Self-Driving Car Nanodegree

P4 – Advanced Lane Finding

Raymond Ngiam
November 16th, 2017

I. Definition

Project Overview

Lane line detection system is a crucial element in an advanced driver-assistance system (ADAS) or a self driving car. Together with a calibrated camera system, a good lane line detection system serves to provide position information of the vehicle relative to the detected lane. Furthermore, it also provides curvature radius of the upcoming lane segment. The information provided help the autonomous steering control system to keep the vehicle on the center of the lane and also anticipate upcoming turns. Hence, a good lane line detection system is indispensable for modern ADAS and self driving cars.

Problem Statement

The goal of this project is to create a software pipeline to identify the lane boundaries from video stream of a front-facing camera on a car. The tasks in this project include:

1. Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
2. Apply a distortion correction to raw images.
3. Create a thresholded binary image for lane segments.
4. Apply a perspective transform to rectify binary image ("birds-eye view").
5. Detect lane pixels and fit to find the lane boundary.
6. Determine the curvature of the lane and vehicle position with respect to center.
7. Warp the detected lane boundaries back onto the original image.
8. Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

II. Background

Details on the input video stream

The input video stream, ‘project_video.mp4’ is captured from a front-facing camera on a car. The camera is mounted at the center of the car, and the frame size of the video is 720x1280 pixels. An example snapshot of the video stream is as shown in Figure 1 below:

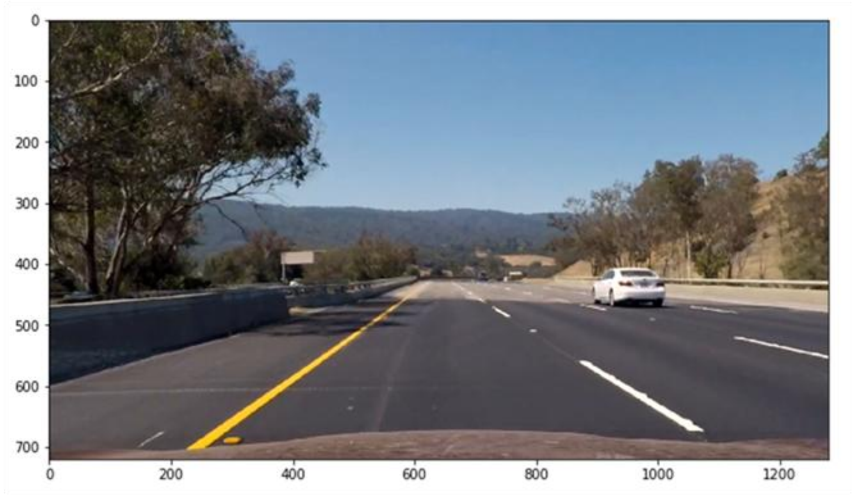


Figure 1 – One static frame from the video stream.

The video is captured in Junipero Serra Freeway near Mountain View, California. Figure 2 below shows a circle drawn on Google Map to coincide with the first left curve in the video. The radius of the first curve is approximately 1 km (bear in mind this is a very rough estimate though).

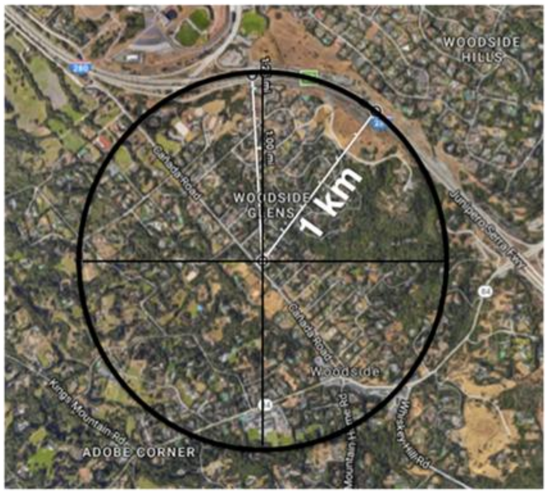


Figure 2 – Google map display of the first left curve in the video.

The lane section as highlighted in the red region of interest (ROI) in Figure 3 covers approximately lane area of width 3.7 m and length 30 m.

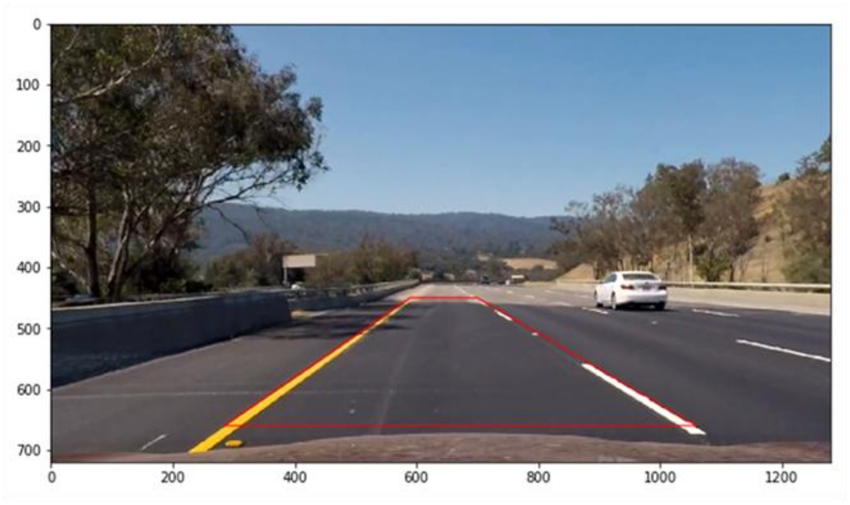


Figure 3 – Approximation of lane width and height. Width of the lane as enclosed by the red ROI is approximately 3.7 m, while the length is approximately 30 m.

III. Methodology

Camera Calibration

Polynomial camera model in OpenCV is used for the calibration of the camera. The polynomial model uses three parameters (k_1, k_2, k_3) to model radial distortions, and two parameters (p_1, p_2) to model the tangential distortions.

$$x_{corrected} = x(1 + k_1r^2 + k_2r^4 + k_3r^6) + (2p_1xy + p_2(r^2 + 2x^2))$$

$$y_{corrected} = y(1 + k_1r^2 + k_2r^4 + k_3r^6) + (p_1(r^2 + 2y^2) + 2p_2xy)$$

where $(x, y) = coordinate\ of\ an\ old\ pixel\ in\ the\ input\ image$

$$r = \sqrt{x^2 + y^2}$$

After distortion correction, the camera model is defined with a projection matrix as shown below:

$$w\begin{bmatrix} x_{corrected} \\ y_{corrected} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

where $w = scale\ factor\ under\ perspective\ transformation$

$f_x, f_y = camera\ focal\ length\ in\ pixel\ coordinates$

$c_x, c_y = optical\ center\ in\ pixel\ coordinates$

$(X, Y, Z)^T = world\ coordinate\ of\ a\ point$

A 9x6 checkerboard calibration target is used for the calibration of the camera. Calibration images are captured with the target in different poses relative to the camera. 20 calibration images in total are captured and are as shown in Figure 4 below:

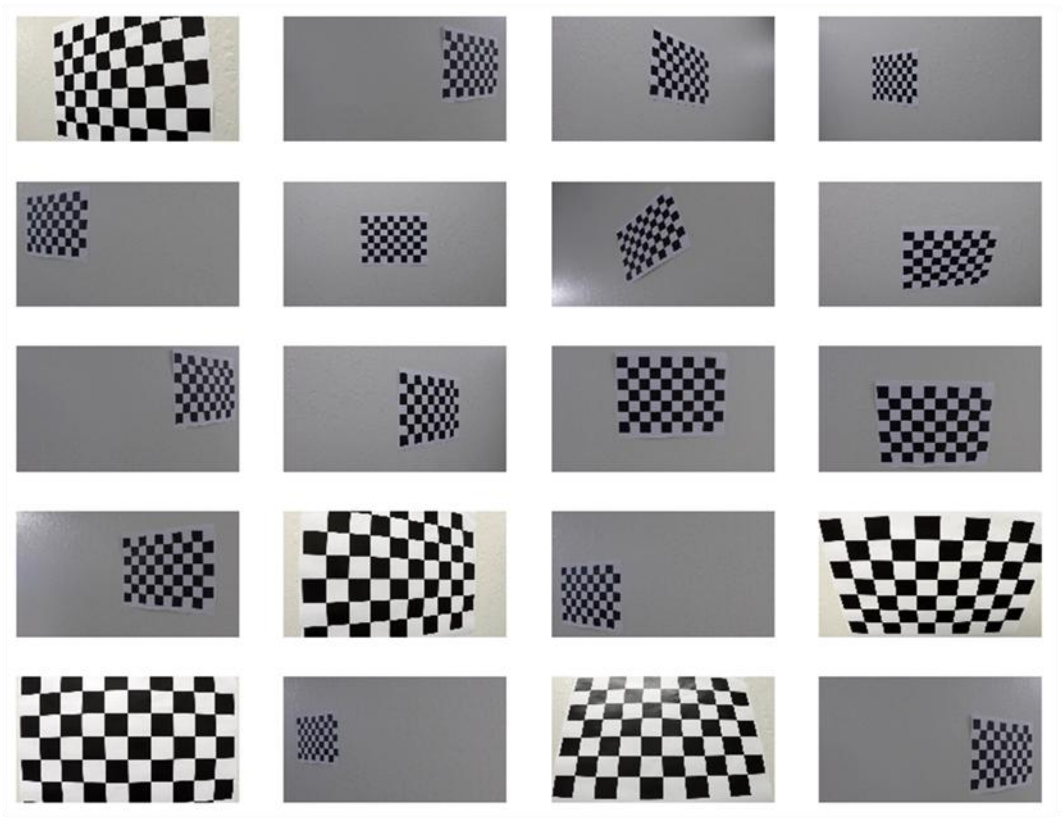


Figure 4 – Calibration images.

'findChessboardCorners' function in OpenCV is used to detect the calibration points. An example of the calibration point detection is as shown in Figure 5.

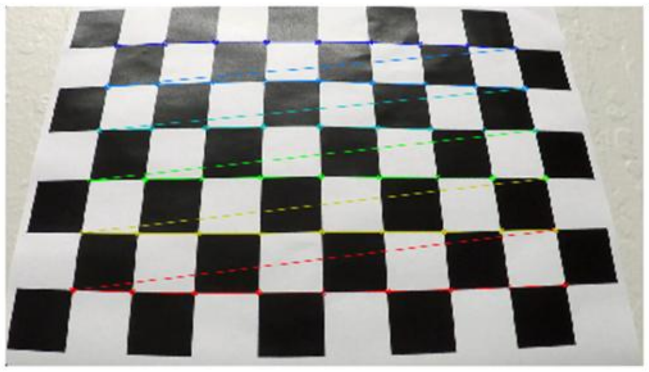


Figure 5 – Detected calibration points from 'findChessboardCorners' function.

OpenCV 'calibrateCamera' function is then used to perform the calibration. Distortions coefficients, camera projection matrix, and poses of the calibration target for each image are returned from this function.

The distortions coefficients and camera projection matrix are useful in removing distortions in the image using the OpenCV 'undistort' function. Figure 6 below shows an example of distortion removal based on one of the calibration images.

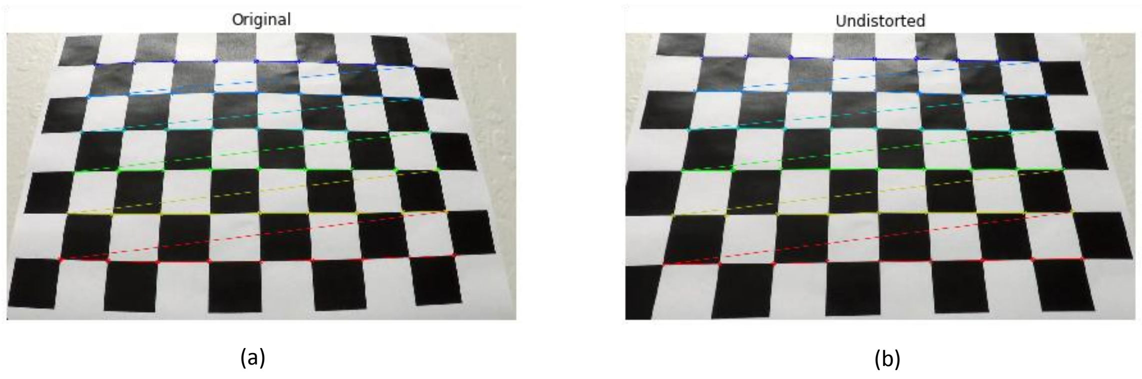
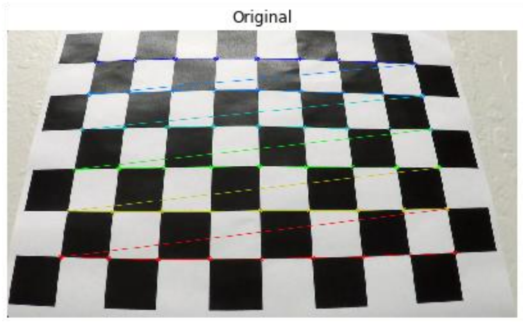
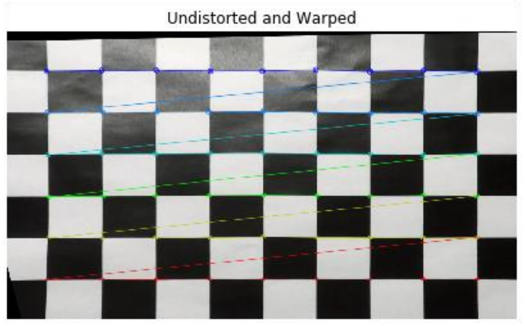


Figure 6 – (a) Original calibration image. (b) Undistorted calibration image.

The undistorted image can be rectified into 'bird-eye' view perspective by using OpenCV 'getPerspectiveTransform' and 'warpPerspective' function. Figure 7 below shows an example of perspective warping based on one of the calibration images.



(a)



(b)

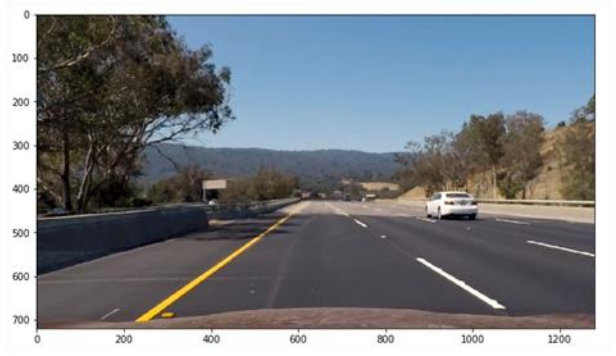
Figure 7 – (a) Original calibration image. (b) Undistorted and warped calibration image.

Lane Detection Pipeline

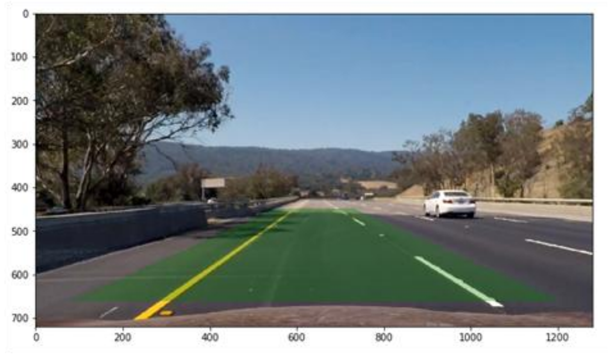
For the sake of explanation, we will be using one static frame from the video to step through the lane detection pipeline.

Distortion removal

First and foremost, camera distortion in the raw frame is removed based on the distortion coefficients and camera projection matrix. Figure 8 below shows the undistorted static frame and a ROI enclosing the lane lines.



(a)



(b)

Figure 8 – (a) Undistorted image. (b) Undistorted image with ROI.

Image cropping

The image is cropped based on the bounding box of the ROI in order to reduce processing time of subsequent operations. Figure 9 shows the cropping output.

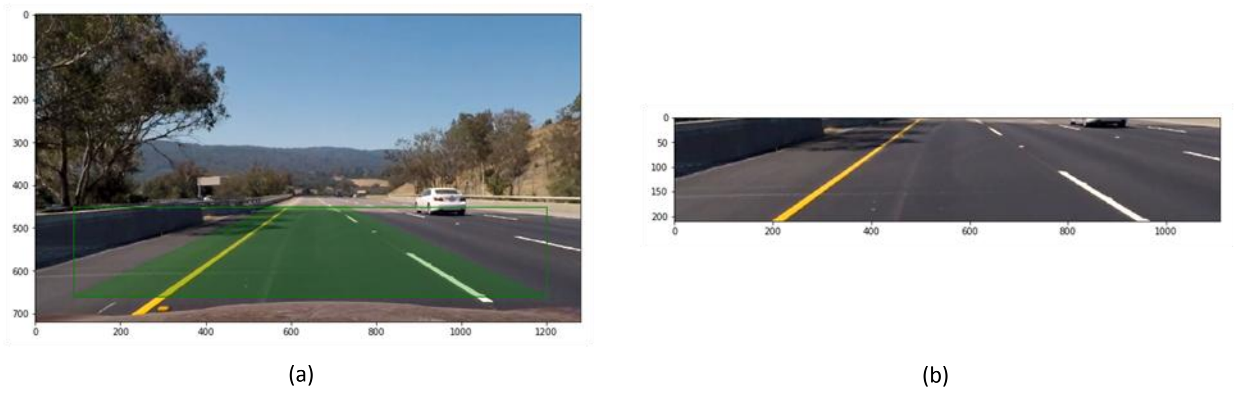


Figure 9 – (a) Undistorted image with ROI. (b) Image cropped based on bounding box of the ROI.

Segmentation

The next step is to perform segmentation for the lane lines. Thresholding in absolute x-axis edge gradient and intensities of HLS color space are used for robust segmentation of the lane lines. The threshold limit for absolute x-axis edge gradient is set to be [30,255]. On the other hand, threshold limits in HLS color space are defined to be [0,50] for the Hue channel, and [80,255] for the Saturation channel.

Segmentation output of both methods are overlaid together to form a single binary image. Figure 10(a) shows segmentation output of both methods in different colors, i.e. green and blue for thresholding in absolute x-axis edge gradient and HLS color space respectively. Figure 10(b) shows the binary image from the union of the two segmentation outputs.

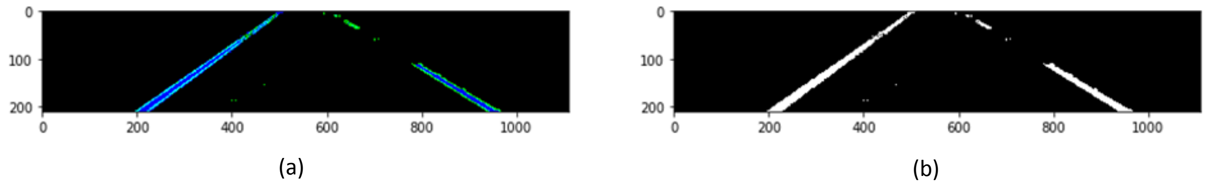


Figure 10 – (a) Segmentation output of absolute x-axis edge gradient (shown in green) and HLS color space (shown in blue). (b) Binary segmentation result from the union of the two segmentation outputs.

The binary segmentation output is then restored onto a blank image with the original frame size (720x1280 pixels). At the same time, image coordinate of the midpoint at the bottom row of the ROI, which is approximately the vehicle center point, is also calculated in the full frame image. The output of this phase is as shown in Figure 11.

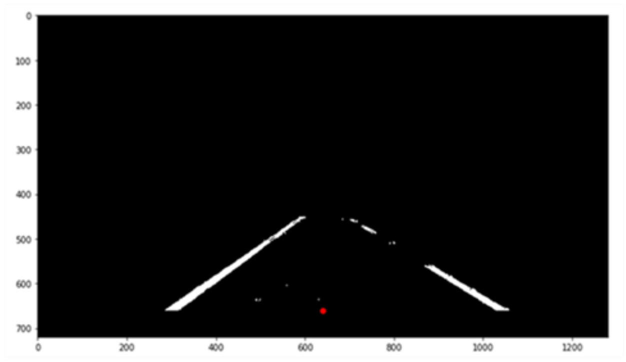


Figure 11 – Restore the binary segmentation output to original frame size. Red dot indicates the approximate vehicle center point.

Perspective warping

Next, perspective transform is applied to rectify the image into bird's-eye view that lets us view the lane from above. Source points for the perspective transform are defined such that there are four vertices enclosing a straight lane. On the other hand, destination points for the perspective transform are defined such that the rectified lane is on the center of the rectified image with some buffer spaces horizontally.

The size of the warped image is defined as 640x360 pixels, considering the length of the predefined lane section is longer than the width. In addition, this also serves to down sample the image resolution for faster processing.

The approximated car center point in the full frame image computed previously is also transformed under the same perspective transformation onto the warped image.

The outcome of the image warping phase of the pipeline is as illustrated in Figure 12.

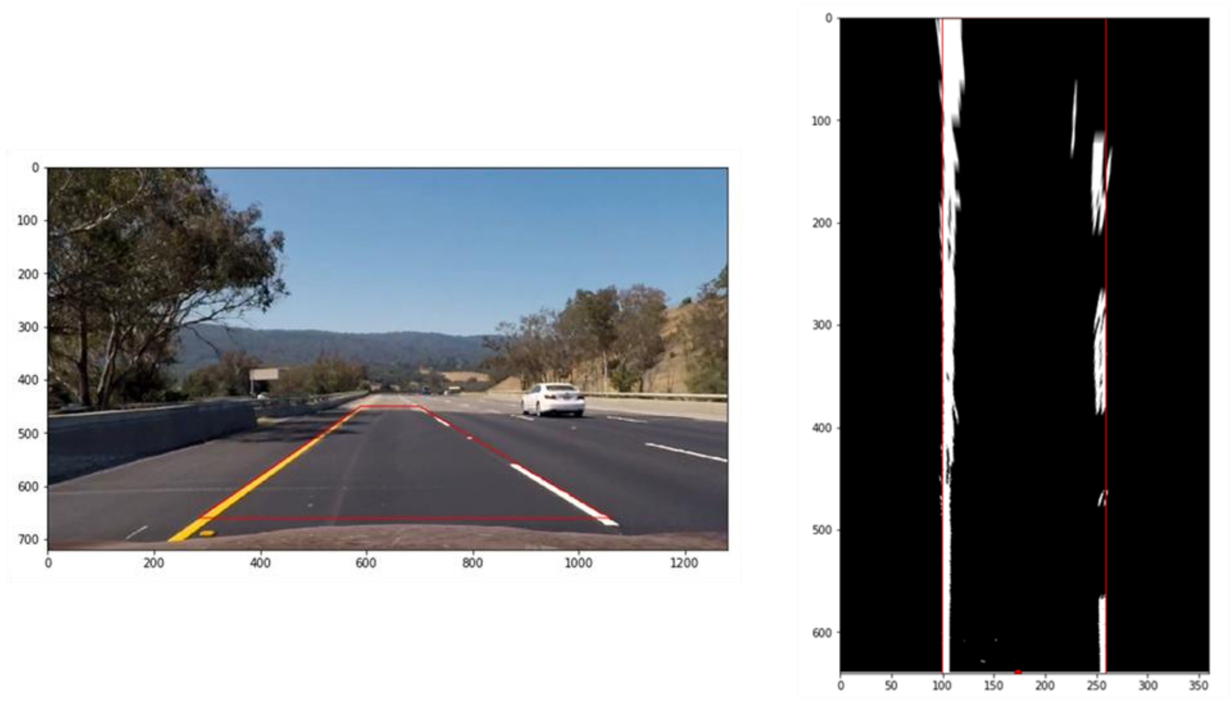


Figure 12 – (a) Undistorted image, source points for the perspective transform are the 4 vertices of the ROI in red. (b) Warped image, destination points for the perspective transform are the 4 vertices of the ROI in red, the vehicle center point is also transformed into the warped image and is shown in red dot at the bottom of the image.

<u>Segmentation enhancement</u>

Next, image morphology is applied onto the warped binary image to enhance the segmentation. Since we were using absolute x-axis edge gradient for the segmentation, there are chances that we obtain two vertical edges instead of one full rectangular block for a lane line segment. Hence the image morphology operation we use here is a closing operation with a 7x7 pixel kernel. This serves to join the gap between the detected edges. Figure 13 shows the effect of closing morphology operation.

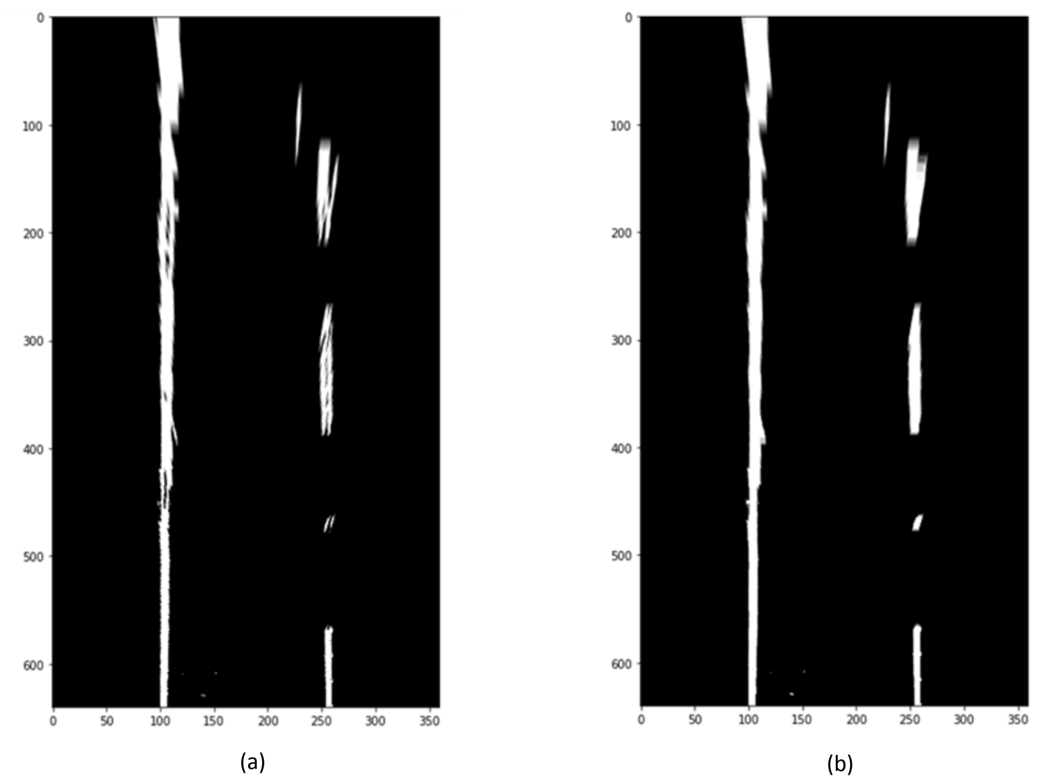


Figure 13 – (a) Warped binary segmentation image. (b) 7x7 closing morphology operation on the warped binary image.

Contour processing

This is then followed by contour processing operations which first convert individual pixel blobs into contours, and then perform filtering based on properties of the contours. Two particularly essential contour properties that we used are elongation and minor length.

Elongation of a contour is defined as the ratio between the major axis length and the minor axis length of the contour's fitted ellipse. As its name implies, it measures how 'elongated' a contour is. The higher this number is, the more 'elongated' is the contour.

On the other hand, minor length of a contour is defined as the shortest length (in pixel) of the contour's fitted rotated rectangle.

The filtering criteria we used are in the following order: (1) area > 40 pixels, (2) elongation > 2.0, (3) minor length > 4 pixels. The outcome of the contour processing based on the previous image is as shown in Figure 14.

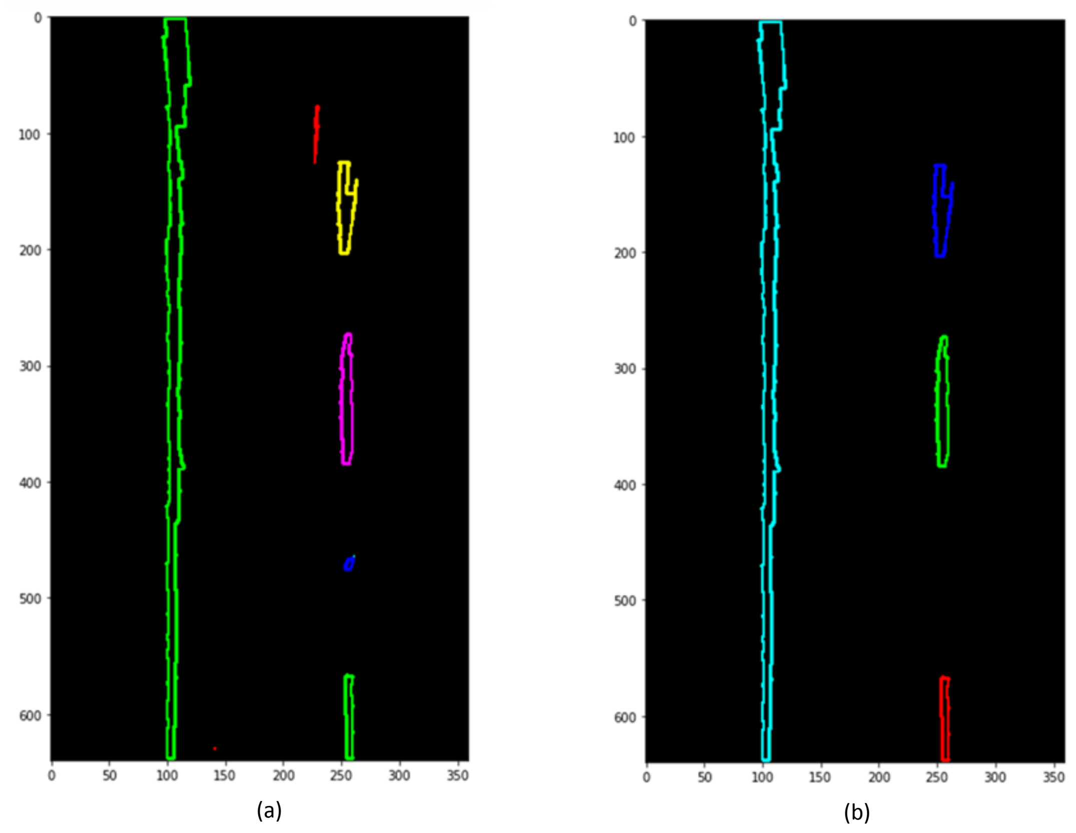
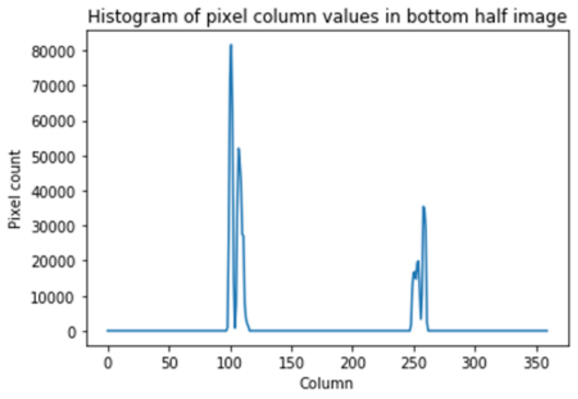


Figure 14 – (a) Contour output of the enhanced binary segmentation. (b) Filtered contours.

Lane curve fitting

Up until here, we still need to decide explicitly which contour pixels belong to the left lane and which belong to the right lane.

A histogram is computed along all the columns in the lower half of the image. However, due to the contours are hollow in the middle, there are discontinuities in the histogram that correspond to hollow part of the contours. 1D smoothing of kernel width 30 is applied to the histogram to smooth out these discontinuities. The peaks in the left and right halves of the smooth histogram are good starting estimates for the left and right lane lines. The outcomes of the histogram computation are as shown in Figure 15.



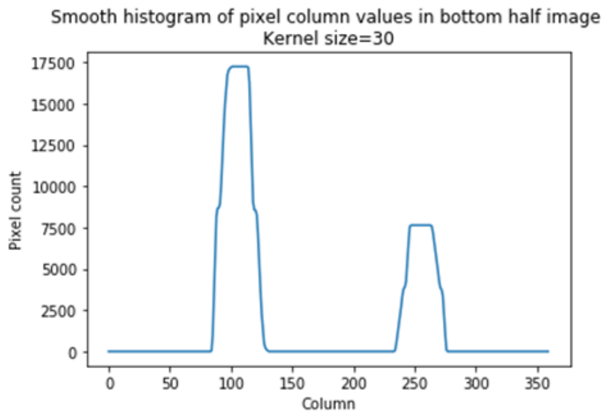


Figure 15 – Histograms for starting column estimations of left and right lanes.

The warped contour image is partitioned into 9 equal horizontal slices. Two sliding windows of width 101 pixels are centered on the estimated starting points for left and right lanes respectively for the bottommost slice. Any contour pixel that falls within the sliding windows will be regarded as pixels of the respective lane lines.

For each sliding window, average column value of the selected pixels is computed and used as the column value to place the sliding window on the next horizontal slice above. This process iterates over the next 8 horizontal slices.

If the detected contour pixels for both lanes exceed a predefined size, 20 pixels in this case, two second order polynomial curves are fitted from the selected contour pixels of left and right lane respectively.

$$x = f(y) = Ay^2 + By + C$$

We are fitting for $f(y)$, rather than $f(x)$, because the lane lines in the warped image are near vertical and may have the same x value for more than one y value. The fitting is done with ordinary least squares using Numpy's 'polyfit' function. The outcome is as shown in Figure 16(a).

Once we know the fitted lane curve in one frame, we do not need to perform the blind iterative search again for the subsequent frame. Instead we can just search in a margin (e.g. 50 pixels) around the previous line position as shown in Figure 16(b).

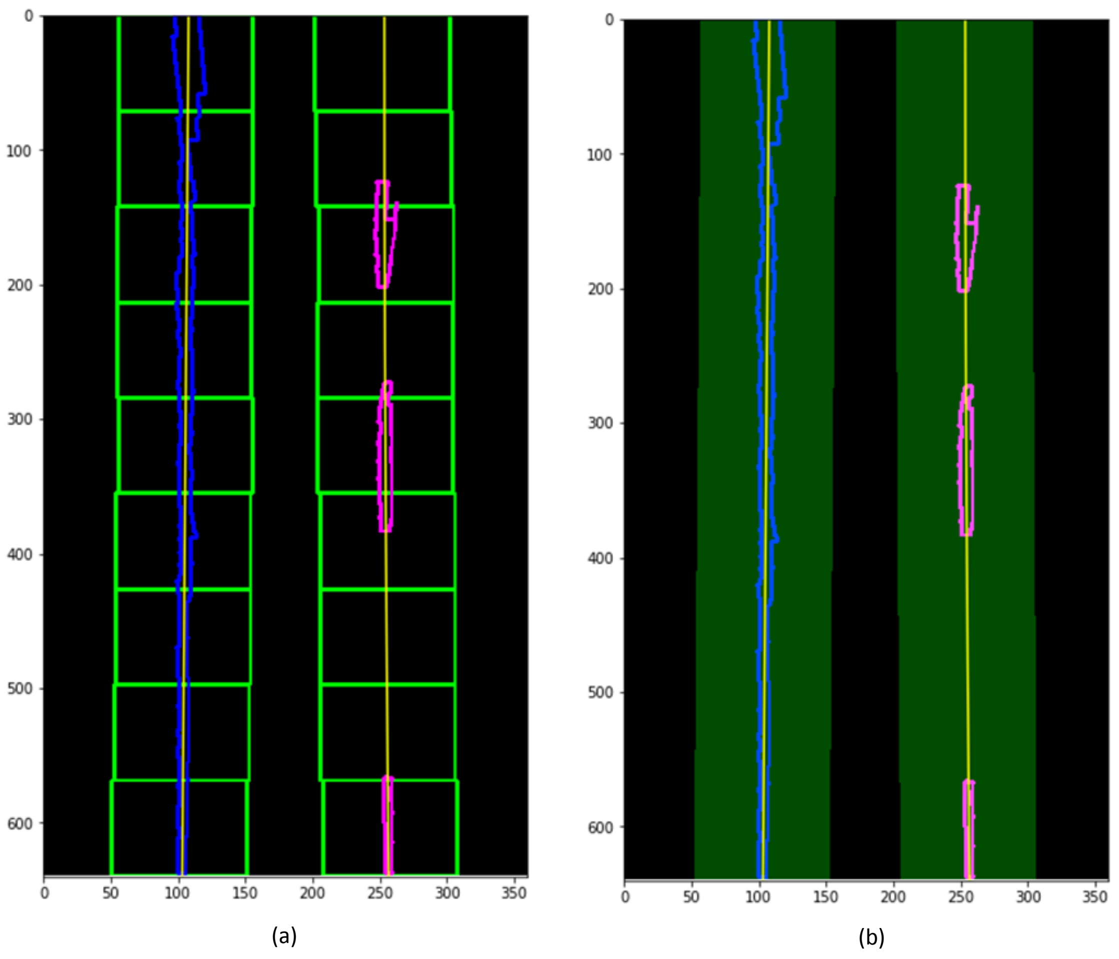


Figure 16 – (a) Blind search for selecting contour pixels for left and right lanes. (b) If left and right lane lines are detected in previous frame, ROIs for selection of contour pixels are just ±50 pixel from the previous fitted lines.

Lane curve smoothing

The lane detection pipeline up until here works considerably well. However, there might be abrupt changes in the line detections from frame to frame. It would be preferable to smooth over the last n frames of video to obtain a cleaner result. In our case, we used the last 5 frames for smoothing of the lane lines.

The fitted curves of the last 5 frames are stored using a customized Line object class. These fitted curve points are concatenated and then used to fit a new second order polynomial curve, which serve as a smoothed version of the curves over last 5 frames.

One full curve in a single frame contains 640 points as we generated the curve along entire y-axis of the warped image using the fitted polynomial parameters. To speed up the fitting process, we sub sample the curve points by approximately 25%. This is done by skipping 25 points between each sampling of a full curve.

The fitting of polynomial here is done using the approach of iterative reweighted least squares (IRLS) [2][3]. This approach has the effect of mitigating the influence of outliers in least squares fitting by introducing a weight, w_i for each data point. The IRLS fitting starts by assigning equal weights for each data point, and then iteratively refines the weights based on the absolute distance of each point from the fitted curve. The higher the absolute distance of the point from the fitted curve, the lower the weight assigned. Thus, objective of outlier removal is achieved.

Consequently, this serves to improve the robustness of the lane detection. If a new curve is detected with abruptly different shape from the last 4 detected curves, its data points will be treated as outliers and will be taken into account at lower weights.

Lane Related Measurements

In order to obtain meaningful measurements, we need to convert the warped image from pixel coordinates into real world metric coordinates.

Based on the metric measurements of the straight lane section that we are projecting onto the warped image, we can obtain the pixel conversion factors for x axis, s_x and y axis, s_y. The straight lane section is projected onto the warped image to be a rectangle of width 153 pixels and of height 640 pixels. Given the actual measurements of width 3.7 m and height 30 m, s_x and s_y can be computed as follows:

$$s_x = \frac{Metric\ width}{Pixel\ width} = \frac{3.7\ m}{153\ pixel} = 0.024\ m/pixel$$

$$s_y = \frac{Metric\ height}{Pixel\ height} = \frac{30\ m}{640\ pixel} = 0.047\ m/pixel$$

Vehicle offset from lane center

Vehicle offset from lane center can be obtained by computing the distance between the transformed car center point and the lane's midpoint at the same row in the warped image. This pixel distance is then multiplied by the x axis conversion factor, s_x to obtain the actual offset in meters.

Curvature radius of the lane

The previous fitted curves in pixel coordinates are converted to metric coordinates by multiplying with the conversion factors. New second order polynomial curves are then fitted using these converted curve points in meters.

The parameters of these newly fitted curves are then used to determine the lanes' curvature radius in meters, which is given by [1]:

$$R_{curve} = \frac{[1 + f^{'}(y)^2]^{\frac{3}{2}}}{|f^{''}(y)|}$$

$$f(y) = Ay^2 + By + C$$

$$f'(y) = 2Ay + B$$

$$f''(y) = 2A$$

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

The final curvature radius is computed as average of left and right lanes' radius.

Result visualizations

For visualization of the detected lane in the warped image, the two fitted lane curves are concatenated to create a polygon with the color green. It is then overlaid onto the warped image. The vehicle center position and the lane center are visualized as two short vertical lines at the bottom of the warped image as shown in Figure 17(a). Orange line indicates vehicle center position and yellow line indicates lane center.

For final visualization, the green polygon is unwarped using the inverse perspective transformation matrix and then overlaid onto the undistorted video frames. The computed lane curvature radius and lane center offset of the vehicle are displayed on the upper left of the video frames as shown in Figure 17(b).

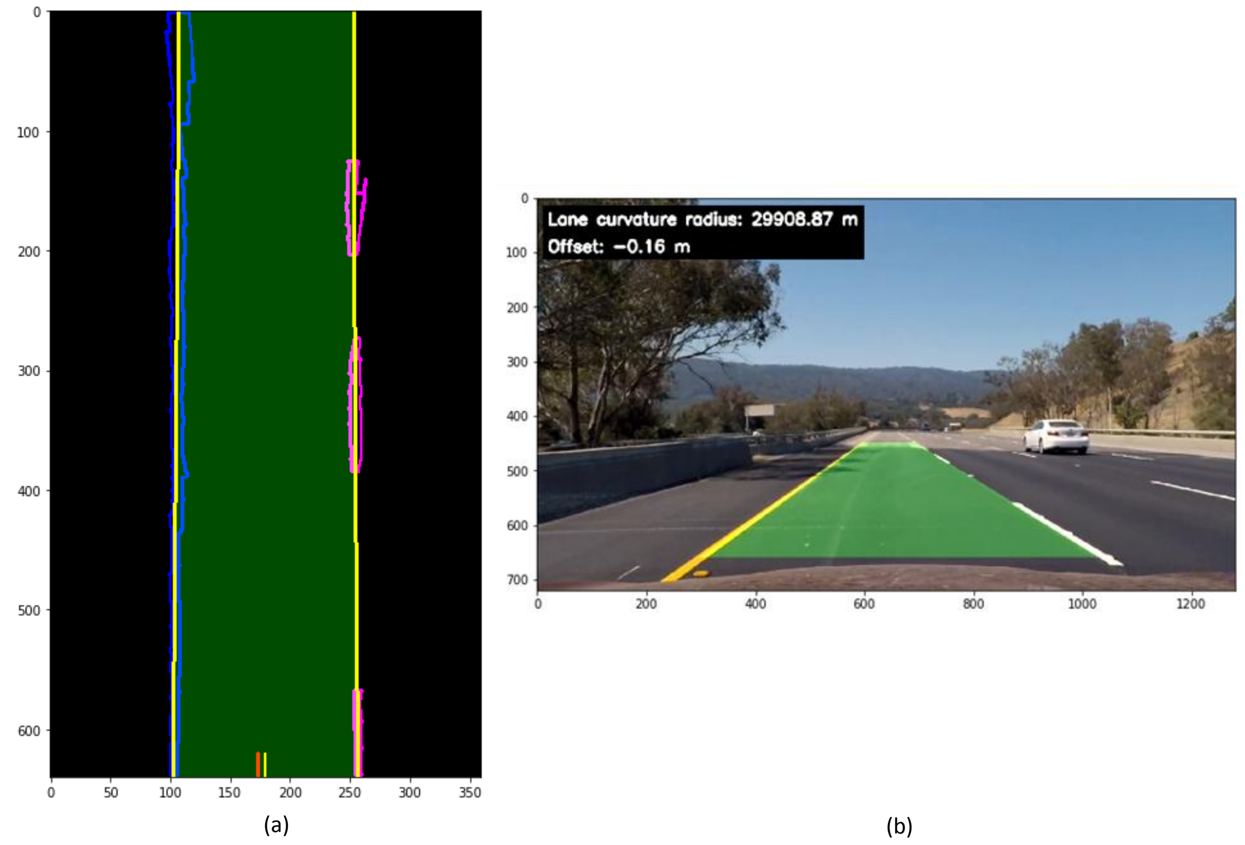


Figure 17 – (a) Detected lane visualization in warped image. (b) Final visualization with lane region augmented onto undistorted image.

IV. Results

Pipeline Evaluation

The lane detection pipeline is tested by running with 'project_video.mp4'. The lane detection is robust and consistent throughout the entire video (See the video files 'project_video_warped.mp4' and 'project_video_final.mp4').

V. Conclusion

Reflection

One critical issue faced during the implementation of the lane detection pipeline is detection failure in one single frame will return abrupt change and faulty result, causing the final video visualization to be jumping occasionally from frame to frame. This is solved by implementing smoothing of fitted lines with fitted results from last 5 frames and using the IRLS approach to remove potential outlier lines. This solution works considerably well to improve the robustness of the lane detection pipeline for the video being tested.

Up to this stage, the pipeline is still susceptible to some potential failures. The pipeline is likely to fail when there are long cracks on the road surface. Since we are using x-axis absolute edge gradient for segmentation of the lane lines, long cracks with high contrast on the road might be misdetected as lane lines and consequently causing the lane lines to be fitted incorrectly.

The pipeline may also fail if ambient illuminations are too dim for a prolonged duration, i.e. more than 5 frames. For instance when there is a dark cloud or shadow casted from a flyover bridge above. Illumination condition that is radically different can cause the segmentation function to fail to capture the correct lane segments. One possible workaround for this is to increase the number of frame used for the IRLS smoothing of the lane lines. Thus, even though the pipeline failed to detect new lane lines, it can still retain the past high confident results.

There are some approaches that might make the pipeline more robust. First, more color spaces like ARgYb, Cielab, Cieluv, YUV, etc., can be explored to find the optimal image channel for the segmentation of the lane lines. Dynamic thresholding approaches can also be tried to see if it can improve segmentation results. Lastly, deep learning approaches can also be used for classification of lane line segments in the undistorted images.

References

[1] Interactive Mathematics – Applications of Differentiation – 8. Radius of Curvature. https://www.intmath.com/applications-differentiation/8-radius-curvature.php (Retrieved 11, Nov 2016).

[2] P. J. Huber – Robust Statistics. John Wiley & Sons, New York (1981).

[3] F. Mosteller, J. W. Tukey – Data Analysis and Regression. Addison-Wesley, Reading, MA (1977).