Finding Lane Lines on the Road

Author: Svetlana Strunjas

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

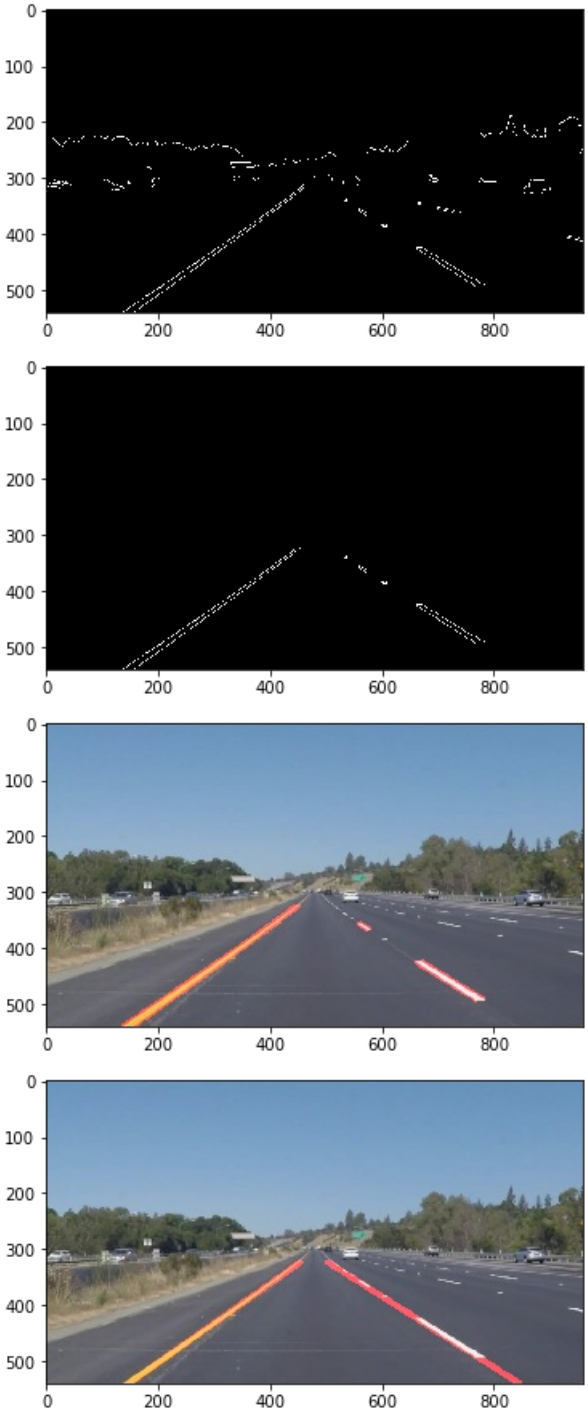
Reflection

1. Pipeline description

My pipeline consists out of next basic elements:

- Gaussian blur computation on an initial image
- Canny edges detection
- Polygon drawing elements for masking relevant parts of Canny edges
- Masking relevant parts of Canny edges and set other to black
- Hough transformation on a masked image
- Average slope and intercept computation from Hugh lines (for creating two smooth lanes from Hough lines)
- Weighted average slope and intercept computation from Hough lines. Weight is determined by a length of each Hough line. This function is used to create two smooth lanes and eliminate noise.
- Computing pixel points for a line, given slope and intercept
- Computing smooth lanes from Hough lines (by using previously mentioned average slope and intercept computation and computing pixel points for a smoothed lines)
- Computed smooth lanes with removed outliers from Hough lines (using some of the previously mentioned functions as well - similar to the previous function)
- Drawing given lines on a given image and returning a merged weighted average image (using a simple function for drawing lines and some other functions)

  These are different phases of a lane detection task on an image:



Besides the previously mentioned basic building blocks of my pipeline, I also wrote :

- Function that aggregates all pipeline steps from an initial image until a point where hough transformation lines are computed. This is a common task for all parts of the assignment, and variations of a solution are accomplished by adding one more step after this one, for some of approaches.
- Final functions that detected lanes on a road. All of them call the previous function, and depending of a subtask, the call a function for either average smoothing or weighted average smoothing or no smoothing at all is executed. After that computed lines are merged with a initial image in the same function .

2. Potential shortcomings with my current pipeline

The pipeline probably doesn’t contain all needed checks for borderline cases. I did some checks, however more could be added.
The pipeline could be simplified a little bit.

Another shortcoming is related to the way I implemented solution for the challenge part of the homework. I managed to detect lanes correctly for the most of challenge video frames, by using weighted average intercept and slope that eliminated some of noise. I also carefully chose vertices for a mask polygon and was able to remove additional noise. However , there are a couple of frames where a given car is driving over very light part of a highway, almost as light as lanes itself. I was not able to correctly capture lanes here. So this solution needs at least one improvement.

3. Possible improvements to my pipeline

Capturing curved lanes (in the challenge part) by lines is probably not an ideal solution to the problem. If I had more time , I would probably research ways to capture curved lanes via polynomial functions. Handling very light parts of a highway when detecting lanes is also not handled properly.