Hyunjun Yoo
20505006

Finding Lane Lines on the Road

The goals / steps of this project are the following:

* Make a pipeline that finds lane lines on the road

* Reflect on your work in a written report

1. Pipeline Breakdown

To accurately extract the edges, the implemented pipeline consists of 6 sequential operations. From the original image, a triangular region of interest, drawn in Figure 1, is cropped to remove unwanted features and reduce unnecessary computation.

Figure 1. Image cropping for a triangular region of interest

To identify the white and yellow lines from the image, thresholding is then applied to the region of interest based on the pixels' RGB value. This process filters out any pixels with either red or green value below the threshold. An example result of the thresholding is shown in Figure 2.

Figure 2. Image thresholding using RGB value

With the lanes clearly distinguished from the surrounding, the image is converted to grayscale and blurred using Gaussian filter, which is a common preprocessing technique for edge detection. Canny edge detection is then applied to the preprocessed image with lower bound threshold for hysteresis process, removing weak edges that can be caused by small, unintended features.

Lastly, the image goes through Hough transform to convert the pixels found from Canny edge detection into a set of edges that can be used to identify the large, global line. An example result after Hough transform is shown in Figure 3.

Figure 3. Detected edges after Hough transform

In order to draw the line on the left and right lanes, draw_line was modified to extrapolate a global line from the set of small edges. First, the slope of all edges was calculated and distributed to several buckets of the equal range. Two buckets with the highest count of edges were chosen as the true edges for the left and right lane. Left lane is extrapolated from the bucket with lower range of slope (the "left bucket"), and right lane is calculated from the other (the "right bucket").

Both left and right buckets were then filtered again with the coordinate of the edges – If the edge in the right bucket had an x-coordinate that belongs to the left bucket, that edge was removed. Figure 4 shows the result of this filtering process. The edges for left and right lane are drawn in green and blue, accordingly. The edges that were filtered out from either stage is drawn in red.

Figure 4. Filtered edges separated by the side of the lane

The edges filtered are then used to find the left and right lane using the best-fit extrapolation (implemented in `numpy.polyfit()`). The left and right lane estimated as the result is shown in Figure 5.

Figure 5. Estimated left and right lane

2. Challenges

The pipeline described in this report is heavily dependent on the success of the effect of image cropping and RGB thresholding. The algorithm is making two assumptions about the input image:

- There are little to no unwanted objects, such as car or trees, within the region of interest
- Only the lanes will have the colour of white and yellow, with specific level of brightness

This proves to be a very fragile solution, as there are many cases that can break this algorithm:

- Car suddenly makes a left (or right) turn, and the lanes that need to be detected no longer exists within the region of interest
- RGB thresholding picks up a white or yellow car that is driving in front of the camera
- The car lane is darkened by the shadow from a tree, or the area near the car lanes are illuminated by a bright source of light

Some of these cases are illustrated in the example shown in Figure 6. In this figure, the ground is too bright to be properly filtered out from RGB thresholding, and introduces huge amount of edges that doesn't represent the lane.

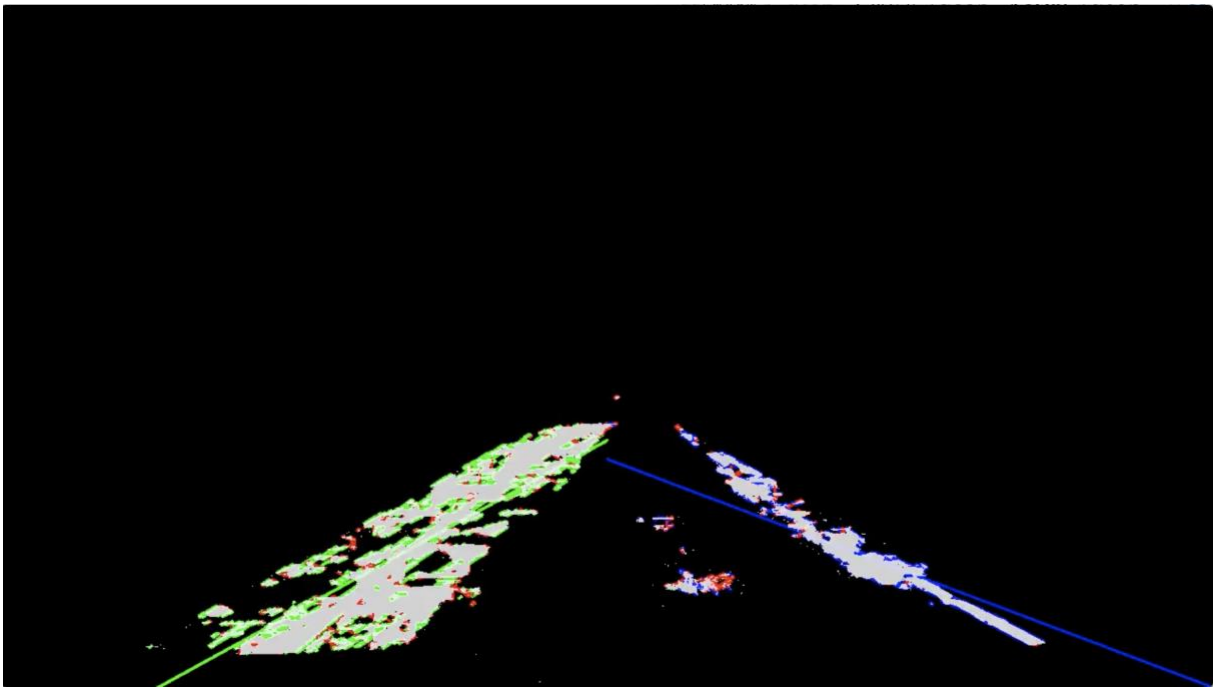

Figure 6. Detection result from a different example with higher brightness

3. Improvements

As mentioned in Section 2, the most important issue to be dealt with the pipeline is its narrow applicability – the algorithm only works for the given example data. A generic approach is required to address this problem. One alternative is to dynamically set the threshold value for RGB thresholding by sampling the overall brightness of the image. One can also refine the processing after Hough transform to recognize two nearly-parallel lines adjacent to each other (appearing from the edge of the lane) for both left and right lane instead of simply averaging the slope of all edges.