A4: Advanced Lane Detection Writeup

Advanced Lane Finding Project

The goals / steps of this project are the following:

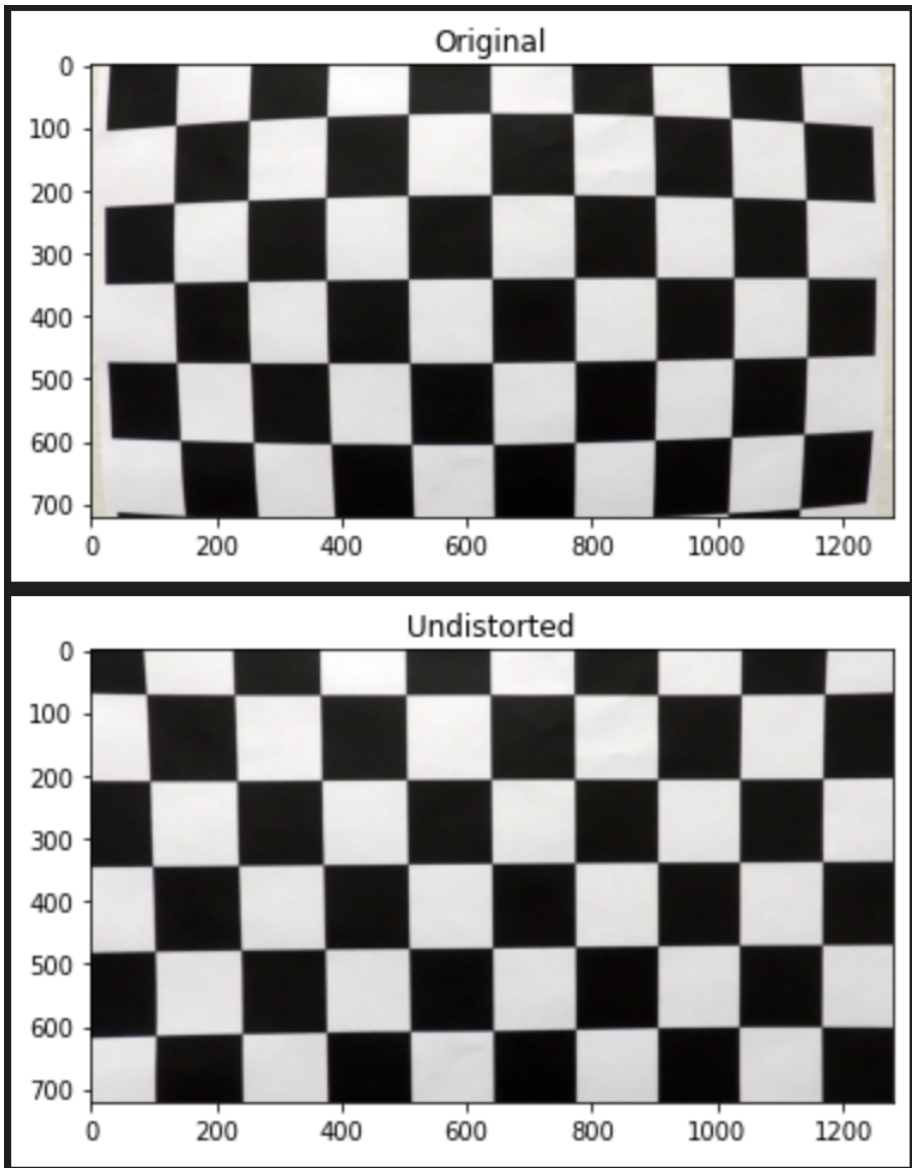
- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- 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.

Rubric Points

Camera Calibration

1. Have the camera matrix and distortion coefficients been computed correctly and checked on one of the calibration images as a test?

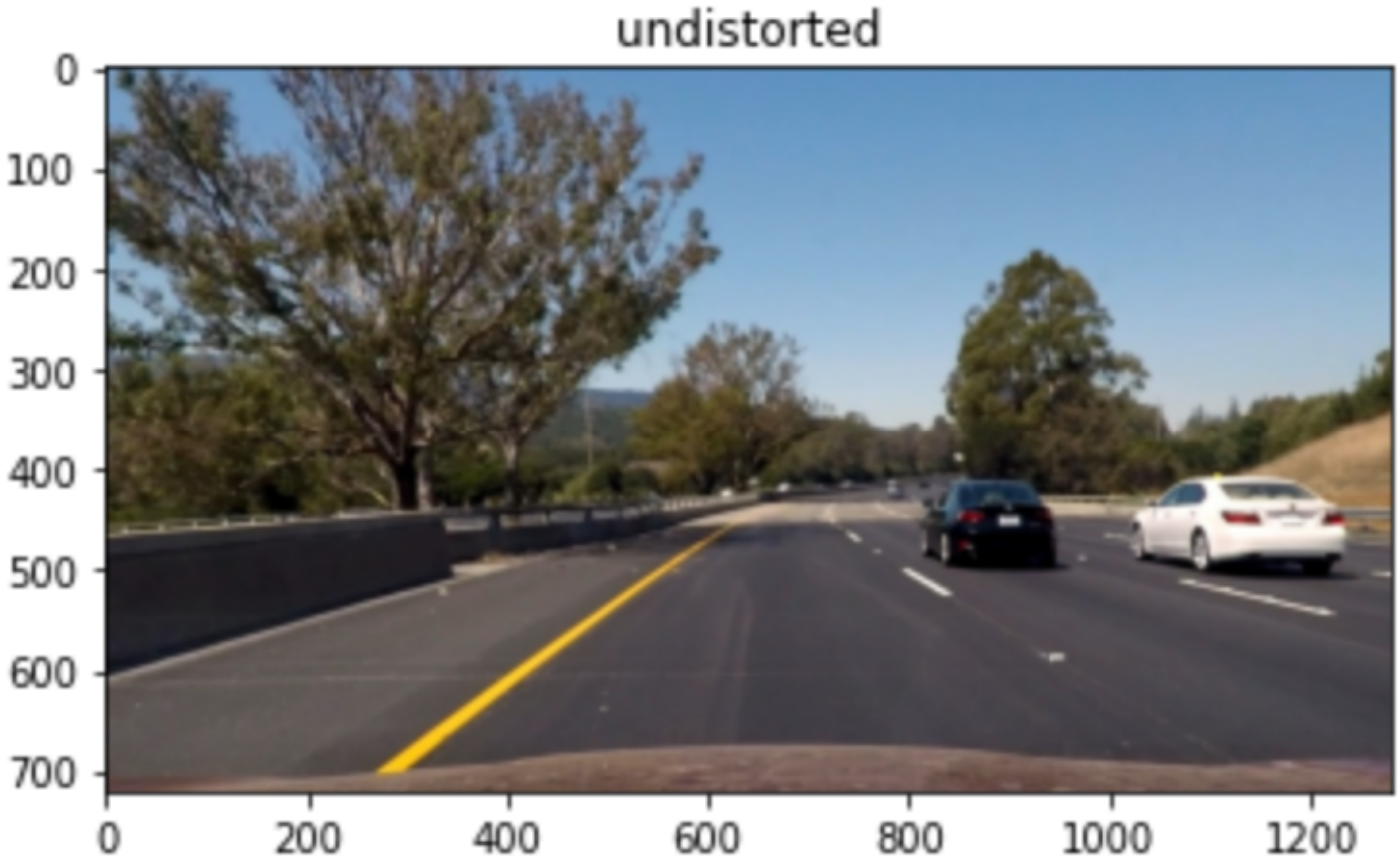
Yes, I iterated through the chessboard images and calibrated the camera matrix by finding the corners on the boards and storing both the object points and image points of said corners. Through the use of the cv.calibrateCamera() function, I was able to retrieve the camera matrix and distortion coefficients. I then used the cv.undistort() function on each image to undistort them:



Pipeline (single images)

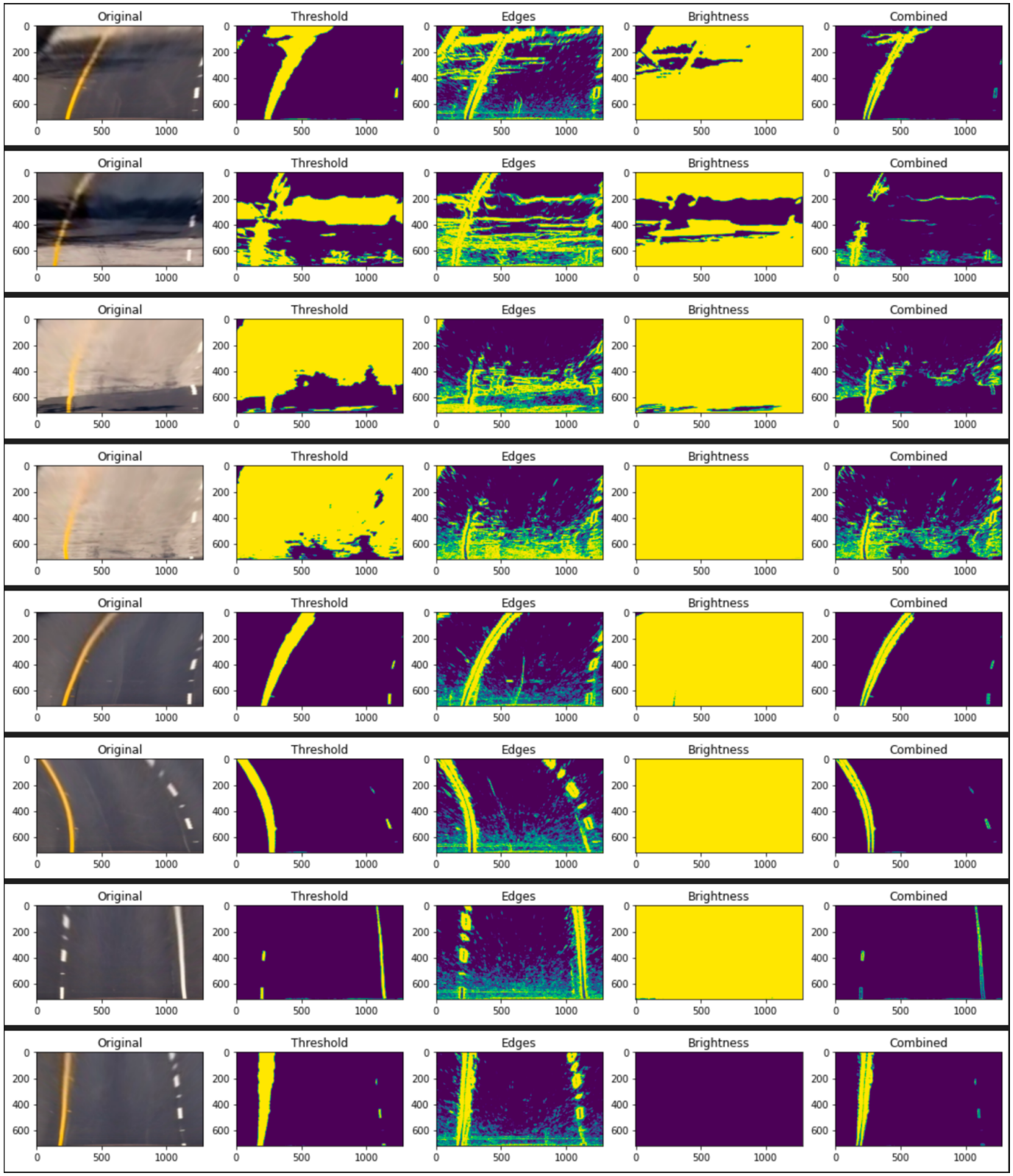
1. Has the distortion correction been correctly applied to each image?

Yes, I used the previously calculated camera matrix and distortion coefficients to easily undistort each road image using the cv.undistort() function.



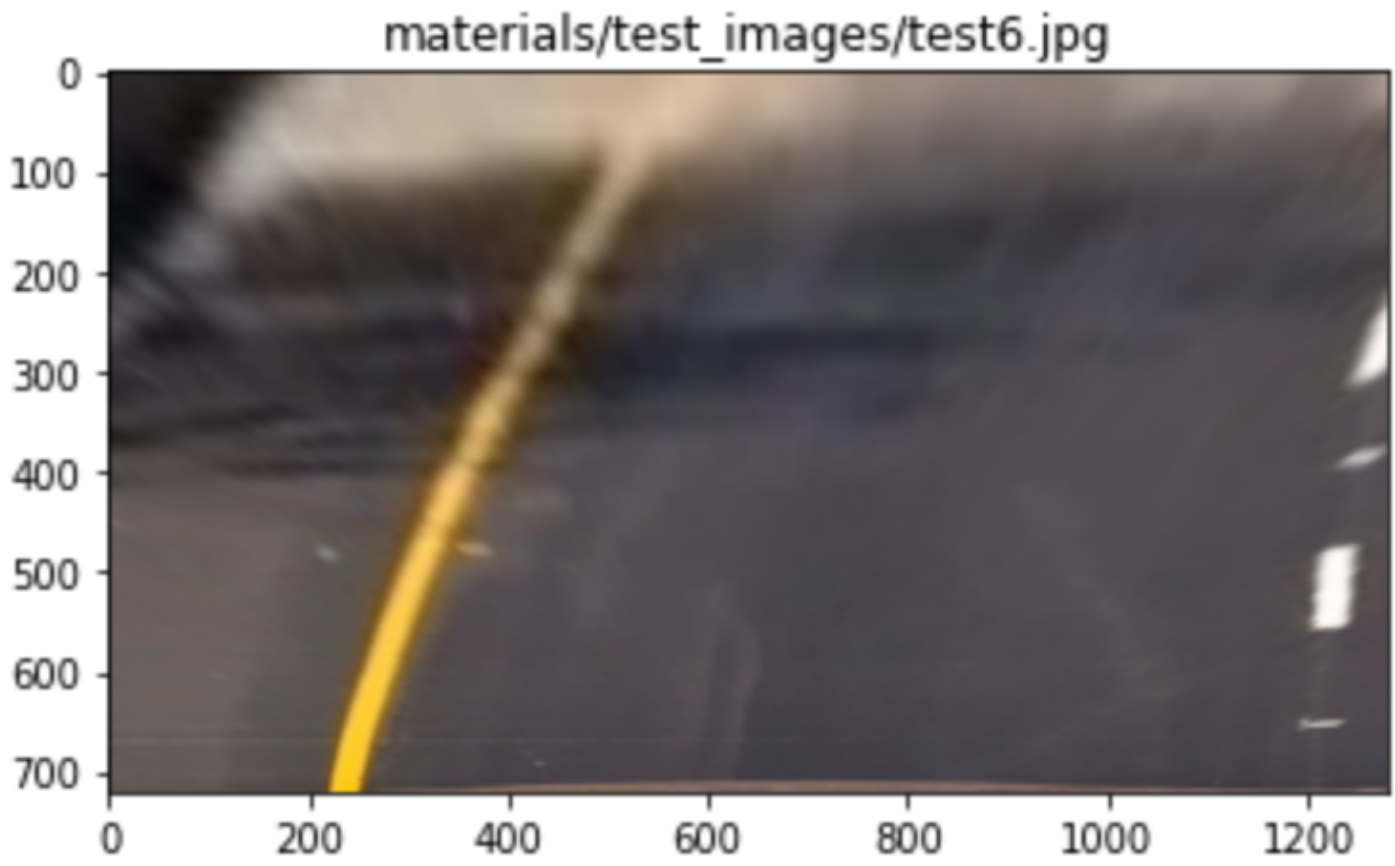
2. Has a binary image been created using color transforms, gradients or other methods?

Yes, I converted each image to the HLS color space and performed binary thresholding on both the saturation and luminescence channel. I also implemented edge detection using the sobel operator and filtered these three results together to obtain a single binary image that best shows the lane:



3. Has a perspective transform been applied to rectify the image?

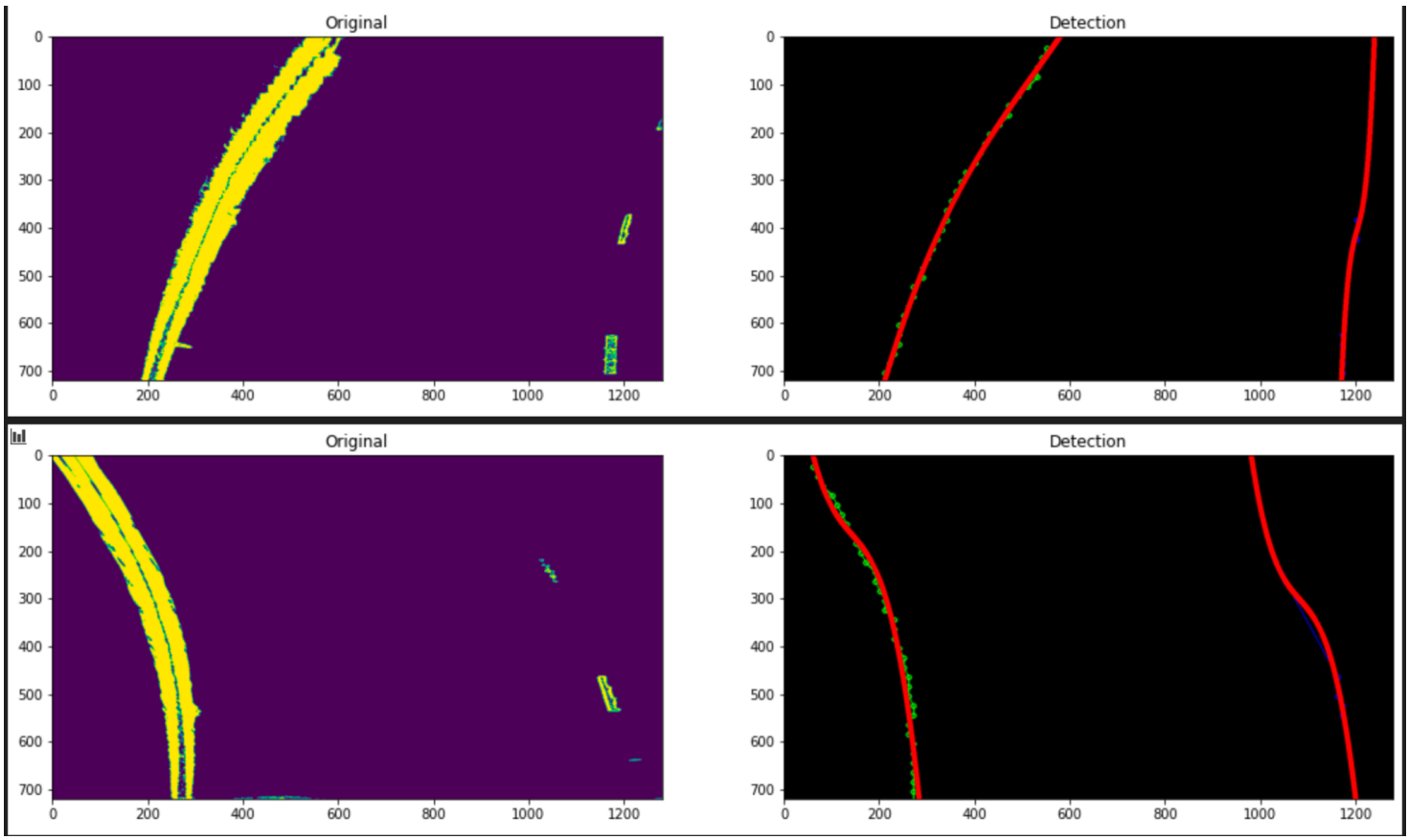
Yes, I created a perspective transform function that calculates src and dst vertices and uses both the cv.getPerspectiveTransform() and cv.warpPerspective() functions to obtain a birds-eye view of an input image.



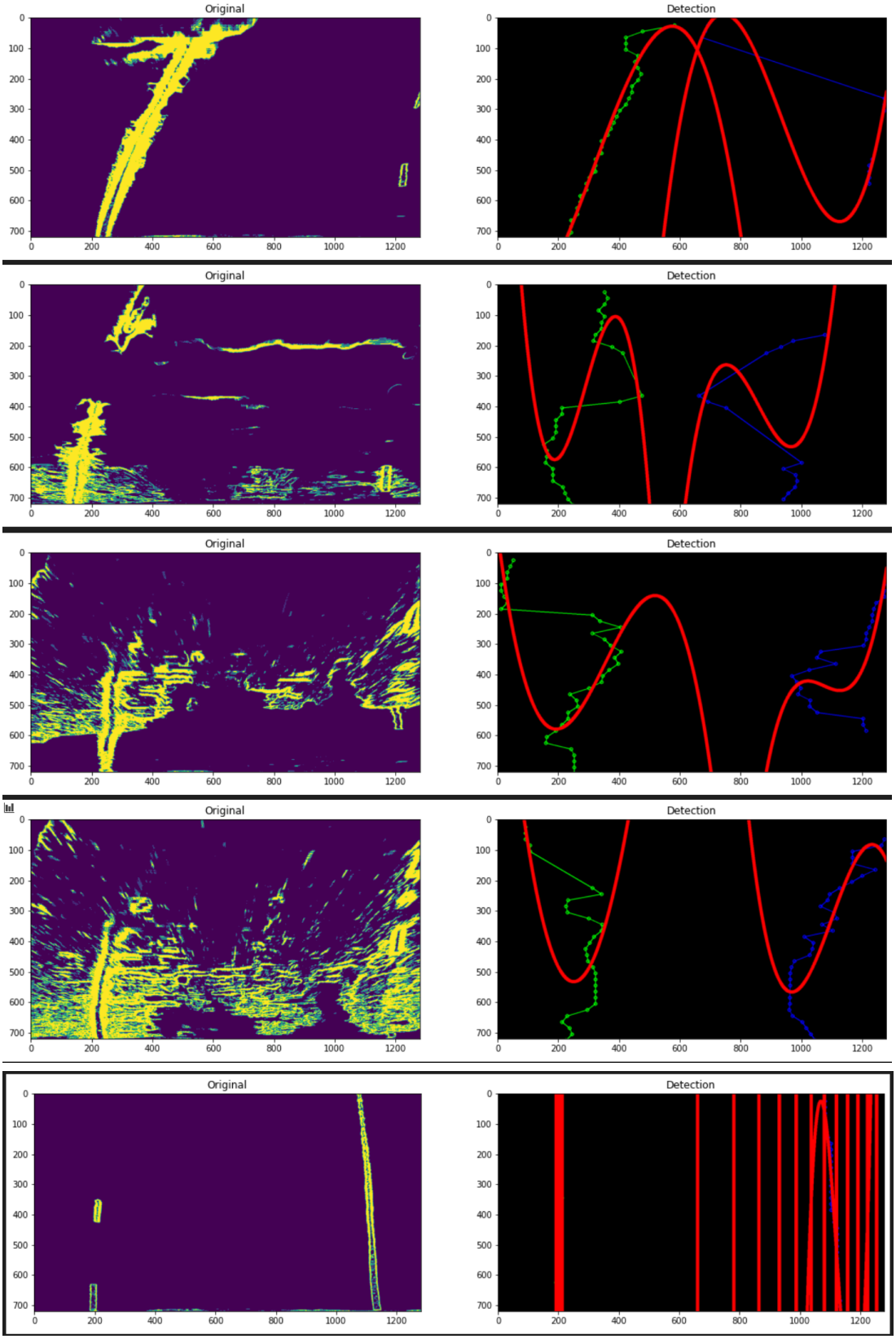
4. Have lane line pixels been identified in the rectified image and fit with a polynomial?

Yes, to a degree as I will explain below. I performed a sliding window search on the binary images from step 2 and stored each region with a certain number of non-zero pixels into an array. I then found the average coordinate position for every row of regions and stored that. I plotted a polyline using these values which roughly resembled the lane markings. I then plotted two 3 factor polynomial lines through each polyline (using scipy's curve_fit() function) which would accurately represent each lane marking.

For two of the images, the output was perfect:



However the 3 factor polynomial curve fitting did not always work as there were also several instances in which it failed to detect the lanes (see images below).


Original
Detection
Original
Detection
Original
Detection
Original
Detection
Original
Detection

5. **Having identified the lane lines, has the radius of curvature of the road been estimated? And the position of the vehicle with respect to center in the lane?**

Unfortunately, after multiple attempts to identify the lane pixels (over a few days), I sadly ran out of time and was unable to perform this final step. (Post submission of this assignment, I would be very interested in learning more about how to correct what went wrong).

Pipeline (video)

1. Does the pipeline established with the test images work to process the video?

In theory the pipeline does work on videos, however it seems to take an incredibly long time to process videos and therefore it is clearly not as effective as it could be. My pipeline was only able to process the video partially before I had to submit and stop the jupyter notebook kernel. (I plan to upload the fully processed video to my repository once it has finished exporting from moviepy).

The current version of my video result is found in the output_video folder in my repository.

Discussion

I believe that the reason for my inadequate results stems from my binary thresholding. While I attempted to use multiple techniques to filter out noise, on the majority of the test images there was still a lot of residual noise, which made it difficult for the pipeline to detect what was a lane marking and what was random noise. In the end, it detected a lot of noise as being a part of the lane and failed to detect other parts of the lane that had been somewhat filtered out previously. This made it difficult for the polynomial fit to accurately represent the lane markings, which led to strange results (such as parabolas, etc.)

It is because of this that I believe that if I were to pursue this project further, the main way I would improve on it would be to fix the thresholding. I don’t believe that my binary thresholding method requires much improvement other than some slight rewrites and tweaking of values.

While I did not finish step 5 (identifying the curve of the lane and vehicle position), I did think a lot about how I would have achieved it:

In order to find the vehicle's position, I would calculate the mean x position of both polynomial lines at a y-position of zero (this would be on the warped image). I could then state that the distance between the vehicle and the center of the lane would be the absolute value of the average x position subtracted from the center x position of the image (width / 2). I would also find the radius of the curve by iterating through segments of each polynomial line and calculating the equation for each segment. I would use this equation to determine if this segment is straight or a curve. I could then approximate the radius of the curve by calculating the radius of each segment (length of the segment / (2 * pi)) and then returning the mean of that result for each segment.

Thank you.