Finding Lane Lines on the Road

By Abdelhalim Ragab – halim@ragab.me - 10/12/2017

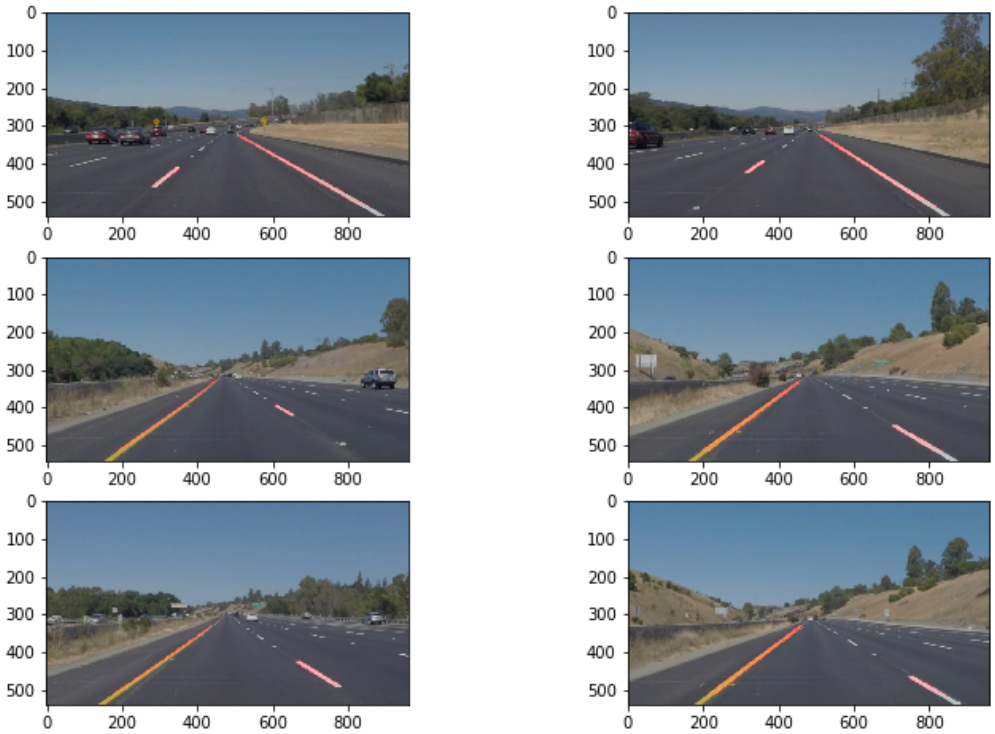
Introduction

The goal of this project is to find lane lines on the road. The project was implemented per the requirements, however it seems the requirement was simplified to be suitable for beginners in the self driving car field (including me) and it doesn't reflect real world requirements for finding the lanes in all conditions e.g. foggy weather, rain, snow, ice, night, sharp curved lanes, turns, … etc.

Pipeline

The implemented pipeline consists of 7 steps

1. Color selection – Select white and yellow colors from existing color image
    - Optional and can be turned on/off by a flag
2. Grayscale
3. Gaussian Blur
4. Canny line detection
5. Region of interest selection
6. Hough Line detection
7. Weighted Image - overlaying the detected lines on top of the original image



Most of these steps has parameters that can be tunes to work better on certain environment. For example, in step #1, the color ranges, one for white and another for yellow, has been tuned to work on the images in the test set.
For the purpose of the project, the parameters are fixed but in real life scenario there might be multiple sets of parameters that can be selected by the algorithm based on the environment (Night vs. Day, Winter vs. Summer, Southern areas vs. Northern areas, State or Country, etc …)

In step #6, In order to draw a single line on the left and right lanes, I created a new `draw_averaged_lines()` function that calculate the slopes/intercepts for all detected lines in the image, average them and extrapolate the line to lower 2/3 of the image. This new feature and the functionality can be turned on and off using the flag.

```
use_average_lines = True
```

Also, as the lane lines should not differ too much from one frame to another, the algorithm caches the previous frame lines and average them with the calculated average for the current frames as a way to use historical values to smooth the detected frame lines.

Pipeline shortcoming

As indicated in the introduction, the requirement of this project was simplified, so for real world application, there will be large number of shortcomings in the pipeline to deal with different environments:

- Identifying lanes at night
    - Even with night vision camera, accuracy and resolution won't be the same as day light, also lane lines will have different colors so color selection needs to be tuned for these colors.
- Fog
- Snow
- Shaded/Dark areas
- Heavy Traffic
    - Too many cars and heavy traffic will make lane lines hidden from camera

Looking only at the project requirement:

- False detection when there are some white/yellow areas in the region on interest that may cause the detection algorithm to consider them as part of the lane lines. This happens even with the averaging, as the averaging does not differentiate between the found lines in the image and consider them all equal.
- The same issue happens when there are big change in the road color itself, like when there is an old section followed by a section with new asphalt, where the difference in colors may cause false detection for a line.

- Another shortcoming in the pipeline is assuming that lane lines are always lines, which means it will not detect highly curved lanes accurately.

Pipeline improvements

One potential improvement for the problem of false detection of white/yellow objects is to filter lines that are horizontal or close to horizontal based on their slopes. Another potential improvement is to calculated weighted average instead of regular average, and give more weight to longer lines.

To detect curved roads, poly fitting techniques can be considered.

Conclusion

Although the requirement for the project was simple, this project was eye opener for the large amount of details that needs to be considered for only one “relatively small” aspect of the self driving car. I’m sure that the techniques we used in this project is just the tip of the iceberg, and we have unlimited possibilities for improvements.