Finding Lane Lines on the Road

Tiffany Huang

April 10, 2017

1 Description of Pipeline

My lane-finding pipeline consists of 6 steps. It takes a camera image as input. Fig. 1 shows an example input image and the following figures show the image after each step of the pipeline. The first step is to apply a color filter to the image to find the white and yellow lane lines and filter out most of the rest of the road. I used `red_threshold = 200`, `green_threshold = 200`, and `blue_threshold = 0`. The example image after color selection is shown in Fig. 2.

Figure 1: Example image after color selection.

The second step is to crop the image to a region of interest where the lane lines are most likely to be found. This step makes it easier to avoid noise or outliers from other parts of the image that may have passed the color filter such as the sky or faraway lane boundaries that belong to other lanes. Fig. 3 shows the example input after the region of interest mask has been applied.

Next, the image is grayscaled so that it can be used as input into the OpenCV Canny edge detector. In the fourth step, Gaussian smoothing is applied to the grayscale image to reduce noise in the image with a kernel size of 5 as shown in Fig. 4.

The smoothed image is passed to the Canny edge detector which finds edges in the image. The Canny transform is an important step because the lane boundaries can be detected by strong changes in the gradient where the image transitions from black pixels (pixels that didn't make it through the two previous filters) to pixels that passed through the color and region of interest filter. An example output is shown in Fig. 5 using a Canny `low_threshold = 50` and `high_threshold = 150`.

Finally, a Hough transform is applied to the image using OpenCV to detect the lines in the image. Once the lines have been detected, they are drawn on top of the original image like the examples shown in Fig. 6. The first image is the example image used in the above figures and the next two images are other example outputs. The Hough transform parameters used were `rho = 2, theta =` $\frac{\pi}{180}$, `threshold = 15, min_line_len = 5`, and `max_line_gap = 5`.

To draw a single line each for the left and right lane boundaries, I modified the `draw_lines()` function as follows. For each line found using the Hough transform, I calculated the slope and length of the line. If the slope was between -2 and -0.5 (remember that for image coordinates, the origin is in the top left corner, y increases going down, and x increases going right) and one of the endpoints of the line segment was on the

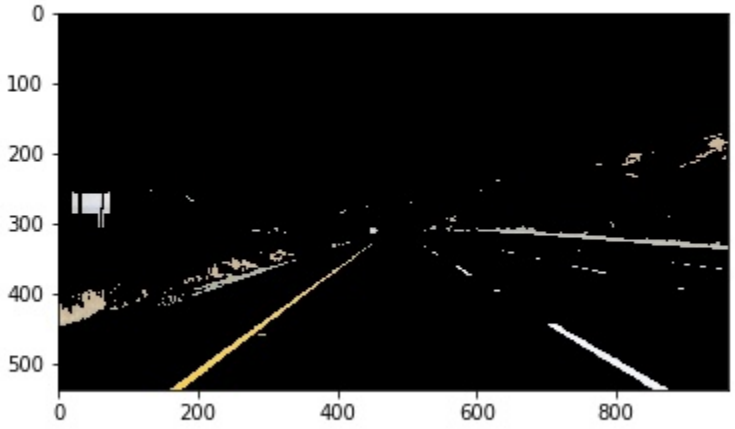

Figure 2: Example image after color selection.

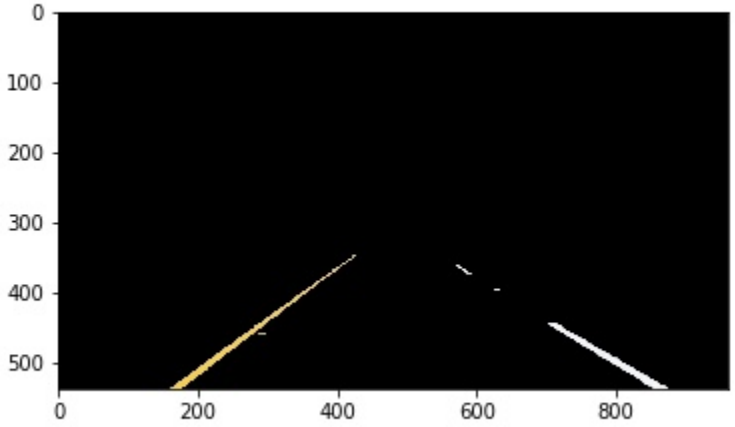

Figure 3: The region of interest mask applied to the example image.

left side of the image, then I categorized this line segment as part of the left lane boundary. If the slope was between 0.5 and 2 and one of the endpoints of the line segment was on the right side of the image, then I categorized this line segment as part of the right lane boundary. The lower and upper bounds of the slopes as well as the left and right side image constraints prevent strong outliers from affecting the average slope and position of the lines. Example outputs for this modification are shown in Fig. 7.

I then found the average position of the all the lines on the left and right side and found the weighted average slope of the left and right side, where the weight was the length of the line segment. The slopes of longer line segments were weighted more heavily to increase robustness to outliers and noise. Finally, using the average positions and the average slopes, the left and right lines were drawn by extrapolating each side from the bottom of the image to the top of the region of interest.

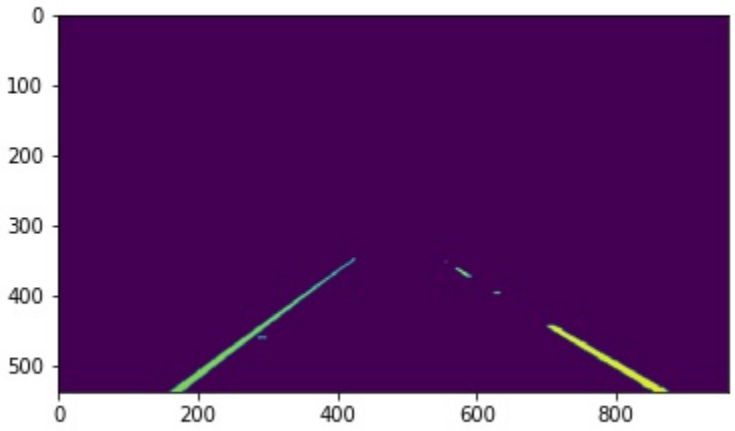

Figure 4: Example image after Gaussian smoothing.

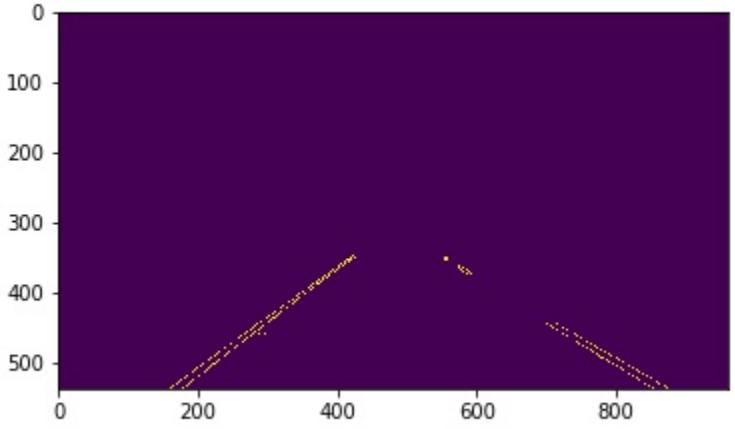

Figure 5: A Canny edge detector applied to the example image.

2 Potential Shortcomings

A potential shortcoming for this pipeline would be if there were extra road markings in the road besides the lane boundaries. For instance, if there were white arrows or letters painted on the road in the middle of the lane, then this pipeline would think these markings were part of the lane boundaries. Those edges and lines would be included in the averaging and extrapolating, which would make the lane detection very inaccurate.

Another possible shortcoming is if the car changed lanes. Depending on which lane boundaries you want to detect, the current region of interest could be too small to capture both the previous and the new lane.

If the car is closely following another car that is very light in color, the lane detection would go awry because if the other car is close enough, it will be in the region of interest and it will pass through the color selection filter if it is a light color such as white. The pipeline will think it is part of the lane boundary and the output will be very wrong.

Furthermore, if the lane had a sharp turn in it, the pipeline could not pick up on the curve because the Hough transform is used in this pipeline, which does not have the ability to detect curves. The assumption made in this project is that the lane boundaries on either side are straight for a certain distance in front of the car.

Finally, as for many other computer vision algorithms, this pipeline is not very robust to lighting changes. If the road is shaded by trees, the color of the lane markings in the frame could change significantly and may no longer pass the color thresholds set in the pipeline. However, if the color threshold is lowered to account for this, we could pick up a lot of other things on the road that are not part of the lane boundaries within the region of interest in frames that have more sunlight.

3 Possible Improvements

A possible improvement to the pipeline would be to keep some history of the output in the previous frame. Since the lane boundaries are unlikely to change very rapidly, they should be smooth from one frame to the next. Running a low-pass filter to smooth the output from the previous frame to the current frame, should make the lane boundary extraction much smoother and more robust to outliers that might cause the slope of the lanes to change too much between frames.

Another possible improvement would be to implement a way to detect lighting changes and adapt the pipeline based on that. For instance, a possible solution could be to average the pixel intensities of the entire frame and if the average intensity drops below a certain threshold, then the color selection thresholds would change to account for the different lighting.

A third improvement would be to weight lines closer to the car higher than line segments farther away from the car. This step would improve the stability and robustness of the pipeline because the lines closer to the car are easier to see with the camera and they are straight relative to the vehicle, which avoids the problem that the Hough transform does not model curved lines.

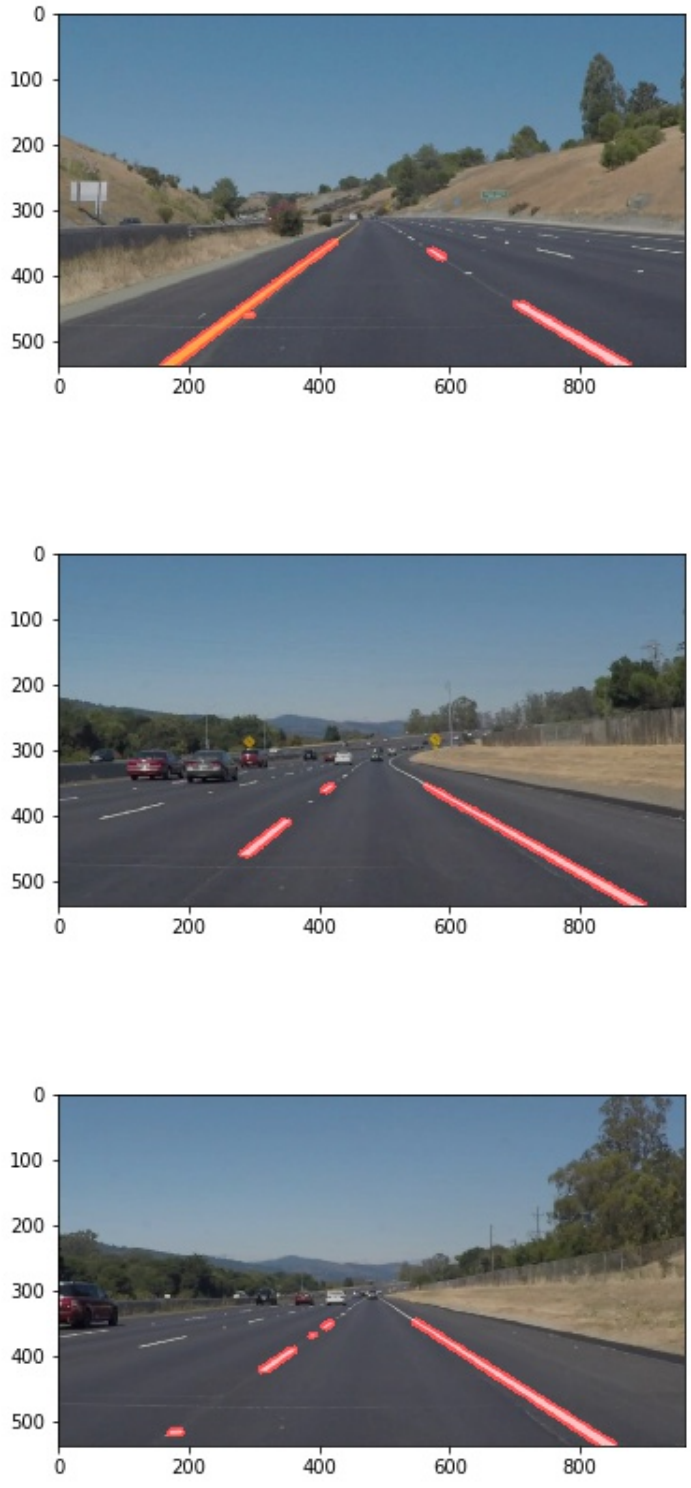

Figure 6: Examples of the final output of the Lane-Finding Pipeline

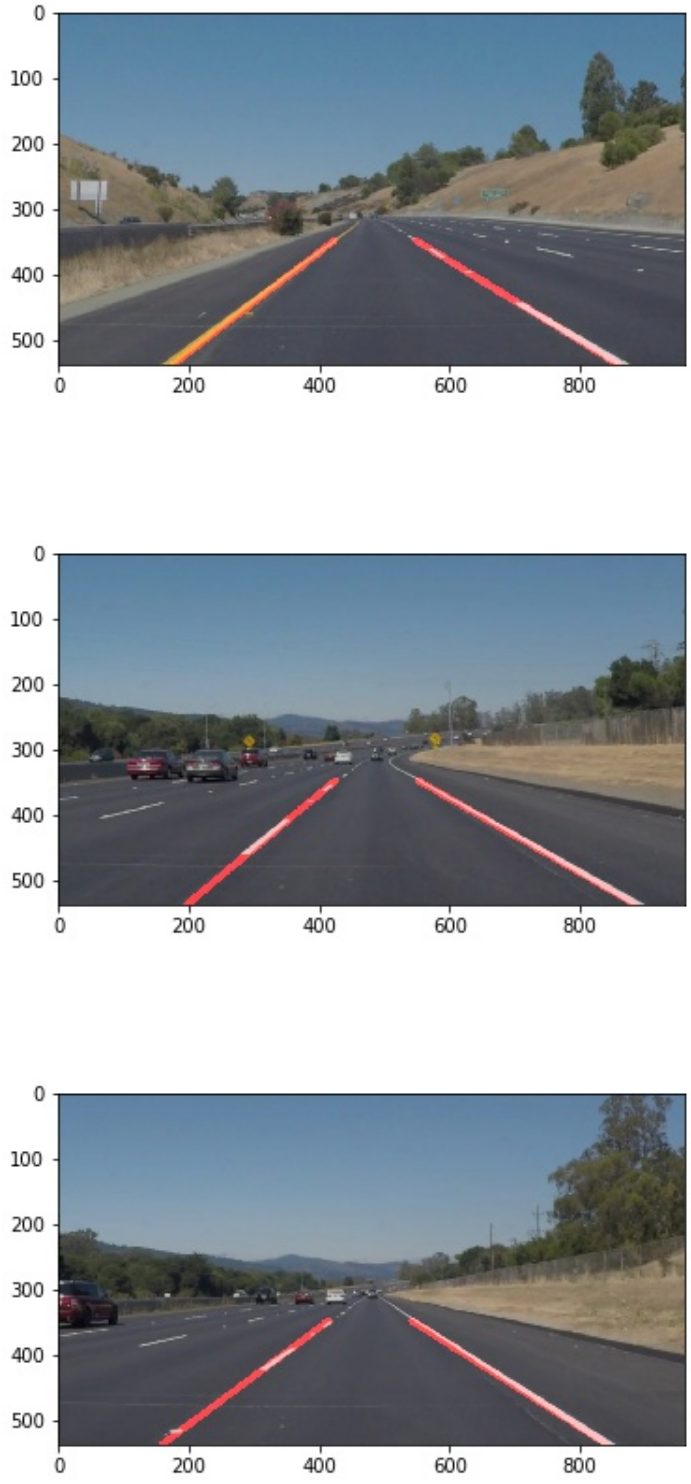

Figure 7: Examples of the final output of the Lane-Finding Pipeline with one line each for the left and right lane boundaries.