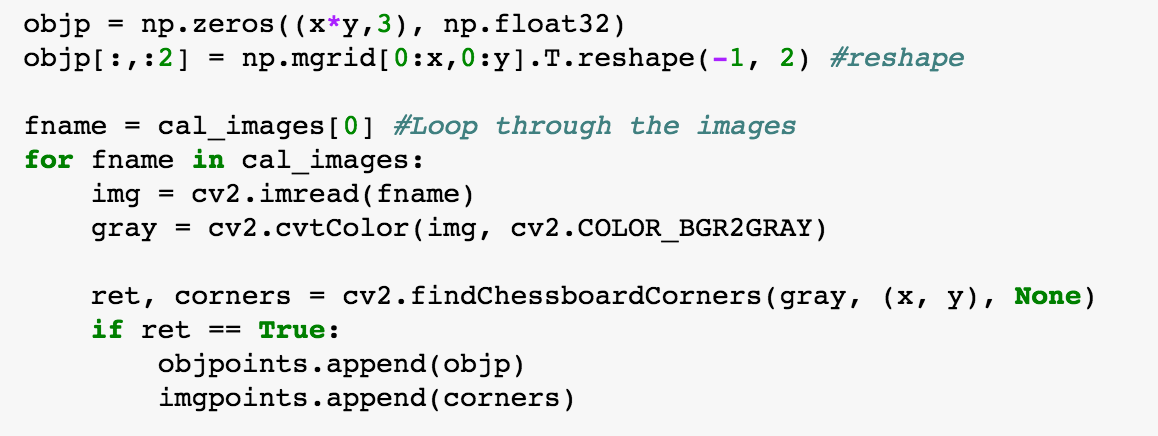
**Camera Calibration**

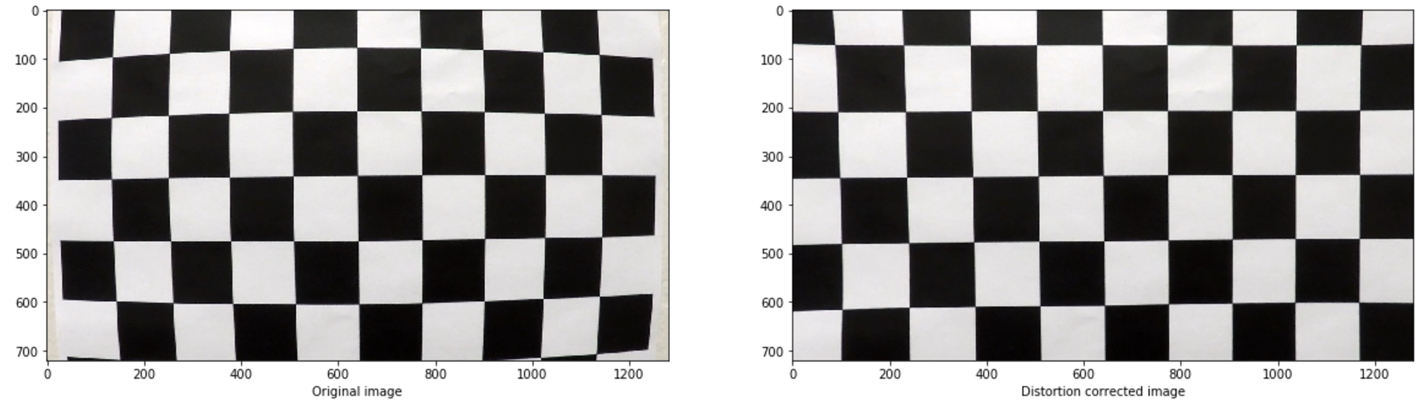
**1. Briefly state how you computed the camera matrix and distortion coefficients. Provide an example of a distortion corrected calibration image.**

I start by creating a function called “calibrate\_camera”. A list of object points are created to coordinate the chessboard corners in the real-world space (x, y, z) with z=0.



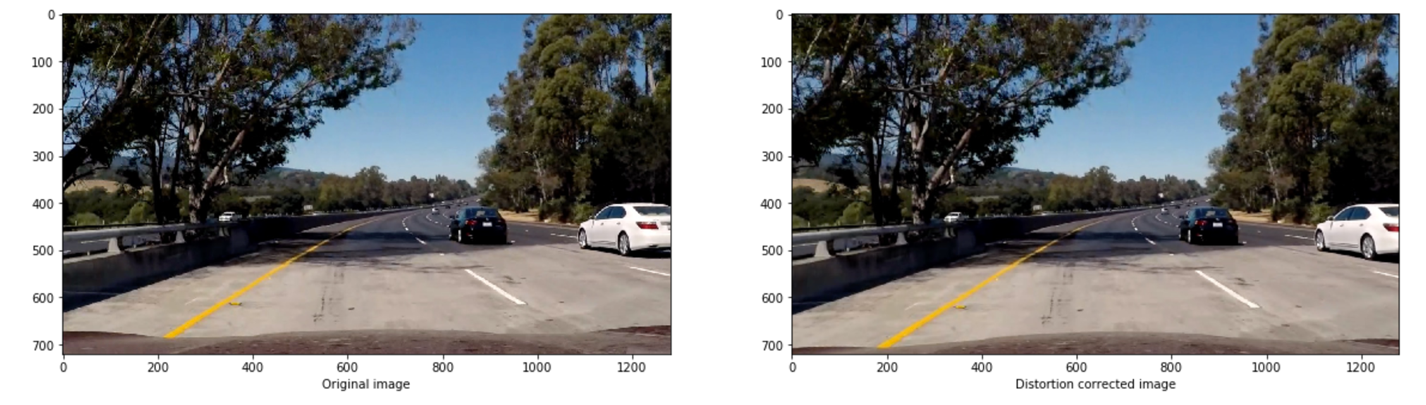
For each of the loop, the (x,y) coordinates are calculated using cv2.findChessboardCorners and stored into imgpoints array. The calibration and distortion parameters are then calculated using cv2.calibrateCamera function.

The following images show the result of camera calibration.



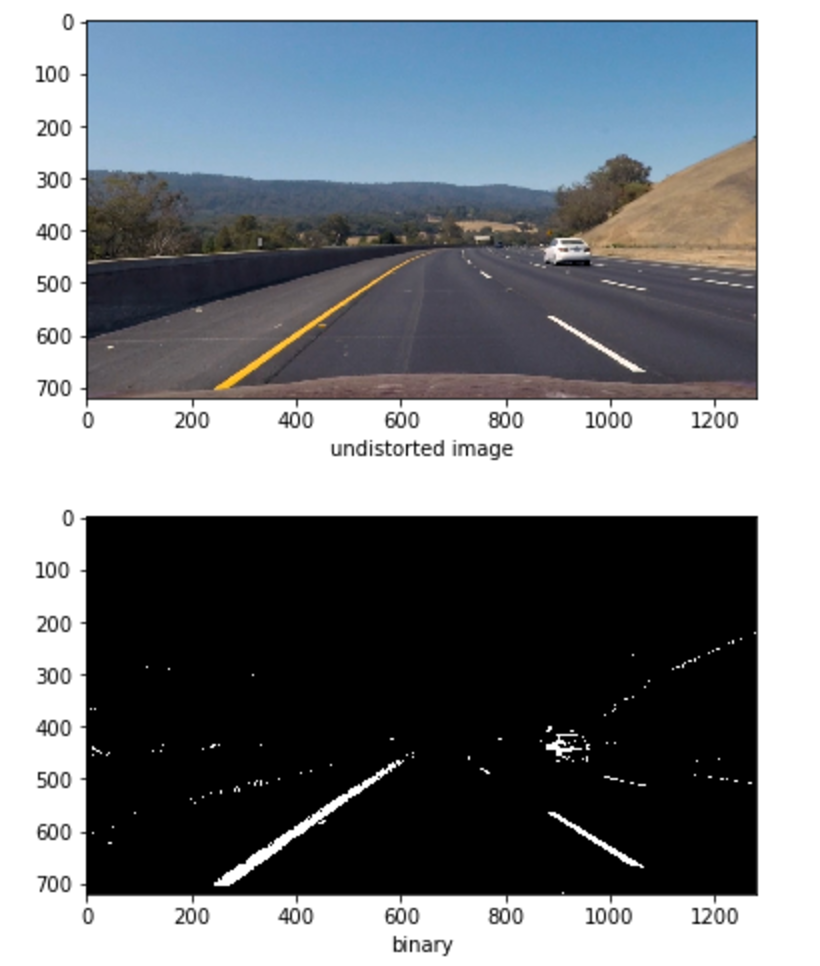
**Pipeline (single images)**

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



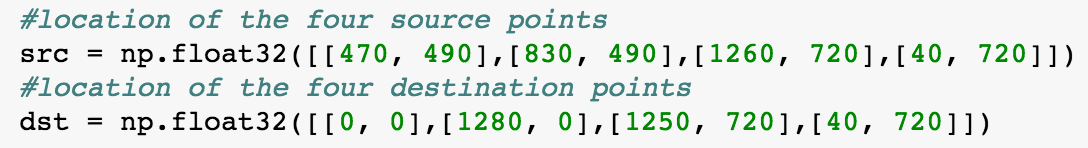
1. **Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.**

I used both magnitude (mag\_thresh function) and direction thresholds (dir\_threshold) plus the yellow and white masks (threshold function) to generate a binary image.

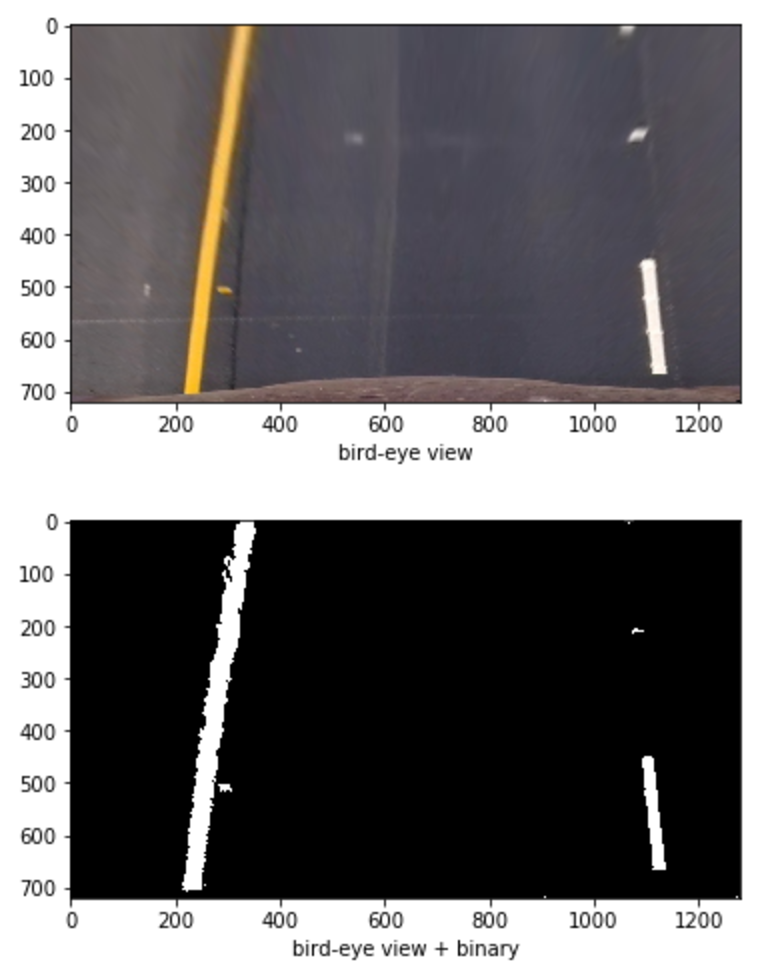


**3. Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.**

Function ‘perspective\_transform’ is used to perform the transformation. The source and destination points are as follows.

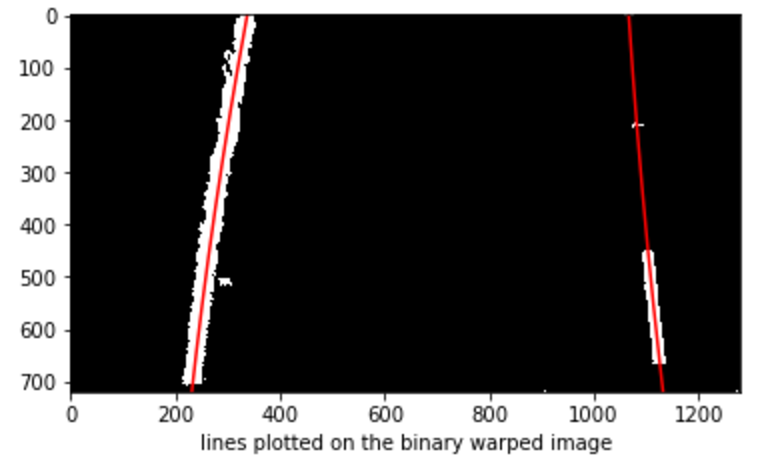


The warped images are shown below.



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

Then I fit my lane lines with a 2nd order polynomial by using the histogram method. The image will be divided into 9 windows horizontally and for each window, the position of the maximum summation of the pixel values over x-axis will be saved in the list ‘left\_lane\_inds’ and ‘right\_lane\_inds’. I then use np.polyfit to fit the 2nd order polynomial function.



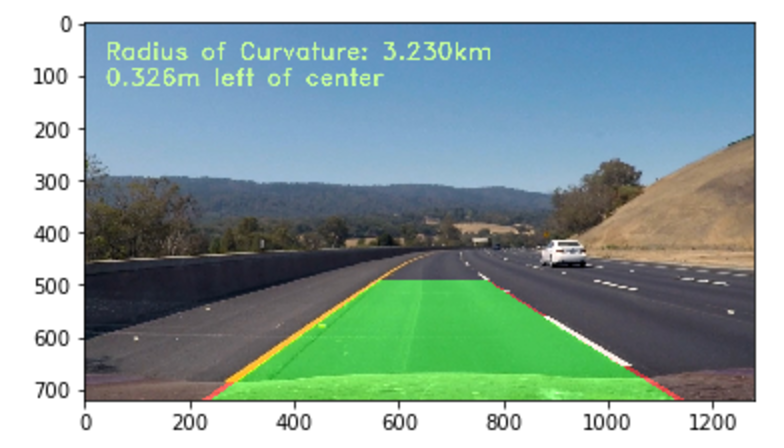
**5. Describe how (and identify where in your code) you calculated the radius of curvature of the lane and the position of the vehicle with respect to center.**

The following function is used to calculate the radius of curvature.

/Users/julienliu/Desktop/Screen Shot 2017-04-05 at 8.25.02 PM.png

The position of the vehicle with respect to the center is calculated by taking the difference of the image center point with the lane center point.

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

****

**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 see the video attached for the final output.

**Discussion**

**1. 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?**

It took me some time to find the best way to get the binary threshold. The final result should be improved with more tuning onto the threshold parameters. Same thing happened in finding the appropriate source points for perspective transformation. I notice the radius of curvature deviates significantly with a small change on the source points.