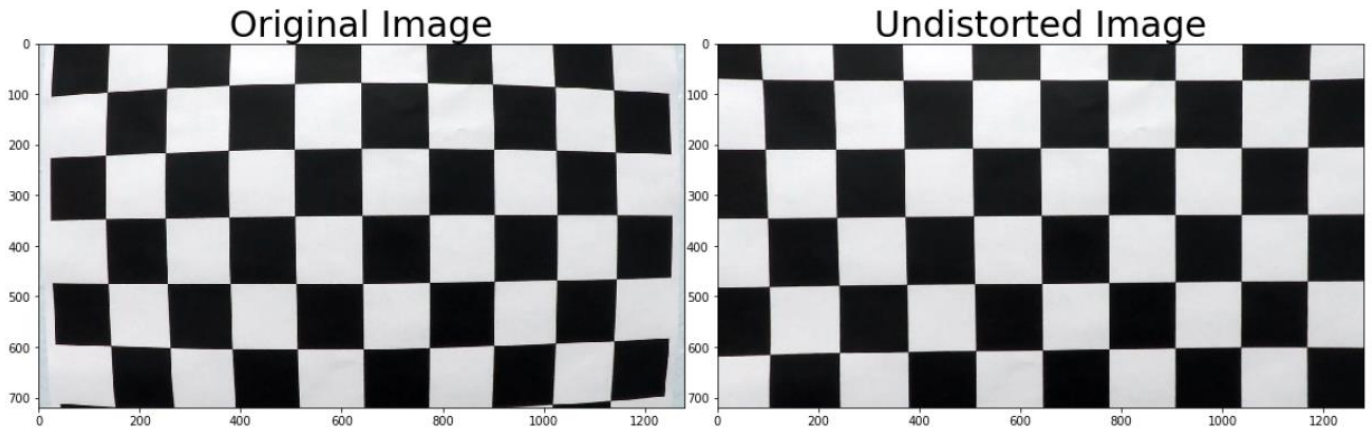
Camera Calibration:

To calibrate the camera, I used the provided images to detect edges and draw them.
In the section Camera calibration using chessboard images (Runs Once) I did this:

1- Read imaged in /camera_cal.
2- Find corners using cv2.findChessboardCorners.
3- Append to objpoints and imgpoints.
4- Calculated the Calibration output using cv2.calibrateCamera.
5- Get mtx and dist.
6- Save output as /cal_pickle.p.

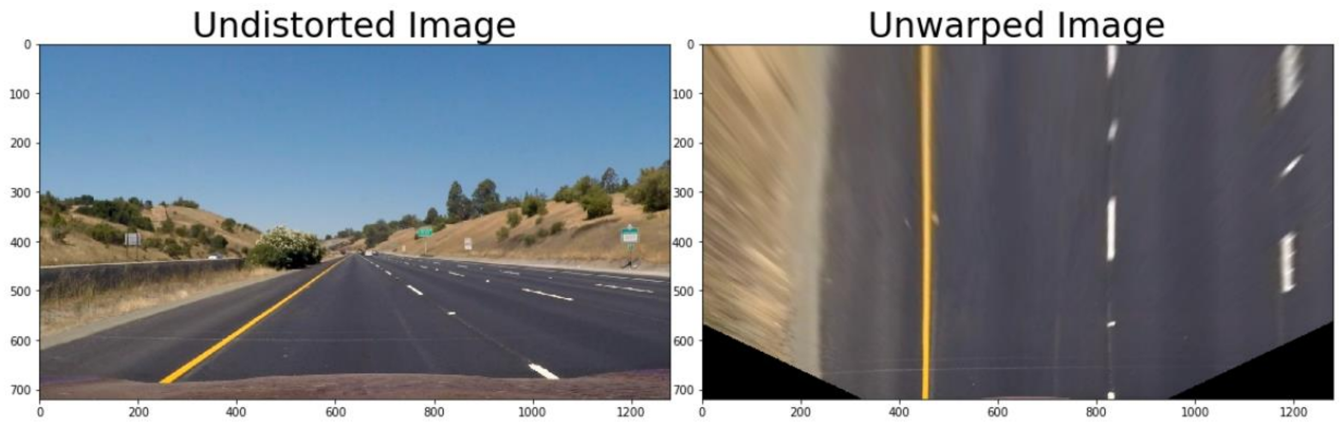
Apply a distortion correction to raw images:

In the next 2 sections I imported the calibration file and showed an example to raw image, the output was like this:



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

I did this in the section Unwarp where I used src and dst variables as params to cv2.getPerspectiveTransform Function and the output was this:



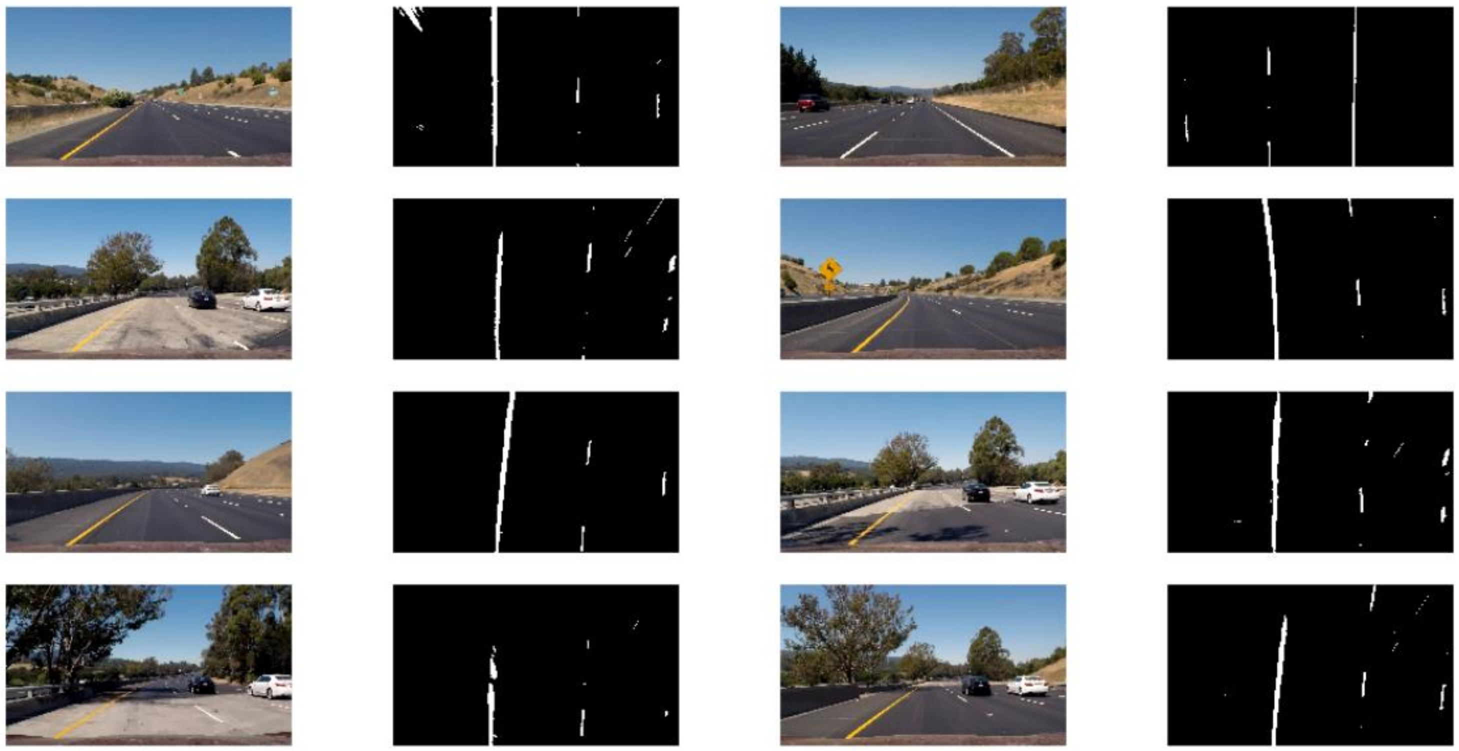
Use color transforms, gradients, etc., to create a thresholded binary image:

I did this in the section Thresholding where I used 2 functions:

1- hls_lthresh which output is the **L-channel** threshold with boundaries 210 to 255.

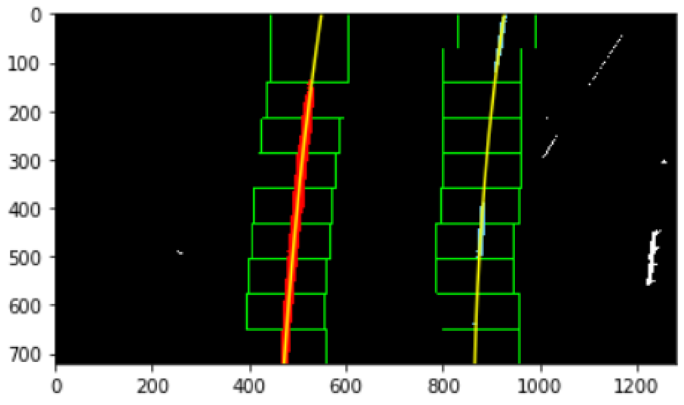
2- lab_bthresh which outputs the **B-channel** threshold with boundaries 190 to 255.

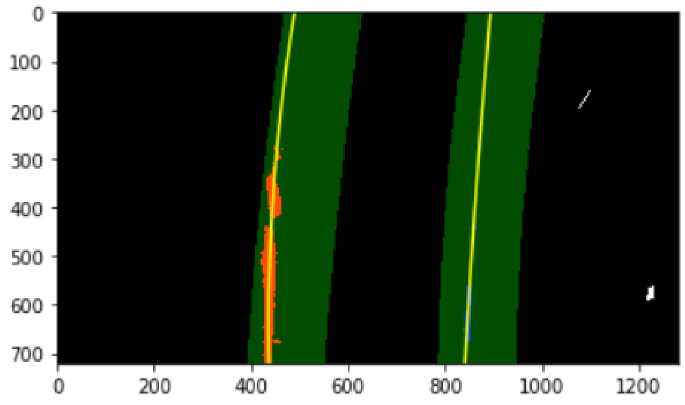
The output was this:



Detect lane pixels and fit to find the lane boundary:

I did this in the section Sliding Window & Poly-fit where I did the sliding window which search the thresholded output to find the lanes and the output was this:





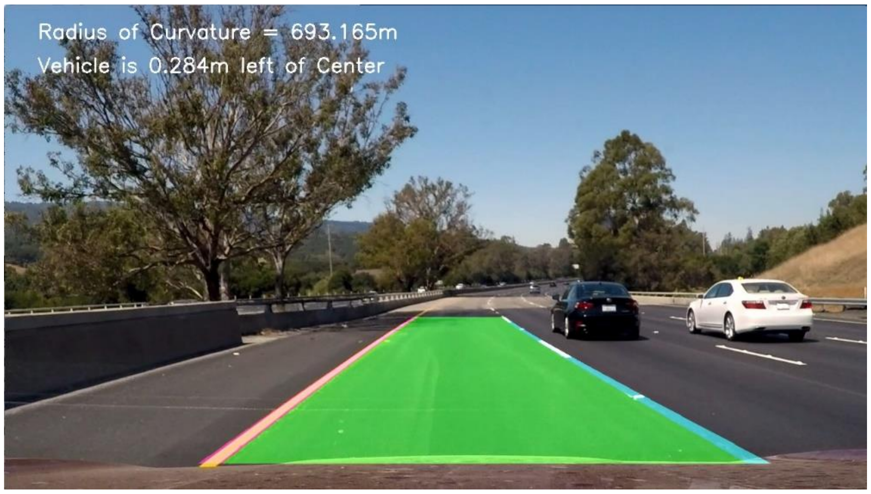
Determine the curvature of the lane and vehicle position with respect to center:

I did this in the section Calculate CurveRad and Dist. to Center.

I calculated the left and right curvatures and the car distance from the center of the mean of l_fit and r_fit intercepts.

Then, in the Draw Lane section of the code draw the left and right lanes with the middle area using the draw_lane function and the output of the calculated curvature and distance to the canter using the draw_data function

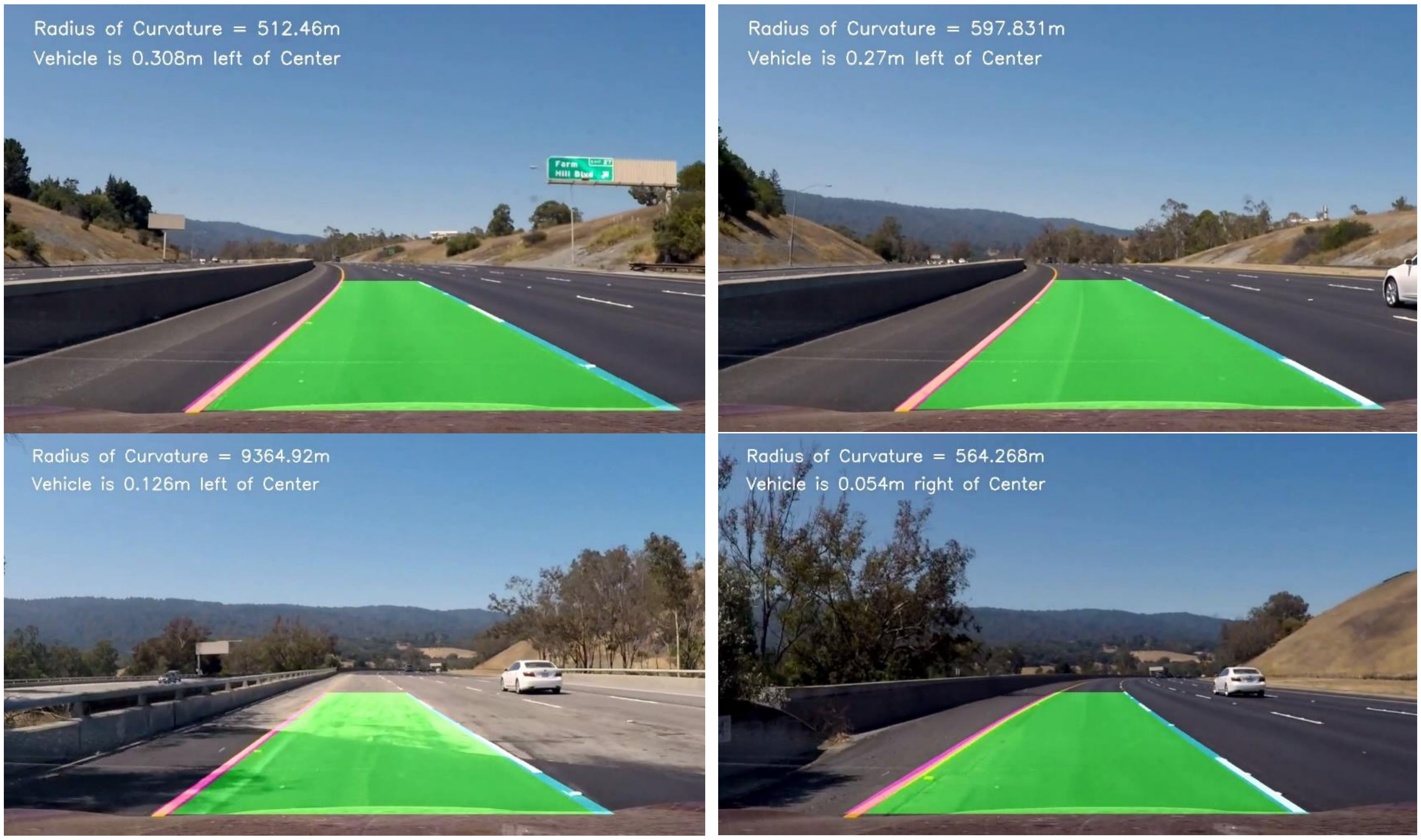
Here an example of the output:


Radius of Curvature = 693.165m
Vehicle is 0.284m left of Center

Video Processing:

Before starting this section I defined the process_image function and Line class to add some sanity check to ensure continuity and smoothness of lanes on left and right.

Some screenshots:


Radius of Curvature = 512.46m
Vehicle is 0.308m left of Center
Radius of Curvature = 597.831m
Vehicle is 0.27m left of Center
Radius of Curvature = 9364.92m
Vehicle is 0.126m left of Center
Radius of Curvature = 564.268m
Vehicle is 0.054m right of Center

Discussion:

1- In the 22nd second in the video when the road color changed, I had a problem that lines are extreamly bad, so I kept the last output to smoothly translate from one line to another.
2- The LAB color space I found very useful and was missed in the lictures.
3- Some improvenets can be done on the thresholding to avoid problems like the one in the first point.
4- I think that the code will fail if the road was more narrow and have more lanes in the field of view of the camera.