# Advanced Lane Finding Project

The goals / steps of this project are the following:

- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- Detect lane pixels and fit to find the lane boundary.
- Determine the curvature of the lane and vehicle position with respect to center.
- Warp the detected lane boundaries back onto the original image.
- Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

**Ipython notebook : lane_finder_notebook.ipynb**

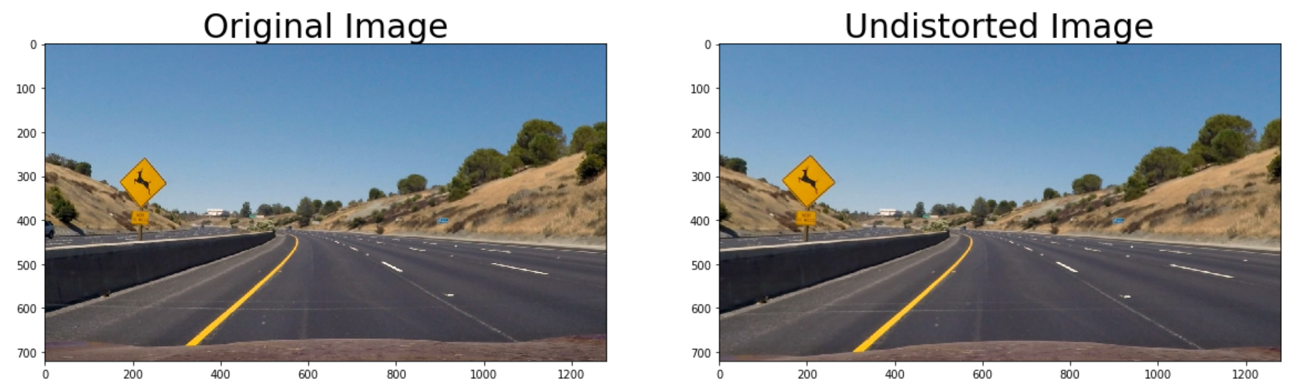
## Camera Calibration

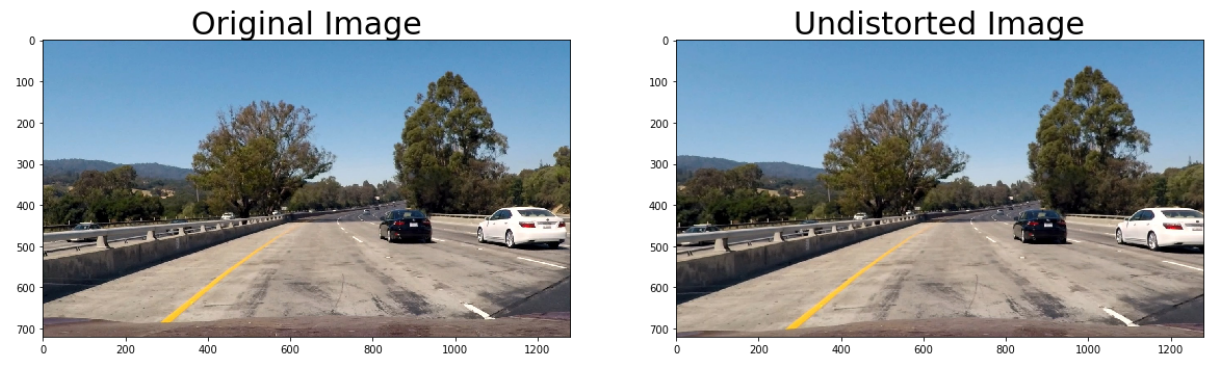
objpoints is used to store  3d points in real world space
imgpoints is used to store 2d points in image plane.

`objpoints` is just a replicated array of coordinates, and `points` will be appended with a copy of it every time I successfully detect all chessboard corners in a test image. `imgpoints` will be appended with the (x, y) pixel position of each of the corners in the image plane with each successful chessboard detection.

To compute the camera calibration and distortion coefficients `cv2.calibrateCamera()` function is used. I applied this distortion correction to the test image using the `cv2.undistort()` function and obtained following result:

### Distortion correction to raw images





## Color transforms, gradients to create a thresholded binary image and using perspective transform

## Following functions were used:

**binarize**(img, s_thresh, sx_thresh,l_thresh)

- To generate threshold binary image
- Img : Raw image
- S_thresh : Threshold saturation channel
- Sx_thresh : Threshold x gradient
- L_thresh : Threshold lightness

To binarize , image is converted into HLS color space. Further cv2.Sobel is used to take derivative in x .

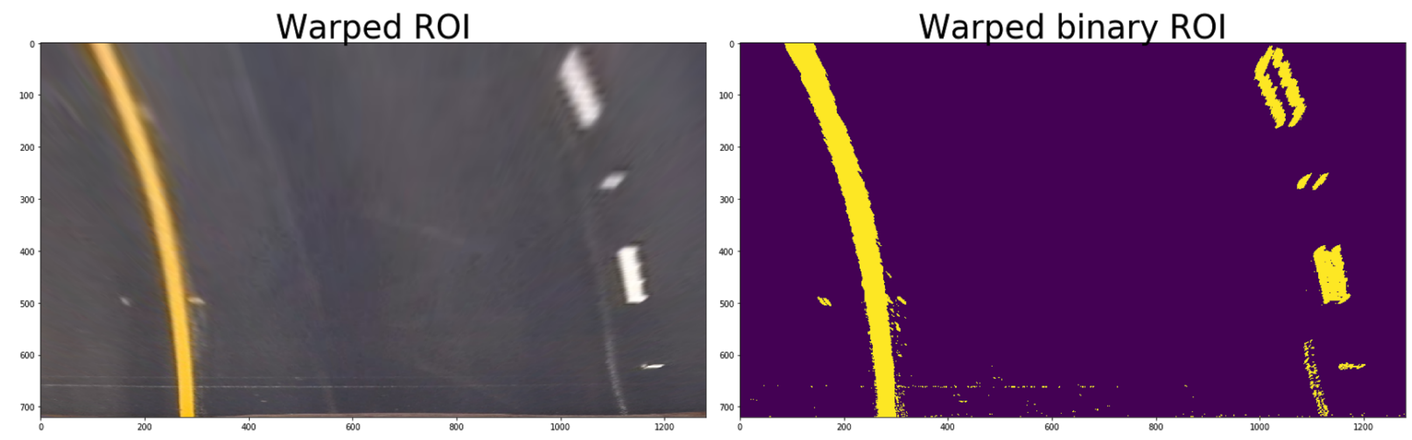
**get_perspective_transform**(image, display=False):

- Takes undistorted image as a input and returns image after perspective transform

- cv2.getPerspectiveTransform and cv2.warpPerspective is used to get the desired result

**warp_binarize_pipeline**

- Step 1: image is undistorted
- Step 2: Undistorted image is passed to get_perspective_transform
- Step 3: Output of step 2 is fed into binarize() to get thresholded binary image



## Detect lane pixels and fit to find the lane boundary

**find_window_centroids** provided by udacity is used to find window centroids .

Further **np.polyfit** is used to fit all the (x,y) points found within the windows identified .

For this part code is present in third cell of notebook mentioned above .

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

**curvature_radius(image, left_fit, right_fit) :** Function is used to find curvature radius

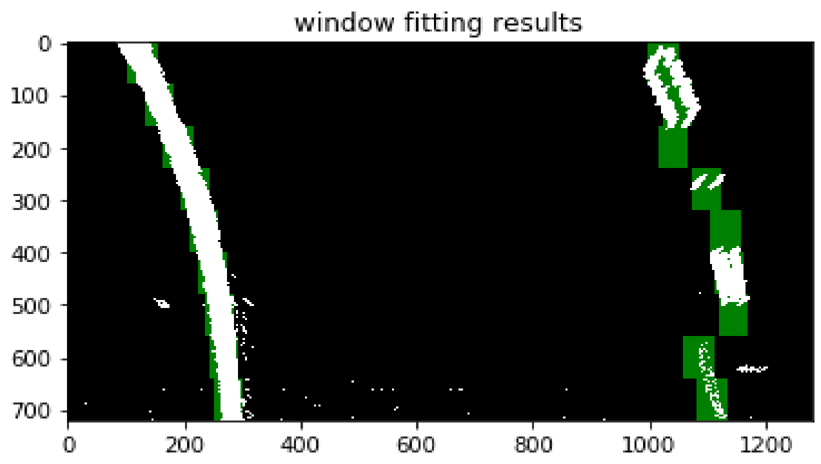
Image : Perspective transformed image

Left_fit/Right_fit: results after using np.polyfit()

**pos_from_center(image, leftx_base, rightx_base):** Function is used to find position from center

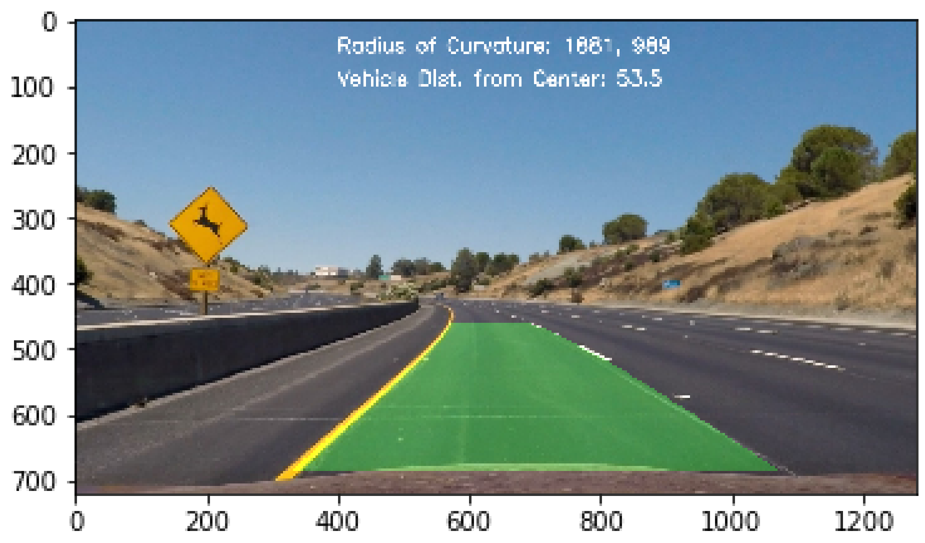
Image : Perspective transformed image

Leftx_base/rightx_base : X coordinate of left and right line identified.



## Warp the detected lane boundaries back onto the original image. Output visual display of the lane boundaries

- **Further image is warped on original image . Boundaries are identified and region between boundaries is shaded with Green color**



Radius of Curvature: 1881, 989
Vehicle Dist. from Center: 53.5
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

## Pipeline:

Further pipeline is constructed to take a video feed and save the output in a video **.**

**Pipeline.py :** Takes care of processing frame by frame images from video

**Advance_lane_functions.py :** Has all the useful functions described above in the writeup

**Output video :** lane_output.mp4

**Discussion:**

The most challenging part of this project was to write  lane line detection or thresholding algorithm which is robust for the project video. Tried different thresholding techniques before settling for the final approach.

Initially lane detection failed in sudden change of lightness or in presence of shadows. Used smoothing from previous frame to avoid bad detection which I feel is more of a hack. As suggested by mentor I will explore contrast gain control technique to improve it further.

This approach might fail in sharp curves . There is a scope of improvement to pass challenge video.