Self-driving car Nanodegree (Term 1). Project 1: Finding Lane Lines on the Road

Ruggero Vasile

May 27, 2017

Abstract

Here is the r for the first project in the self-driving car nanodegree program. The project consisted in developing a operational pipeline to detect lane lines on a street and to test it on image samples and video samples. The pipeline works well on the image samples and on the first two video samples. Several issues are instead present for the optional challenge video that I keep as an open problem.

The pipeline

The pipeline built follows more or less the directives given during the lecture, implementing standard edge detection algorithms, lines extraction, interpolation and extrapolation. Here is the sequence of steps describing the pipeline:

- The first step consists in transforming the RGB image into a one channel image. I chose greyscale transformation but other possibilities are available.
- To average possible noise, I apply a blurring function, specifically a Gaussian blurring. The hyper-parameter, i.e. the kernel extension/dimension is chosen to be equal to 5. However its choice was not so critical.
- On the de noised image, I apply the Canny edge detection algorithm. I start with standard values for the threshold levels and I end up choosing $low\_thresh = 50$ and $high\_thres = 150$.
- Before proceeding, since we are interested in lines on the street only I choose an area of interest in the image, specifically a triangular region which comprehends mainly the street space in front of the car.
- On the selected part of the image I apply the Hough transform and extract straight lines from the appearing edges. This is by far the most challenging step, since parameter tuning was not so easy (several conditions to satisfy). The hyper-parameters chosen are $\rho = 2$, $theta = \pi/180$, $threshold = 60$, $min\_line\_length = 120$, $max\_line\_gap = 100$.
- The result of the previous operation is a set of lines most of which are of similar directions. Therefore the next step was to extract two main directions and locations from this set, corresponding to the two street lines. To do this I employ here a K Means algorithm which clusters the lines together according to their Cartesian parametric representation, i.e. (m, b) and extracts the centroid parameters (I fix to 2 the number of centroids for this particular case). This method worked perfectly for the cases in this challenge, but it is in too much raw form to be applied in general.

- Once the two centroid parameters are extracted I extend the correspondent lines and draw them on the initial input images. In Figs. (1) one can see the result of the pipeline on three of the example images. For the video cases please run the notebook.

Figure 1. On the left original images, and on the right the correspondent images with lines drawn on the top in red.

Potential Shortcomings

The pipeline above defined works good for the images in the example set as well as for the first two videos in the examples. However potential shortcomings can be identified with the current protocol.

- The KMeans algorithm is not flexible enough in its current version. I fixed the number of centroids to 2 which is natural choice considering the problem at hand, but what happens when more than 2 set of lines are identified by the Hough Transform step? Clearly the pipeline would fail. One way

to fix this would be to allow for a variable number of centroids and a post-processing which just selects the two centroids which corresponds to the lines of the street. This is not, however, a trivial problem and would need to be tested in many situations. I leave open the solution to this possible shortcoming since I plan in the future to investigate into it more deeply.

- The choice of the hyper-parameters of the Hough transform are not an easy part. Manual tuning may be dangerous in certain cases. I would opt for an automatic tuning based for instance on some kind on learning algorithm.

Possible improvements

The improvements are essentially the solutions to the shortcomings mentioned in the previous section. The first thing that I would do is clearly improve the clustering algorithm allowing for a variable number of centroids so that it would be able to identify more than two lines in the region of interest. Following this change, there would need to be a way to select which of the lines can be considered street lines. Possible hard coded criteria would be:

- They would need to follow perspective rules, i.e. given that they are parallel, from the perspective of the camera they would appear to be non parallel forming certain angle with the baseline
- Their distance at the baseline would need to be of the order of the car extension.