):**

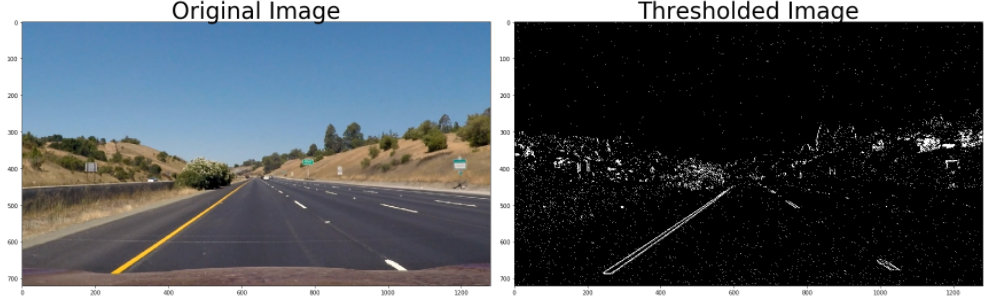
1. Provide an example of a distortion-corrected image.



1. 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.

In the code block from the line 2 to 29 I have done the threshold function which utilizes s matrix, l matrix (HLS), directional gradient and sobel thresholds to figure out the correct lane lines in the images.

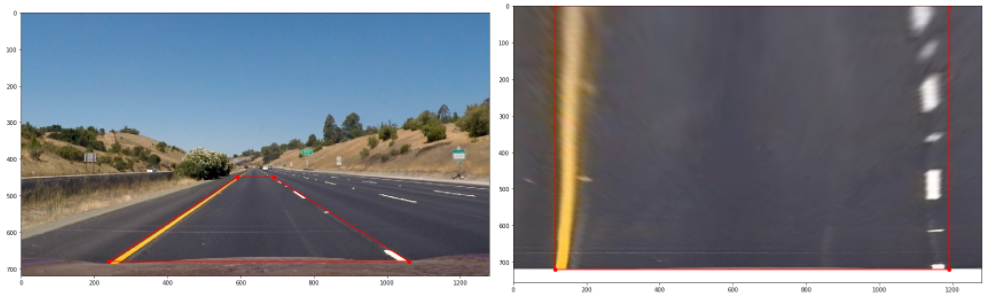
Please find the example for the same below.



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

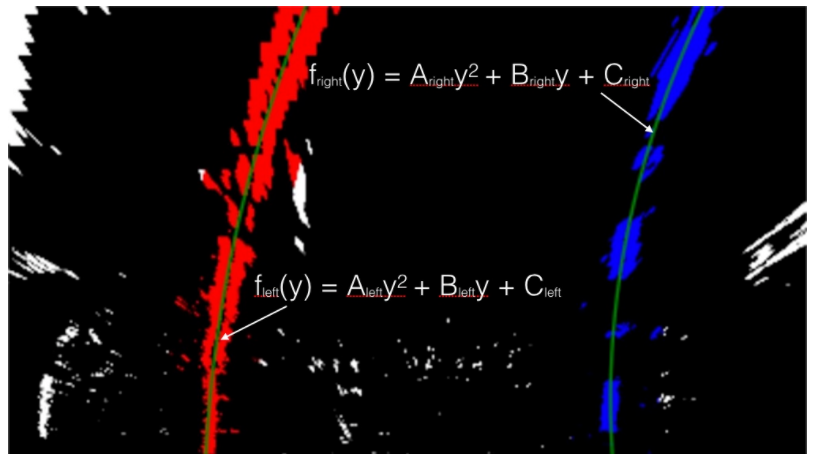
In the fourth code block of ipython notebook, from line 18 to line 31, I have used the warp function which employs the getPerspectiveTransform command from opencv to find the perspective transform the source points to the destination points on the new image.

|  |  |
| --- | --- |
| **Source Points** | **Destination Points** |
| 240, 680 | 100, 720 |
| 590, 450 | 100, 0 |
| 690, 450 | 1180,0 |
| 1060, 680 | 1180, 720 |

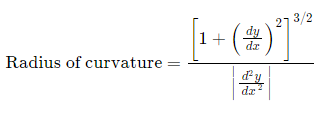


1. Describe how (and identify where in your code) you identified lane-line pixels and fit their positions with a polynomial?

In the lines from 70 to 117, in the fifth code block, I have found out the potential lane line points (X and Y), then using the polyfit function in opencv, in lines 120 and 121, I have found out the second-order equation for the curves as shown in the below diagram.



1. 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.

In the lines 185 and 186 of the fifth code block I have found out the radius of curvature of the left and right lanes using the formula:

Then I have taken the mean to find out the radius of the curvature of the lane.

In the line 155, I have calculated the deviation of the car from the center of the lane by using the center pixel image point and the center of the two lanes near the car using the right and left lane points. Then by taking the lane width as 12 inches approximately and the pixel difference between left and right lane points as 700px. Then using the meter/pixel movement is found as 0.005 using the formula meters/pixel in x-direction= 12\*0.3048/700, then using this relation and difference between the center pixel (640) and the center of the lane points, the deviation of car from the center of the lane is calculated.

From the line

In order to find the radius of curvature, I have calculated the meters per pixel in y-direction. Considering 15 meters of road represented by 720 pixels, then the meters per pixel in y-direction is 15/720=0.021. I have used this ratio along with the pixels in y directly in polyfit function to measure the radius of curvature from the same.

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

I have implemented the code from the line 178 to 194 in the code block 5 to get fill the polylines and then unwrap the image using the M inverse and project the same on the video frame. An example for the same is provided below.



**Pipeline (video):**

1. 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!).

Please find the video in the following [link.](https://drive.google.com/a/g.clemson.edu/file/d/1AaHvlsWve-dVTvszEswBSACfwYDERR3i/view?usp=sharing)

**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?**

I have encountered a lot of problem in choosing the correct threshold to identify the lane lines. The algorithm will fail for different light conditions or different road scenarios. Hence it is necessary to choose a correct value of the thresholds that is suitable for almost all of the scenarios and it requires videos of different light conditions and environments.