Advanced Lane Line Finding

Samuel Navarro

May 30, 2019

Contents

1 Camera Calibration **1**

2 Pipeline Images **2**

2.1 Undistorted test 2

2.2 Perspective Transform 2

2.3 Color transforms and gradients 2

2.4 Fitted Lane Lines 3

3 Discussion **3**

1 Camera Calibration

The code for this step is implemented in the third and fourth cell of the jupyter notebook in the functions `show_image` and `calibrate_camera`.

The output of `calibrate_camera` gives us the objpoints and the imgpoints arrays. I then use the this arrays to obtain the distortion coefficients and the camera matrix to obtain the undistorted image.

An example is in the Figure 1 and 2 1

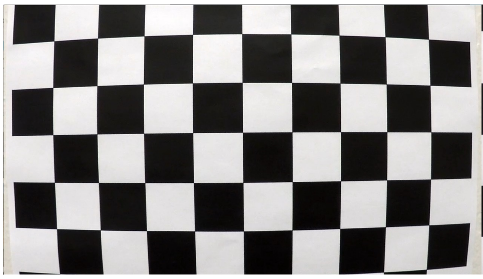

Figure 1: Original

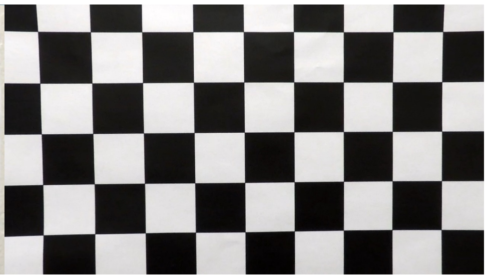

Figure 2: Undistorted

2 Pipeline Images

2.1 Undistorted test

We can se te distortion correction being applied in one of the test images:

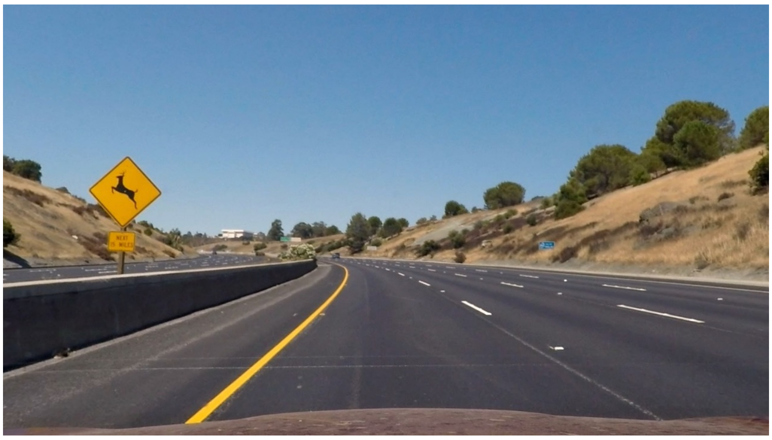

Figure 3: Undistorted test

2.2 Perspective Transform

The perspective transform is in the function `perspective_transform` in the same notebook.

The parameters I used was hardcoded. This is obviously something that can be improved. The most straightforward way I can think of based on what we've learned is applying Hough Transform in the image and apply the perspective transform on that.

The result is in the Figure 4:

2.3 Color transforms and gradients

Then, with the warped image at hand, I used a combination of color and gradient thresholds. The code is in the function `color_pipeline`.

I figure out that when the road is clear, the s_channel is better to find the lanes but when the road is dark because of some shadow caused by the threes, the l_channel was better.

Because of that, I used a combination l and s channel images with and without x gradient.

The result can be found in the Figure 5

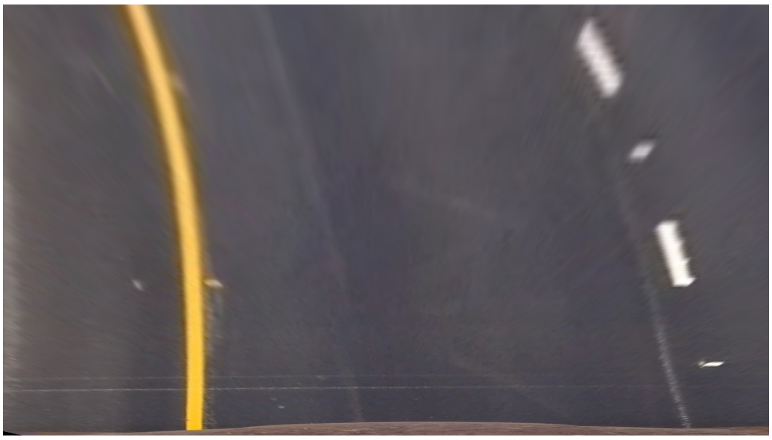

Figure 4: Warped

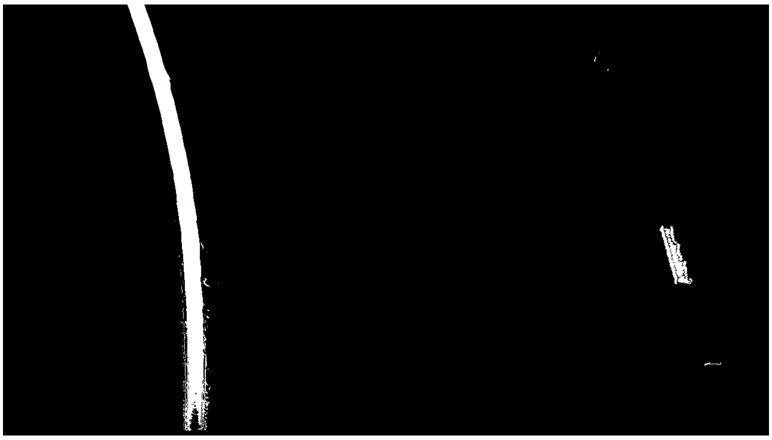

Figure 5: Combined Binary

2.4 Fitted Lane Lines

The functions I used to fit the lines was `search_around_poly`. The result can be found in the Figure 6.

For the video I used the `search_around_poly` but for illustration I used the `old_search_around_poly` function to illustrate the windows.

3 Discussion

- One of the problems I encounter was the fact that you have a difference in the frames when you are trying to diagnose the problems. It seems obvious when you know you are dealing with the undistorted frames but at the beginning I was trying to diagnose the problems with the original frame.

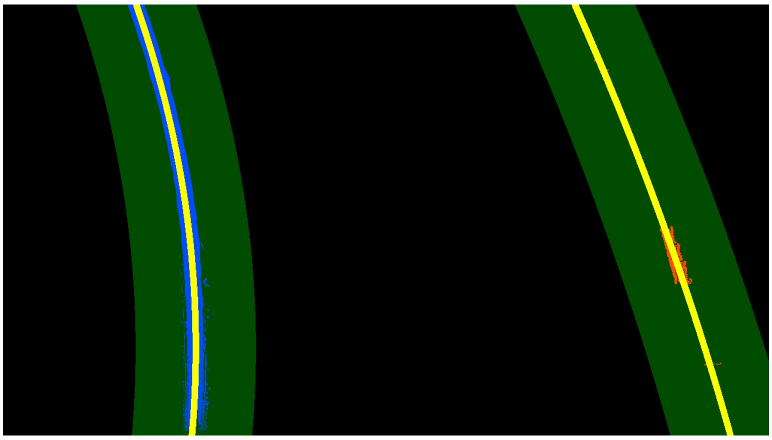

Figure 6: Fitted Lines

One of the things I believed I could make different was to make a moving window to detect the lines. I believed that you can make a mask of the image in places where you got peaks in your histogram.

But, this approach could be more computationally expensive because you have to use ‘cv2.fitPoly‘. The fact that the parameters of ‘cv2.rectangle‘ are just two points makes it easy also to draw the rectangles.