Advanced Lane Finding Project

Report based on the standard template.

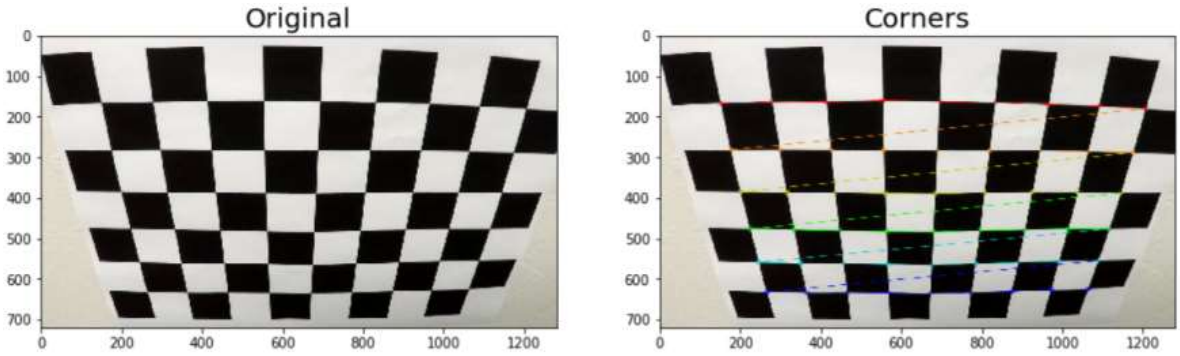
The goals / steps of this project are the following:

- 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.

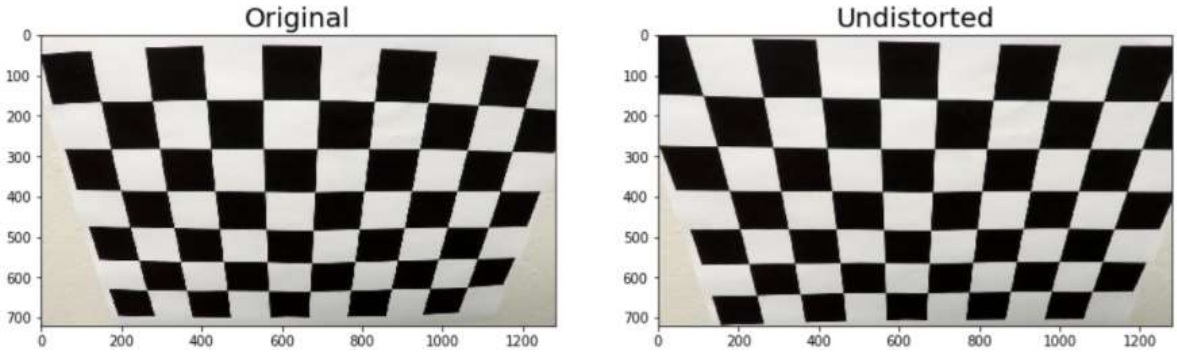
Camera Calibration

I used a chessboard size of 9x6 corners for the project based on the provided images. By means of the OpenCV functions I calculated the correct camera matrix and distortion coefficients.

First, collecting the 2D points and check the result.



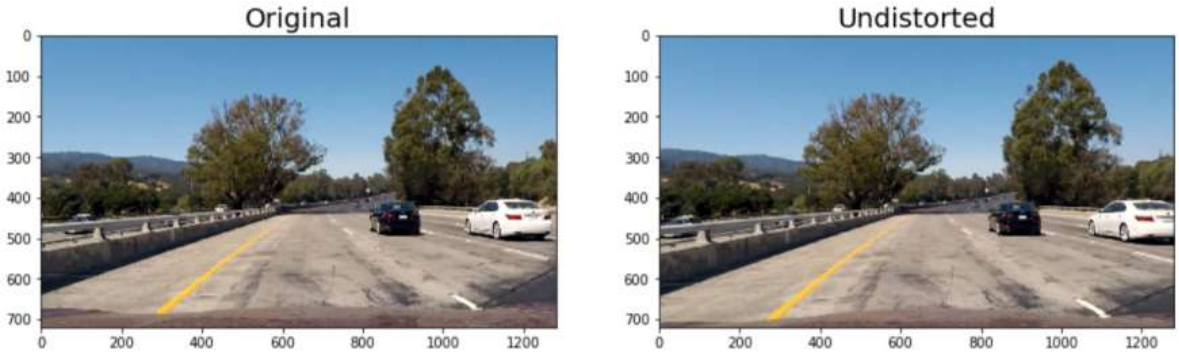
Then, based on the distortion matrix I checked it on the same image.



Pipeline

1. Distortion correction

The previous step is applied throughout the pipeline. For the *test1.jpg* sample, using the pre-computed camera distortion matrices got the following result:

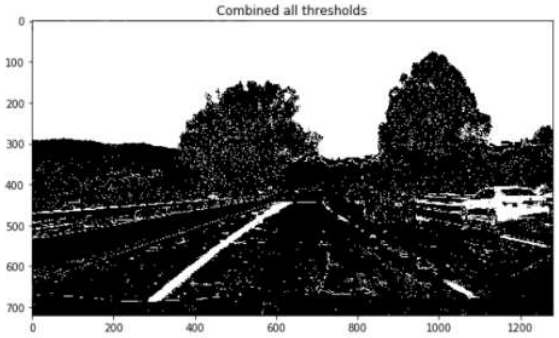


2. Transforms

I create a binary image combining the following thresholds:

- Convert to HLS color space and separate the S channel range (80, 255)
- Convert to HSV color space and separate the V channel range (165,255)
- Sobel Threshold x gradient and separate range (20, 150)
- Sobel Threshold y gradient and separate range (20, 255)
- absolute value of the gradient direction and separate range (0.9, 1.1)

The result:

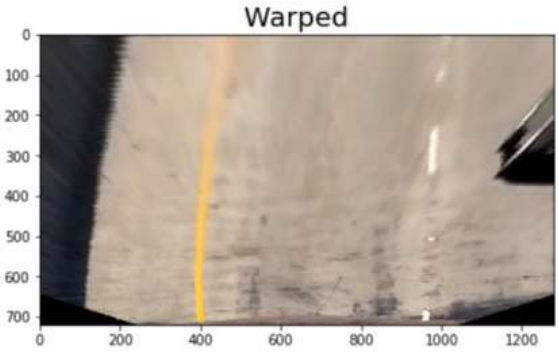


Initially only HLS and Sobel X gradients were used. As it got bad results on the second and third video; in an incremental way I introduced a HSV and Sobel Y gradient filter.

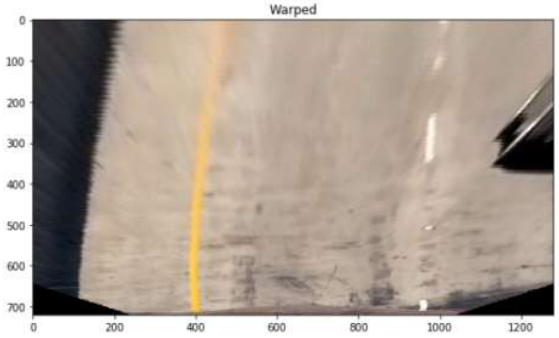
The threshold function was improved too with the ideas from this thread.

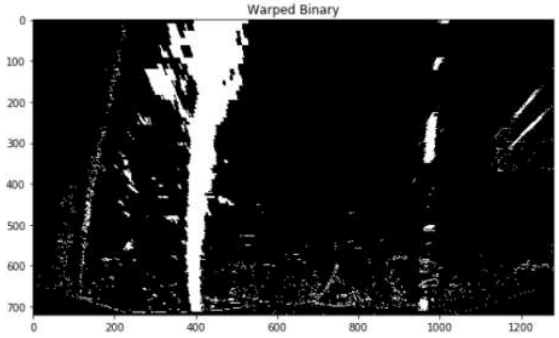
3. Perspective transform

The selection of the perspective transform source and destination points were using ideas from this thread and this one. The result is the following:



Applying the binary threshold to the warped image:



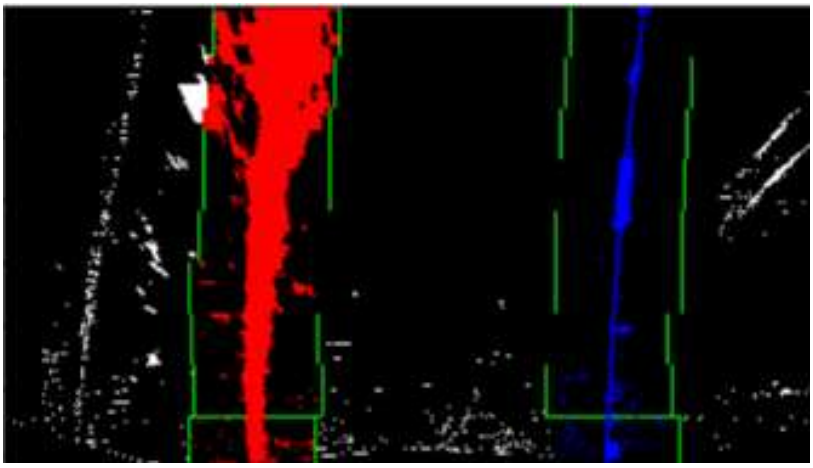


4. Lane finding

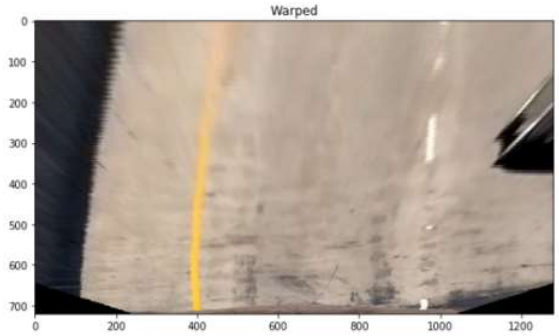
I detect the lines from a warped image using the implementation from the course.

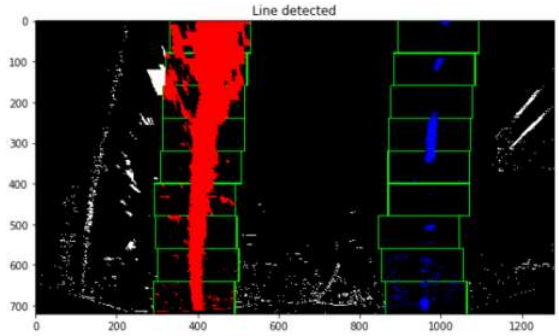
Based on the binary warped image, I get a histogram and find the peaks.

Using the sliding windows technique (9 values), I apply a rectangle on the lines detected and then fit a second order polynomial to each "lane".



Applied to the warped image:





5. Curvature

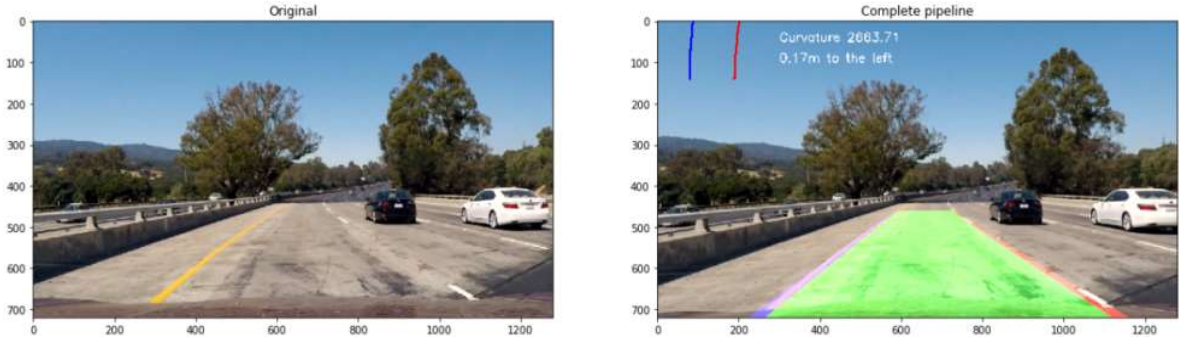
To determine the curvature of the lane I applied the radius of curvature formula

$$\text{Radius of curvature} = \frac{\left[1 + \left(\frac{dy}{dx}\right)^2\right]^{3/2}}{\left|\frac{d^2y}{dx^2}\right|}$$

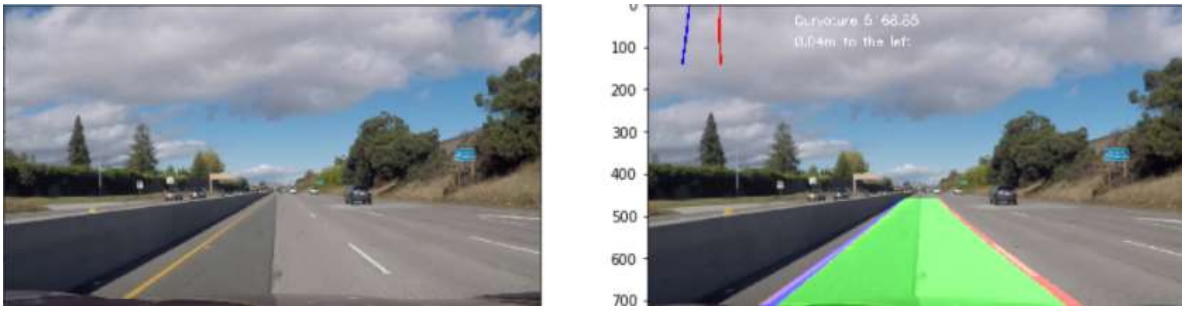
To calculate the offset of the car on the road, first I evaluate the second order polynomial on the first item to get right and left positions and get where the center is. Then compare it with the middle of the binary warped image.

As these results are in pixels; I assume a lane is 3.7m wide (US regulations) for the conversion in the X direction; and that 500 pixels in the Y direction means 30m ahead of the car. The following thread has got useful tips.

The final result applied to the selected sample image:



The same pipeline applied to an image from the second video (challenge_video.mp4):



Pipeline

1. Video

The image processing pipeline applied to the first image worked fine. But, when the pipeline was applied to the second and third videos, in some cases the detection was wrong; several frames were out of scope.

2. Sanity Check

I applied some ideas discussed in the forum.

Store the coefficient of the right and left lanes and compare it with the current frame. If the error is greater than some threshold, it means that the frame is bad.

Initially I tested to average current frame with the previous one (thread) and got some improvements.

Later I kept the latest 10 frames, once a sanity check function is triggered, I used a weighted average with the latest frames with a weight of 0.7 (thread), it got better and is noticeable in the second video.

The threshold value was assigned a value of 200; I got started testing from 100 pixels.

Anyway, the detection is not good on the third video (harder_challenge_video.mp4). The lines are correctly mapped but in several changes it fails (shadows, pavement color changes, strong curves, only one line present)

Considerations

The binary image was improved in several steps. Initially testing on the first video, then second and third.

The third video still has problems with different image conditions so there is a lot of room for improvement on the threshold function. It could be better combined with different threshold or a wider combination.

It would be nice to get a few samples from the second and third video so it is better to try the threshold pipeline in an isolated way as it takes time to process videos.

There is no high confidence about the detection; there is an improvement on sanity check to discard wrong frames.