Advanced Lane Finding

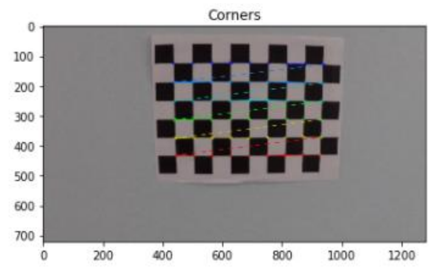
Writeup – Vinicius Abrão da Silva Marques

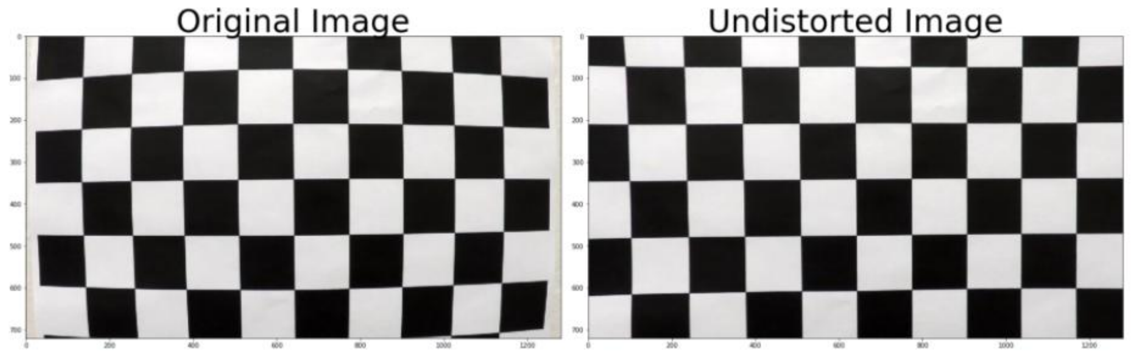
The goals / steps of this project are the following:

- 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.

1 - Compute the camera calibration matrix and distortion coefficients given a set of chessboard images:

The first step is to load the chessboard figures located at 'camera_cal/calibration*.jpg', in which * means the number of the image. After that it is prepared the objpoints list with the 3d points of corners in real world space and the imgpoints list with 2d points in image plane. Given 9x6 chessboard images and the command cv2.findChessboardCorner it is possible to obtain those lists. After I draw and display the corners with command cv2.drawChessboardCorners. With cv2.calibrateCamera it is possible to compute the parameters mtx and dist. The undistorted image is computed finally with cv2.undistort. The Figures below shows the corners found, the original and the undistorted images.



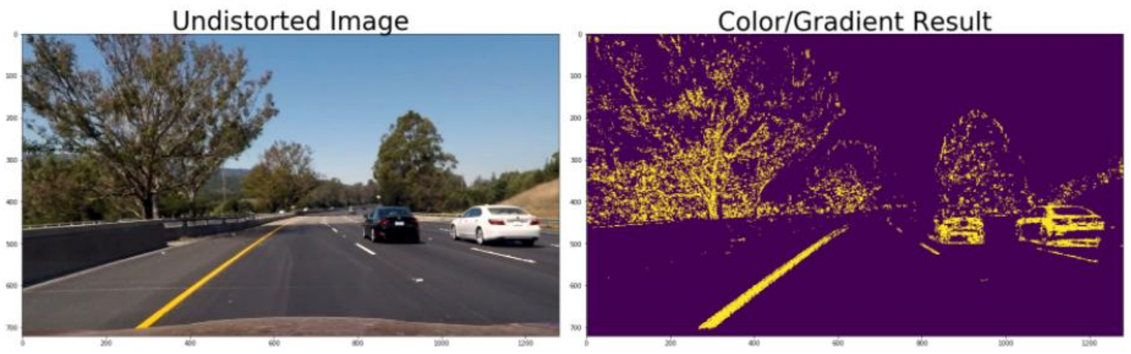


2- Apply a distortion correction to raw images:

The road images are loaded from the path 'test_images/test*.jpg' . The undistorted image is computed with cv2.undistort as before. The original and the undistorted images from the test6.jpg file are presented below.

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

It is implemented the color_gradient() function to convert the figure to HLS color space and separate the S channel. With the Sobel-X, I take the derivative in x and the absolute x derivative to accentuate lines away from horizontal. The values of sx_thresh=(20, 100) and s_thresh=(170, 255) are defined to threshold x gradient and threshold color channel, and finally I combine the two binary thresholds to create a thresholded binary image, as illustrated below.

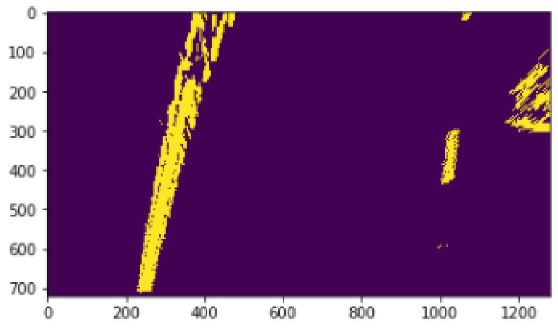


4- Apply a perspective transform to rectify binary image ("birds-eye view"):

The command cv2.getPerspectiveTransform is used to compute the M matrix and with the cv2.warpPerspective it is possible to generate the "birds-eye view". The scr and dst are as following:

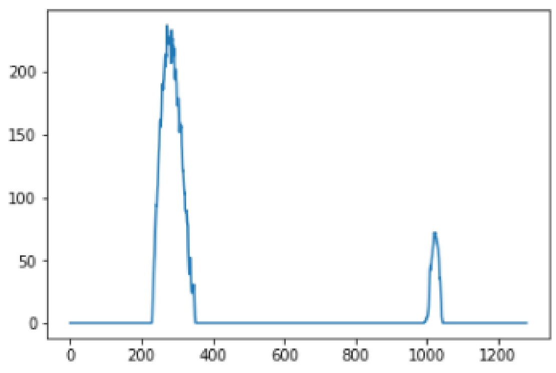
```
#top/left,top/rigth,bot/right,bot/left

src = np.float32([(550,470),(750,470),(imshape[1],imshape[0]),(200,imshape[0])])

dst = np.float32([[offset, ymin], [xmax-offset, ymin],

                        [xmax-offset, ymax],

                        [offset, ymax]])
```

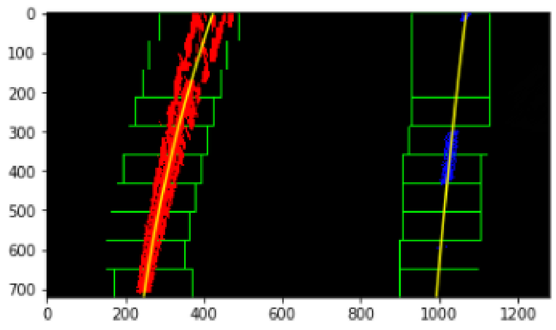
The Figure below presents the result.



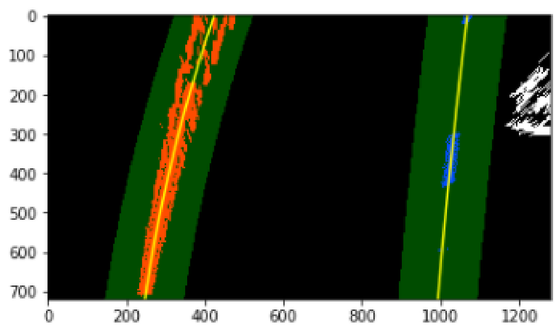
5 - Detect lane pixels and fit to find the lane boundary:

Two different algorithms were implemented to detect the lane lines. The sliding window's method computes first the histogram of the bottom half of the image to localize the line pixels and after that fit a second order polynomial to each using `np.polyfit`, as presented in the following Figures.





The second method fits polynomials based on a search from prior, looking to find pixels in a limited region near to the curve found before, as presented by green (the search area) and yellow (polynomials) in the Figure bellow. According to Udacity, "this will improve speed and provide a more robust method for rejecting outliers".



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

The calculation of R_curve (radius of curvature) and the position of the vehicle with respect to center are done as the following code:

```
    # Define y-value where we want radius of curvature

    # We'll choose the maximum y-value, corresponding to the bottom of the image

    y_eval = np.max(ploty)

    left_fit_cr = np.polyfit(lefty*ym_per_pix, leftx*xm_per_pix, 2)

    right_fit_cr = np.polyfit(righty*ym_per_pix, rightx*xm_per_pix, 2)

        # Calculate the new radius of curvature

    left_curverad = ((1 + (2*left_fit_cr[0]*y_eval*ym_per_pix + left_fit_cr[1])**2)**1.5) / np.absolute(2*left_fit_cr[0])

    right_curverad = ((1 + (2*right_fit_cr[0]*y_eval*ym_per_pix + right_fit_cr[1])**2)**1.5) / np.absolute(2*right_fit_cr[0])
```

That basically implements the equation,

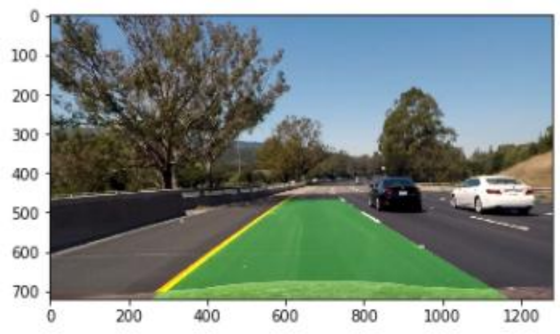
$$R_{curve} = \frac{(1+(2Ay+B)^2)^{3/2}}{|2A|}$$

to compute the radius of curvature and takes, according to the Udacity, with the following assumption to compute the position of the vehicle: “the camera is mounted at the center of the car and the deviation of the midpoint of the lane from the center of the image is the offset we're looking for”. As with the polynomial fitting, to convert from pixels to meters I assume that:

```
# Define conversions in x and y from pixels space to meters
ym_per_pix = 30/720 # meters per pixel in y dimension
xm_per_pix = 3.7/700 # meters per pixel in x dimension
```

7 - Warp the detected lane boundaries back onto the original image:

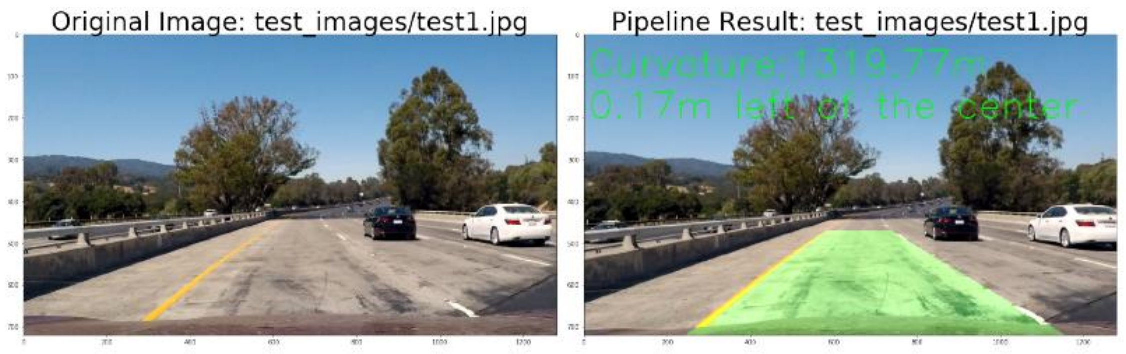
Here the cv2.warpPerspective is used again, however taking the inverse matrix M, which can be computed with the inv command from numpy.linalg, and with the polynomials curves found before. The Figure bellows presents the final result:

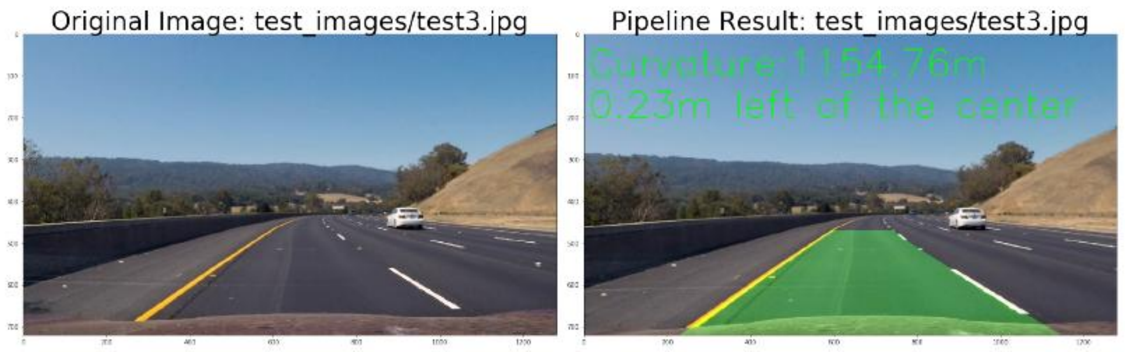


8- Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position:

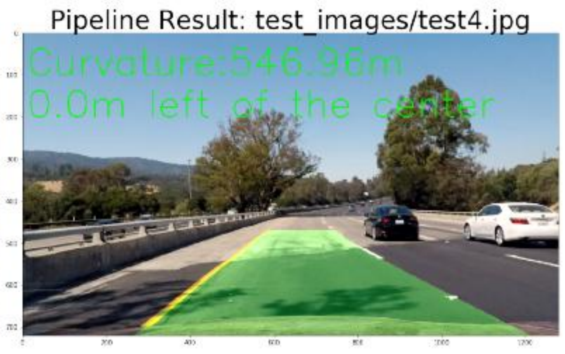
A pipeline is in the sequence implemented in order to input the raw images and output the figure with the detected lane boundaries. The pipeline also implements an outlier rejection and a filter to smooth the lane detection over frames (taking the last 3 frames to compute mean curves). The method based on a search from prior are considered unless some conditional cases are found, that takes into account changes in the curvature from a frame to the next and the coefficients of the polynomials. Finally the curvature of the lane and vehicle position with respect to center are printed over the figures.

The pipeline is evaluated over the 6 figures from the path 'test_images/test*.jpg', that results in the following Figures which also can be found in the path 'output_images/test_images/test*.jpg'.

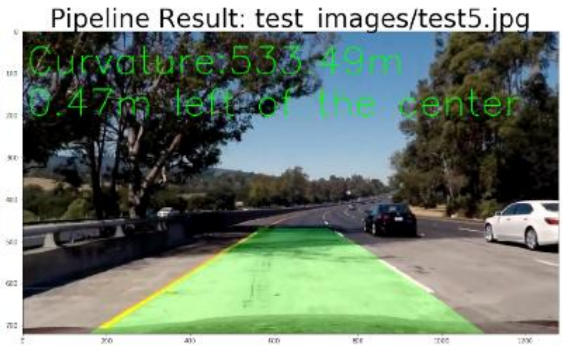

Original Image: test_images/test1.jpg
Pipeline Result: test_images/test1.jpg
Curvature:1319.77m
0.17m left of the center


Original Image: test_images/test3.jpg
Pipeline Result: test_images/test3.jpg
Curvature:1154.76m
0.23m left of the center

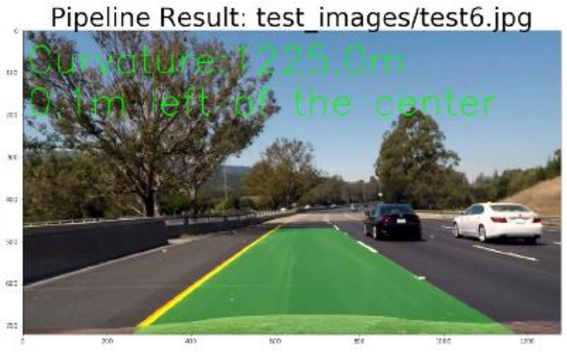
Original Image: test_images/test4.jpg


Pipeline Result: test_images/test4.jpg
Curvature:546.96m
0.0m left of the center

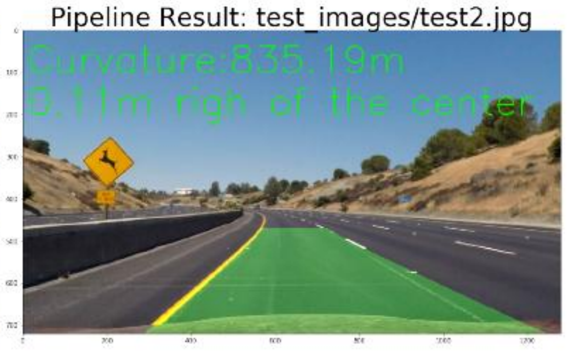
Original Image: test_images/test5.jpg


Pipeline Result: test_images/test5.jpg
Curvature:533.49m
0.47m left of the center

Original Image: test_images/test6.jpg


Pipeline Result: test_images/test6.jpg
Curvature:1225.0m
0.1m left of the center

Original Image: test_images/test2.jpg


Pipeline Result: test_images/test2.jpg
Curvature:835.19m
0.11m righ of the center

The same pipeline is used to compute and to print the curvature of the lane and vehicle position with respect to center over the frames of one video. The output video can be found in the path 'output_images/ project_video.mp4'.

- Discussion

Issues and Challenges

One of the main challenges found in the project is to define better values of scr and dst to the perspective transform, since different combinations may result in better results latter. The best results were found with the given values, however, maybe in different scenarios those values may be changed.

Also I faced an error in my first implementation related to the radius of curvature computation. It was observed that the polynomial curve coefficients had not taken into account the scale factor from pixels to meters. After to compute again the coefficients considering all the scale factors it was possible to better compute the radius of curvature.

Points of failure & Areas of Improvement

The pipeline did not correctly map out curved lines and fail in the challenge videos. It was observed problems to solve when there are pavement color changes in the direction of the road, making the algorithm confused. Some extra filters must be implemented in future works to deal with those scenarios, making sure that is taking only the lane lines.