PROJECT 2 - REPORT

ADVANCED LANE FINDING

INTRODUCTION

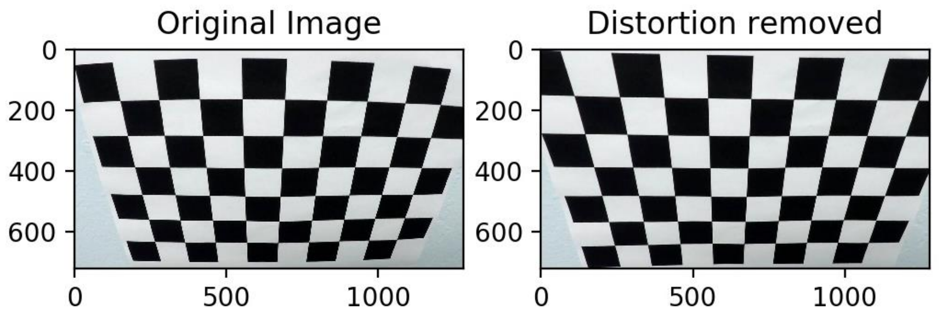
The goal of this project is to develop a software pipeline to determine lane lines and make it robust to different road conditions.

PIPELINE

1) Camera calibration:

The first step is to obtain the camera calibration and distortion correction values. This is obtained by using one of the calibration images provided, compute the matrix, use it in cv2.undistort() to remove the image distortion.

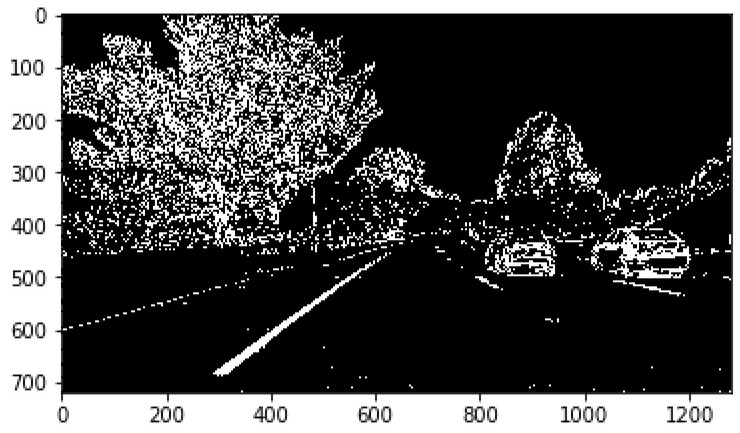
Example: calibration2.jpg



2) Computing thresholded binary image:

The function in the code is threshold(). First, the image is converted from RGB to HLS. Then each required channel is separated. Sobel filter in x-direction is applied to the L-channel and S-channel is thresholded. Both the results are then combined to produce one thresholded image called binary.

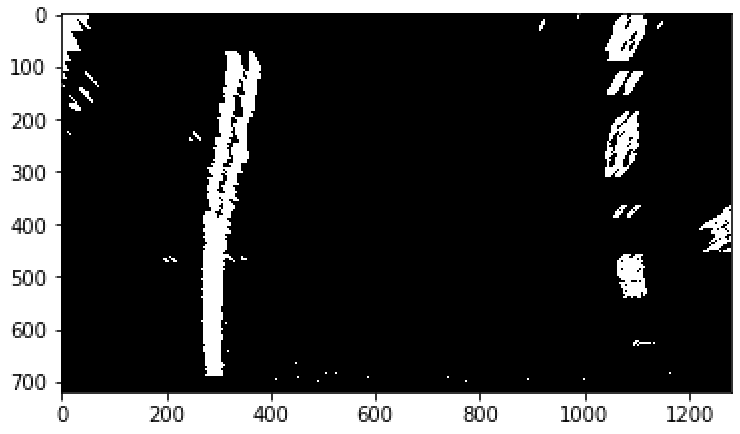
Example: test6.jpg



3) Perspective transform:

The function in the code is perspective(). Firstly, the source and destination points are chosen and the transformation matrix is obtained by using cv2.getPerspectiveTransform(). Then, this matrix is used in cv2.warpPerspective() to obtain the bird's eye view of the image.

Example: test6.jpg

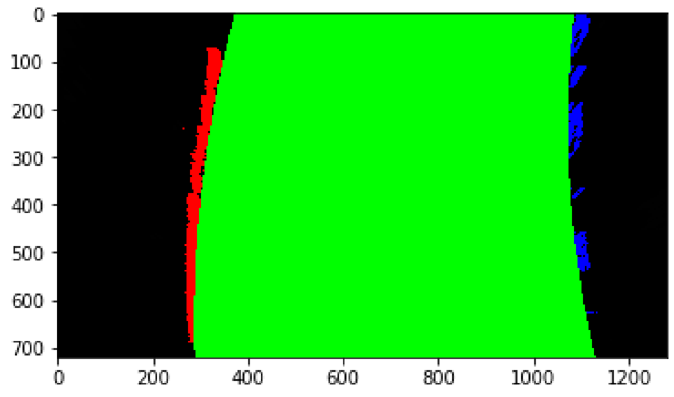


4) Identifying lane lines and fitting a polynomial:

The function in the code is fit_polynomial(). Initially the function find_lane_pixels() is called by this function where first the histogram of the perspective image is computed and the significant peaks are identified.

From those peaks the x-coordinates of the lane lines at the bottom of the image are found. Using them, the sliding window search to find all lane pixels is performed. After when the lane pixels are identified, they are used to fit a second order polynomial.

Example: test6.jpg



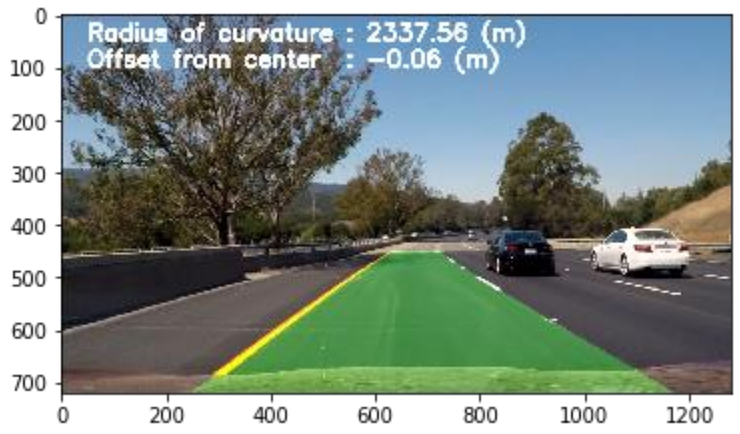
5) Radius of curvature and offset from center:

These parameters are also calculated in the function fit_polynomial(). The radius of curvature is computed by using the formula from the lessons which will make use of the identified polynomial.

Next, the x-coordinates of the lane lines at the bottom of the frame are computed from the polynomial. The center of the lane is found by averaging the x-coordinates of both left and right lane lines. Then the center of the car is assumed to be at the center of the frame and the difference is the offset from center. All pixel values are converted to meters.

6) Marking lane lines in the actual image:

By using inverse of perspective in the function perspective_inv(), the above marked image is bought back to the original image is as follows.


Radius of curvature : 2337.56 (m)
Offset from center : -0.06 (m)
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

Once the lane pixels are found, the algorithm tries to find lane pixels in the next frame by looking +/- margin from the marked lane lines.

7) Project video

The video project_video.mp4 is processed and lane lines are marked. The final output video can be found in output_images folder named as project_video.mp4.

SHORTCOMINGS

This pipeline is likely to fail when there are lines on the road similar to the lane lines. This makes it hard to threshold the lane lines only.

A possible solution for this is by storing lane data for a certain number of frames and using it smartly like to perform the sanity check as mentioned in the tips for the project.

Unfortunately, I am unable to spend extra time on this project's challenge videos. But I am very excited to make my method more robust whenever I can in the future.