A Simple Approach for Finding Lane Lines on the Road

By

Ayham Zaza

A mini-report submitted in partial fulfillment for the requirements of completing Self-Driving Car Nano Degree Program offered by Udacity

Table of Contents

Introduction

Detecting road lanes is a very vital and primary step in developing reliable and robust autonomous driving vehicles. In the following preliminary project offered by Udacity self-driving car development team, some basic computer vision techniques are utilized to detect simple road lanes. This warm-up project aims at devising a structured pipelined function calls that eventually accomplishes the required task and allows additional complexities to be added later. The following section sheds some light on the used methodology and the testing environment. After that, various results are shown and discussed. This reports concludes by listing some current limitations and future direction.

Methodology

We start by converting the input colored images into gray scale one. This down-sampling is necessary to capture various image features prior to applying the famous Canny edge detection algorithm [1]. In order to enhance the recognition process and before a detecting lines using Hough Transform [2], a smoothing function is applied and a region of interest for the most probable lane area is specified. Finally, and after applying some interpolation function, the resulting weighted images are augmented with original ones and the final selection is plotted. Figure 1

```
def process_image(image):
    |
    get_Param() #initialized needed parameters
    gray = grayscale(image)
    blur_gray = gaussian_blur(gray, kernel_size)
    edges = canny(blur_gray, low_threshold, high_threshold)
    masked_edges = region_of_interest(edges, vertices)
    line_image = hough_lines(masked_edges, rho, theta, threshold, min_line_len, max_line_gap)
    lines_edges = weighted_img(line_image, image, α=0.8, β=1., λ=0.)

    return lines_edges
```

Figure 1: Detecting simple lane lines pipeline

The implementation was carried out using Python 3 and utilizing openCV library [3]. The method was first tested over a set of road images and two short videos.

Results & Discussion

Figure 2 shows the results after applying the previously described methodology. The results obtained from the tested videos can be viewed in the accompanied python notebook. It can be visually examined that the method works well for simple lanes.

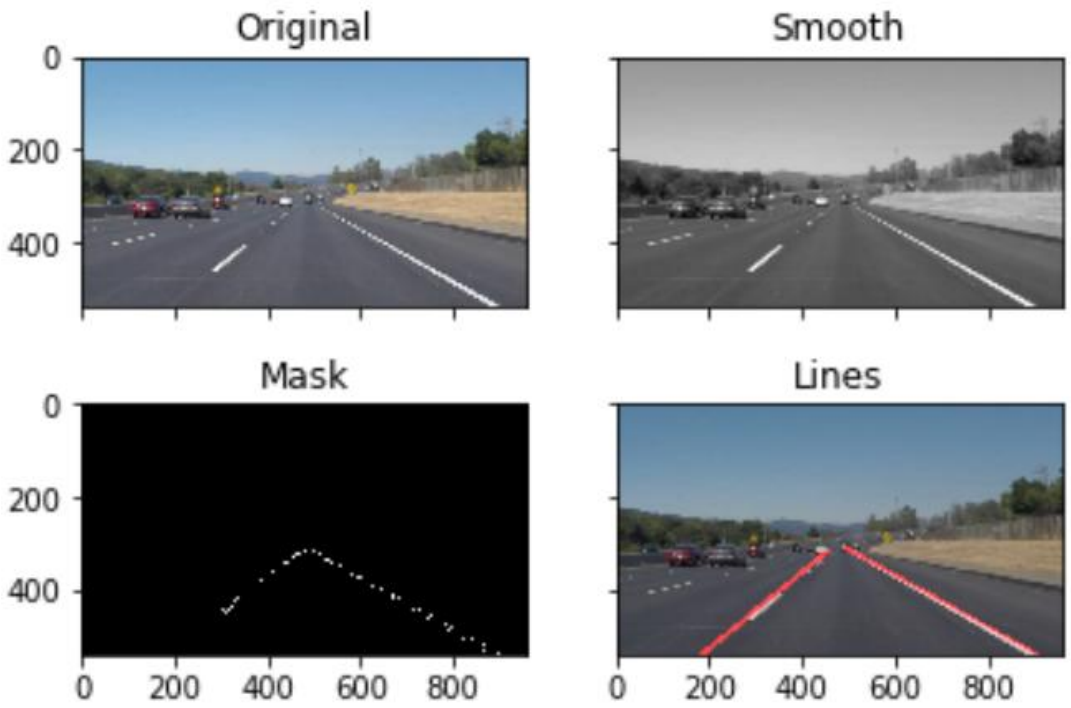

Figure 2: Resulting images after going through the pre-described pipeline and the detected lines

Current Limitations

1. Vertices that define the region of interest are statically defined. To enable dynamic assignment, they should be related to image width and height respectively.
2. At this stage, the interpolation function is roughly defined as the (signed) average of the detected lines. More accurate slope calculations will avoid outliers. That's why in my implementation, the left red line in the second video changes its position slightly when the video is playing.
3. To detect the specified lane, I utilized fixed parameters. I will try testing a dynamic variation in Advanced lane finding project.

4. Curve lines cannot be detected with this implementation. I will aim at adjusting the Hugh Transform. in Project3 so that it detects real roads.
5. Very minor preprocessing was performed. The whole thing might fail if blurred images were utilized or when detecting lanes in a rainy road.

Future Considerations (mainly to be considered in Project 3)

1. Rewriting the interpolation function in more professional and standard way.
2. Enhancing the stability of the detected lines (left lane)
3. Completing the challenge
4. Testing more complex roads (rain, snow, steep curves. etc.)
5. Incorporating some statistical techniques to account for various driving conditions

Conclusion

This report described the implementation of simple lane detection using some computer vision techniques. Despite the various mentioned limitations, it serves as a starting point where more sophisticated ideas can be examined and tested. I would like to thank all Udacity SDCND team for their amazing work and all their exerted efforts to make this program distinguished and enjoyable.

References

[1] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence,* pp. 679-698, 1986.

[2] R. C. Gonzalez and R. E. Woods, "Image processing," *Digital image processing,* vol. 2, 2007.

[3] G. Bradski, "The opencv library," *Doctor Dobbs Journal,* vol. 25, pp. 120-126, 2000.