Contents

1. Revision History

Version 1 : Iniitial Version

Version 2: Released with following changes:

a) ROI based Filtering improved to be trapezium from left and right end to midpoint
b) Few parameters changed for better filtering

2. Introduction:

Following is the organization of the writeup.

a) Goals defines the goals of the section
b) For each step following are defined
    a. Logic followed
    b. Pointer to (or actual) code
    c. Example output
    d. Pointer to actual output
c) Conclusions and next step

3. Output files:

Following is a quick reference to the folder /files:

a) Writeup.pdf -> Pdf of write up file
b) Advanced_car_lane_finding.pynb -> Python script for lane finding
c) Output_images/final_images -> All final results for all images and video
d) Output_images/undistorted_chessbard -> Results of undistortion on chessboard images
e) Output_images/thresholded_images -> Threshold image(Normal view)
f) Output_images/warped_images-> Threshold image (Warped view from top)
g) Output_images/lanes_from_top -> Filled up lanes from top view

4. Goals:

Goals of the project are the following:

a) Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
b) Apply a distortion correction to raw images.
c) Create a thresholded binary mage
d) Apply a perspective transform to rectify binary image ("birds-eye view").
e) Detect lane pixels and fit to find the lane boundary.
f) Determine the curvature of the lane and vehicle position with respect to center.
g) Warp the detected lane boundaries back onto the original image and display along with curvature and vehicle position

3. Camera Calibration & Distortion correction

3.1 Logic used:

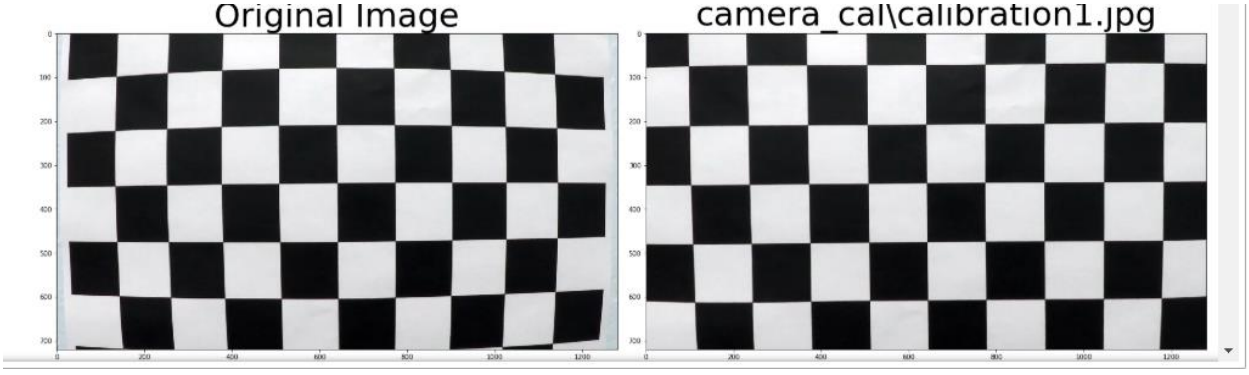
Camera calibration was done by following steps:

a) Find the chessboard corners - *ret, corners = cv2.findChessboardCorners(gray, (nx, ny), None)*
b) Find the objpoints and imgpoints
    a. Imgpoints are a subset of corners found in previous step
    b. Objpoints are projected based on assumed square length
c) Use cv2.calibrateCamera to calibrate Camera with above values
d) Use cv2.undistort to distort image based on output from Camera calibration

3.2 Code

a) Camera Calibration code is present in Cell 4 and Cell 5 of the pynb file

3.3 Example output



3.4 All output

Undistorted output of all chessboard images is present in "output_images/undistorted_chessboard" folder

4. Creating a thresholded Binary image:

4.1 Logic Used

Threshold is based on a combination of following:

- Sobel magnitude gradient with a threshold of 10 to 180
- Combination of R, G and B to find white and yellow lines
  - Y[(R/B> 1.3) & (G/B> 1.3)] = 255
  - W[(R/B < 1.2) & (G/B < 1.2) & (R/B >0.8) & (G/B > 0.8) & (gray >180)] = 255
- Region of Interest – Y > img_height/2, x > 100 , x < img_width – 100

4.2 Sample Code

a) This is present in Cell # 6

4.3 Sample output

4.4 All output

Output of thresholded binary image is present in “thresholded_images”

5. Perspective transform of thresholded images:

5.1 Logic used

The warped image is obtained by calling cv2.Perspective transform of src, dst matrix derived from one of the images. Following values are used

warp_src = np.float32([[297,649],

[991,649],

[619,434],

[656,434]])

```
warp_dst = np.float32([[458,720],
        [823.5,720],
        [458,0],
        [823.5,0]])
```

5.2 Pointer to code

The code here is below and present in Cell 6.

```
def getViewFromTop(orig_img,img_to_warp,src,dst):
    img_size = (img_to_warp.shape[1], img_to_warp.shape[0])
    M = cv2.getPerspectiveTransform(src, dst)
    Minv = cv2.getPerspectiveTransform(dst, src)
    warped = cv2.warpPerspective(img_to_warp, M, img_size, flags=cv2.INTER_LINEAR)
    return warped, M, Minv
```

5.3 Sample output

5.4 All outputs

All outputs are available at output_images\warped_images folder

6. Lane detection

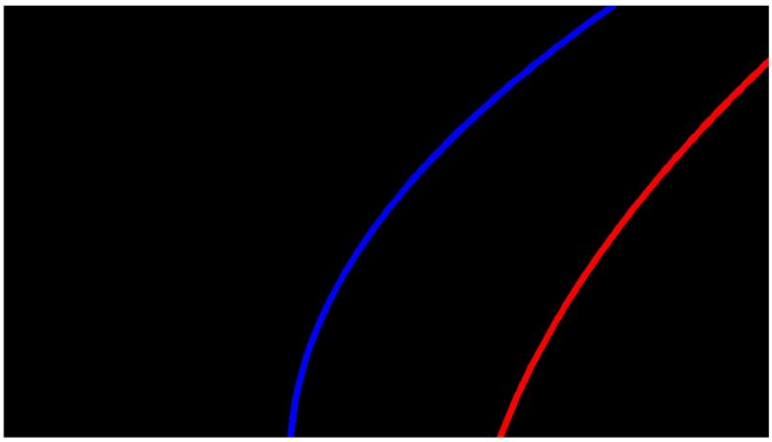
6.1 Logic used

a) Use highest value of Histogram of bottom part of the filtered image . 2 starting points are used ,1 to left of midpoint and 1 to right of midpoint for left and right lanes
b) Use a search rectangle based on the “Polynomial fit” of the line as we create the line
c) Add all points in search rectangle to a “lane_inds” array
d) Fit a polynomial of order 2 to find the left and right lane
e) For video, new points were searched based on line from previous images and a margin. If no good lanes are found, then the lanes are dropped, and a search done from scratch

6.2 Code used

Code consists of 2 functions – 1 to find lanes without any previous data and 1 for repeat frames. Code is present in Cell # 7

6.3 Sample output



6.4 All outputs

All outputs are present in output_images\lanes_from_top folder

7. Calculating curvature and midpoints of lanes and car

7.1 Logic

The logic is just straightforward calculation of curvature from polynomial fit . For midpoints , the lanes x values at y = img_height is taken and averaged.

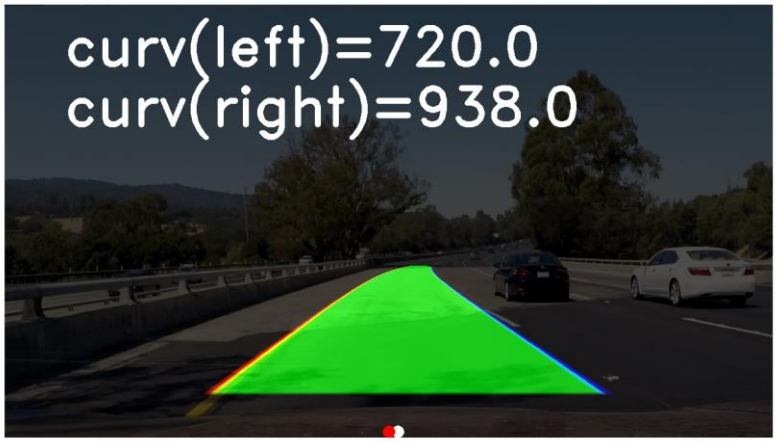
7.2 Code used

```
def calculate_curvature ( y_input,a ,b):
   num = ( 1 + (2*a*y_input + b)**2) ** (3/2)
   den = abs ( 2 *a)
   return (num//den)
```

*left_curvature = calculate_curvature (np.max(ploty*ym_per_pix),left_fit_in_meters[0] ,left_fit[1])*
*right_curvature = calculate_curvature (np.max(ploty*ym_per_pix),right_fit_in_meters[0] ,right_fit[1])*

Sample output:

The red dot shows center of the car and white dot shows center of the lanes.



7.3 All outputs

All outputs are at output_images\final_images folder

8. Final image

8.1 Logic used

Final output is a combination of following:

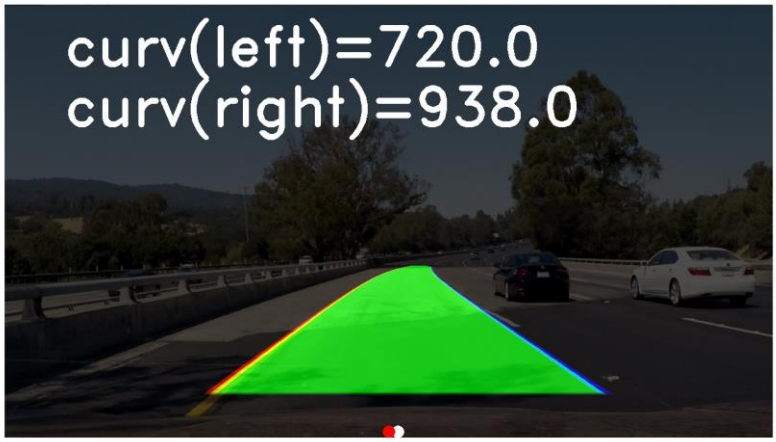
a) Original image
b) 2 Lanes filled up
c) Curvature listed as text
d) Red dot (for center of camera) and white dot (for center of lane)

8.2 Code used

Code used is in Cell #

8.3 Sample output

This is same as previous section



8.4 All outputs

All outputs are at output_images\final_images folder.

The vide output of “project_video.mp4” is in “output_video.mp4” file in same folder

9. Conclusion and areas to improve

The project had challenges mainly in the thresholding part. I still could not use the Sobel gradient or HLS/HSV value effectively despite multiple attempts. These are areas to improve. The challenge video is also presented in the project, but is not perfect

10. Appendix:

10.1 Explanation of Python code:

a) Cell # 1,2 and 3 -> Used for Set up. Import required libraries, set global variables and some helper functions
b) Cell # 4-> Function to calibrate Camrea
c) Cell # 5 -> Thresholding functions
d) Cell # 6 -> Lane Finder functions
    a. Calculate histogram (*hist*)
    b. Find average x and all points in a rectangle(*find_average_x_and_points_in_window*)
    c. Find pixels for a lane and fit a polynomial (find_lane_pixels_and_fit_poly)
    d. Find pixels for a lane and fit a polynomial based on previous frame info (find_lane_pixels_and_fit_poly_repeat)
    e. Calculate curvature of a polynomial (*calculate_curvature*)
e) Cell # 7 -> Function to process an image (process_pixel)
f) Cell # 8 - > Calibrate camera and undistort chessboard images
g) Cell # 9 -> Call process_pixel to output images with line on set of image
h) Cell #10 -> Find lanes on a video