[ENPM673] Project 2

Anton Mitrokhin, Kanishka Ganguly, Cornelia Fermüller

Due Date: March 13, 2019

1 Introduction

In this project we aim to do simple Lane Detection to mimic Lane Departure Warning systems used in Self Driving Cars. You are provided with two video sequences (both are required for this assignment), taken from a self driving car (click here to download). Your task will be to design an algorithm to detect lanes on the road, as well as estimate the road curvature to predict car turns.

2 General Guidelines

In subsequent sections you will find some advice on how to approach the problem. Nevertheless, you are not required to use any of the described methods and we encourage you to explore more - the approach described here will likely not be enough to produce perfect results (and give you a 100% score for this project). It is part of the task for you to experiment and be creative, and find the solution to the problem which gives the best result.

As always, the assignment is to be completed in Python; simple filtering functions are allowed, but advanced color segmentation algorithms are not. You should always e-mail us and ask if you are allowed to use a certain functionality or not. Also keep in mind, that lane detection is a very popular problem and there are many implementations available on the Internet. If any part of your implementation is not designed by you, you will loose a considerable amount of points.

3 Suggested Pipeline

In our simplified pipeline we will process every image frame independently from each other. This pipeline will rely on the fact that the lanes are parallel to each other and almost always 'vertical', if you look at the image straight *from the top*. The actual camera does not look at the road 'from the top', but it always looks at the road at the same angle. So if we could compute the *Homograpy* and distort the image so the lanes look parallel to each other, we will get the top view.

1. Pick one image when the car moves straight and manually extract the coordinated of four points on the lanes (two for each lane). It should not matter which points you select.

2. Using these points, compute the homography (you can use library functions for that), so that the lanes look parallel to each other on the images after warping (see Fig. 1).

Despite computed only on a single image, this homography transformation will work for most images, since the tilt of the camera is constant. Now, we can proceed to the main pipeline:

Figure 1: *An example of road lanes before and after transforming to top view. The actual image was taken from [1].*

3.1 Step 1: Prepare the Input

1. Undistort the image (using opencv)- camera parameters are provided with the videos.
2. Denoise the image - so that the noise doesn't aggravate any of the further image processing tasks.
3. Apply edge detection to extract edges (you can use Canny or Sobel).
4. Extract the region of interest (ROI). **Note:** For this assignment, we only allow you to crop top half of the image (sky). You *cannot* assume that the car does not change lanes, and define small regions around lanes as ROIs - the whole bottom half of the image has to be in your ROI.

3.2 Step 2: Detect Lane Candidates

You could take two approaches here: use Hough Line transform (in this case, it is possible that you do not even need the homography step). Or build the histogram of lane pixels.

3.2.1 Hough Lines

This approach might not work well on curved roads and/or turns.

1. Find the Hough lines from the edge image acquired earlier (Fig. 2 (a)).
2. Find out the peak Hough lines and extrapolate lines in each group (Fig. 2 (b)).

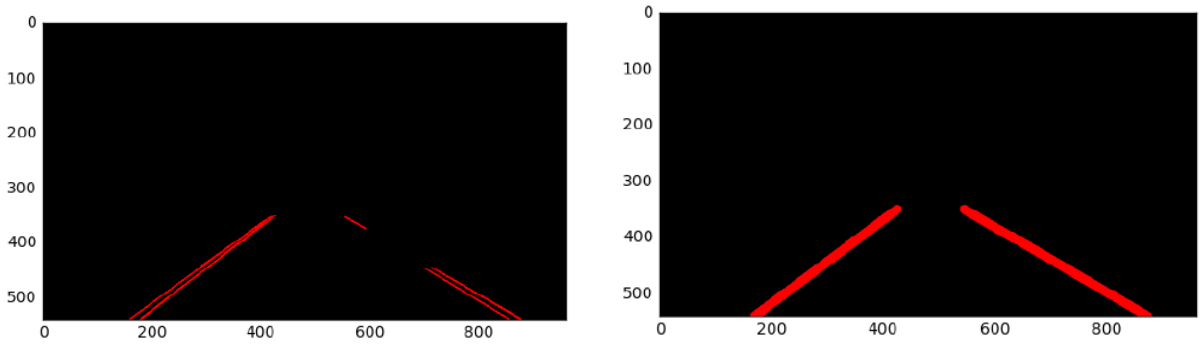
3.2.2 Histogram of Lane Pixels

For this approach you need the homography step. Additionally, you will need to pre-filter the image to extract candidate lane pixels; you can start with a simple Sobel filter, but eventually you might need to consider color segmentation.

1. Generate a histogram of pixel count along the y-axis (Fig. 3 (a)).
2. Extract the top regions with highest pixel count, they will correspond to lanes (Fig. 3 (b)).

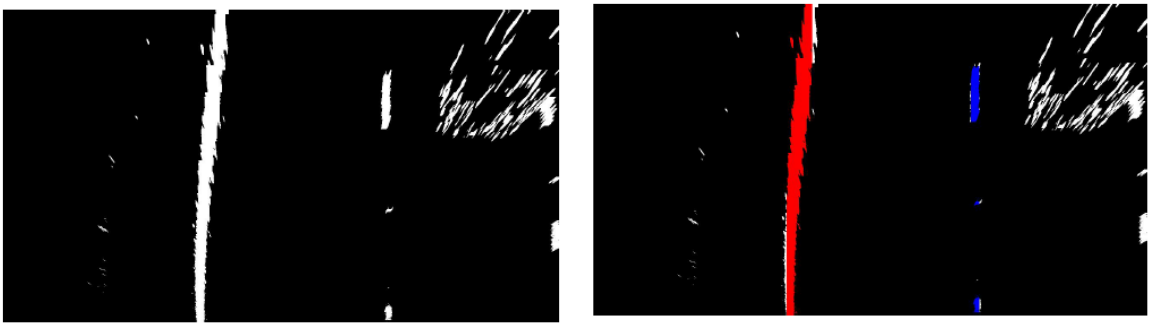
3.3 Step 3: Refine the Lane Detection

Not all lanes are straight - we recommend fitting a polynomial to the detected lane candidates for better results (Fig. 4 (a)). You are required to output lane detection *only for your lane*. Overlay the detected lanes on the original input image, and *also* show the mesh covering the lane itself (an example output is shown on Fig. 4 (b)).



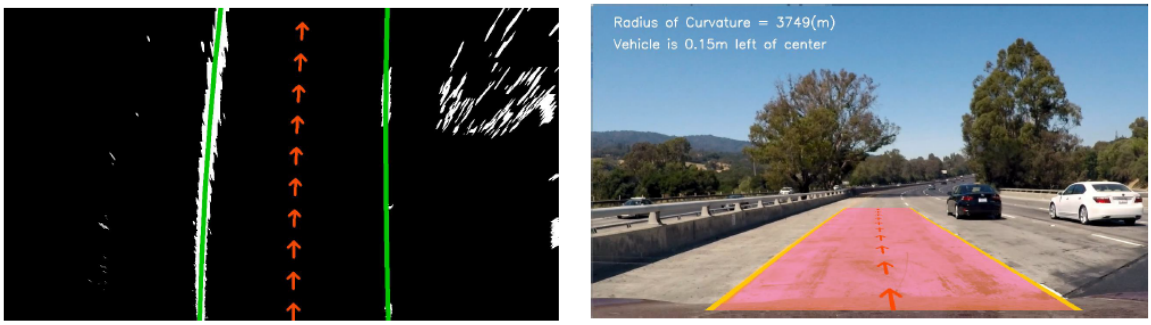
(a) Hough Lines (b) Extrapolated Hough Lines

Figure 2: *An approach using Hough lines*



(a) Lane pixel candidates (b) Top of the histogram (in red)

Figure 3: *An approach using histograms of pixels*



(a) Polynomial fitting (b) Backprojection onto the original image

Figure 4: *Polynomial fitting and an example of the final output*

3.4 Step 4: Turn Prediction

One idea to predict turns is to extract the central line (averaged over left right lanes) and check if the central line has positive gradient (curving towards right) or otherwise. Another way is to use the curvature of the polynomial (fit to right or left lane) to predict the turn. There are many other ways you could explore, essentially the approach you choose should depend on the way the lane detection works.

4 Report

For each section of the project, explain briefly what you did, and describe any interesting problems you encountered and/or solutions you implemented. Include the following details in your writeup:

- Your understanding of homography and how it is used (if you used it).
- Your understanding of how Hough Lines work (whether you used them or not!).
- How likely you think your pipeline will generalize to other similar videos.

5 Grading

Lane detection is worth 80 points, turn prediction is 20 points. In grading of this project we will consider:

- The quality of lane detection (how tightly the lanes are fitted)
- How robust lane detection is - how often does it fail?
- How 'jumpy' the lane detection is from frame to frame
- The quality of your turn prediction
- Given your approach, how likely it is to work for different videos (including ones with lane changes)

6 Acknowledgement

We are using the Advanced Lane Detection dataset from Udacity - Self Driving Nanodegree program.

7 Collaboration Policy

You are allowed to work in teams of up to 3 and discuss the ideas with any students, but you need give credits in the report, if they are not from your team. If you **DO USE** (try not to, it is not permitted) external code - do cite the source. For other honor code refer to the University of Maryland Honor Pledge. You are responsible for finding your teammates, and the projects can be done alone. Nevertheless, you may contact the instructor or the TAs and we might be able to help you find a teammate.

You should take the effort to search online for any help regarding library function usage, however, any concept related queries can be discussed during Offce Hours.

References

[1] B.-S. Shin, Z. Xu, and R. Klette. Visual lane analysis and higher-order tasks - a concise review -. *Machine Vision and Applications*, 25, 08 2014.