Seminar: Technology of Autonomous Vehicles

Topic 1: Camera-based Lane Detection

Topic Description

Topic Summary

The detection of lanes on highways, and other public roads, is an elementary task in the perception of an autonomous vehicles. The main goal of this topic is to provide an overview of the current state of the art for lane detection, and to implement a simple lane detector for highways.

The lane detector should be implemented in Python or C++. For basic computer vision routines, the common open-source library OpenCV should be used. The detector should be designed and tested based on video data of a mono camera from highways in Germany or the United States. For this purpose, multiple data sets are publicly available for download. The final lane detector should be critically evaluated by the team, and compared to state-of-the-art approaches known from the literature. An outlook of possible improvements / further research should be given.

Prerequisites: basic computer vision, Python or C++, (OpenCV)

1 Basic tasks

The basic tasks must be completed before the first deadline (see Moodle!).

Basic Reading

Before you get started, you should be familiar with the basics of computer vision and pattern recognition. Basic books, such as [1] and [2], can help to fill essential gaps in your background. They can also be used as references for basic things throughout your project.

Next, you should familiarize yourself with the OpenCV library and its basic routines.[1] A brief introduction is given in the paper [3]. A more comprehensive introductions is found in the book [4]. But there are many more resources available online that you can make use of, including the documentation of OpenCV.

A standard algorithm for lane detection that you can use includes a combination of multiple filters, in particular the following two:

- Edge Detection (see, e.g., [1, Chapter 7]): You should be able to name and explain a few approaches for edge detection, e.g., based on Canny edge detection or the Sobel filter.
- Hough Transformation (see, e.g., [5]): You should understand what it does and be able to explain how it essentially works.

Finally, you should find some basic literature or technical blogs on lane detection. There is plenty of material available on using different methods (including image processing and deep learning). The goal is to get an overview and a first idea of how to implement these methods.

[1] `https://opencv.org`

Installation of the Toolchain

For the first deadline, you must have a basic toolchain up and running. This includes an installation of Python and OpenCV that is working on your computer. Moreover, you should have a first dataset available, i.e., some video of driving on a multi-lane highway in Germany or the United States.

You should also be able to do some basic image transformations (e.g., edge detection, gray scaling) with your toolchain. Ideally, you already have an edge detection running for your highway data.

2 Final Results

The goal of the implementation part of your seminar project is to design an algorithm for lane detection that successfully works on some example video material. In the end, you should be able to show the performance of your algorithm in a video during the presentation. Note that it is not necessary for your implementation to work in real time.

Ideally, you also want to compare different approaches. These approaches may evolve according to your learning experience during the project – i.e., something that was tried first and did not work so well, then some improvement was made, and so on. In addition, you may play around with different types of image preprocessing (grayscale, region of interest, etc.) and different filters (edge detection, Hough transformation, etc.). Possibly you also want to introduce new filters or methods, based on your own research. Ideally you are able to improve your algorithm over a few design steps.

Another research direction is to implement a lane detection approach based on deep learning. Modern image processing often makes use of Convolutional Neural Networks (CNNs). There are also many software libraries available that you should take a look at, such as TensorFlow[2] or PyTorch[3].

For the presentation of results in your final report, it is always a good idea to define come up with some type of performance measure(s). For example, for what fraction of the total time a lane boundary is successfully detected, possible distinguishing between false positives and false negatives. Then you can present the performance of different approaches in a table or figure. Computation time is also an interesting aspect of your implementations. Consider presenting some statistics about it in your final report.

An important extension is to not just try to identify the lane boundaries at the current position of the car, using straight lines. For most automated driving funtions, it is beneficial to cover also the future shape of the lane, e.g., by providing some curvature information. Ultimately, the lane boundaries could be represented by fitting polynomial curves into the image ahead.

Other possible areas of research include the testing of your algorithm on different types of roads (different line colors, high curvatures, etc.) or under harsh conditions (fog, snow, etc.)

3 Content of the Report

The first part of your report should provide an introduction to your topic. This includes a short paragraph explaining your topic in detail, and how it fits in the context of automated driving. It is also important to motivate your topic, i.e., explain why it is important to consider this topic. The introduction should be followed by a brief literature overview. For the seminar report, it is not necessary to cover a long list of references. About 4-5 scientific papers or books, most of which you ideally found on your own, are enough. The entire introduction should be not more than 1 page in length.

In a second part, you need to provide some background on the methods and algorithms that your implementation is built upon, i.e., edge detection, Hough transformation, neural networks, etc. Keep this part rather brief, around 2-3 pages, but concise and to the point. This background is not the main

[2]https://www.tensorflow.org

[3]https://pytorch.org

focus of your report. However, the reader should get the necessary information and notation to follow the rest of your report.

The third part describes your particular implementation details, i.e., how you modified or combined the basic methods described in the second part. Try to make this readable for your peers. In essence, persons with a technical background similar to yours, who are not familiar with your particular topic, should be able to follow it.

The fourth part is about your results. In your presentation it is nice to show some cool videos here. In the report, you have to use pictures, figures, and tables instead. You should thoroughly evaluate the performance of your algorithms, their robustness and their practicality. Use your engineering judgment, as well as common sense, to come up with interesting perspectives. Then think about how you can illustrate the results as clearly as possible, using representative pictures, figures, and tables. Also, do not forget to compare your results to the current state of the art and discuss potential directions for further improvements and future research.

The third and fourth part should make up the bulk of your report. The total length of the report should be about 10-12 pages. You can end the report with a short conclusion, which should not be more than a paragraph. Ask yourself what are the 2 or 3 main lessons from the project, that you would take away for yourself.

References

[1] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed. Cham, Switzerland: Springer Nature, 2022.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. New York: Springer, 2006.

[3] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifek, “A brief introduction to OpenCV,” in *35th IEEE International Convention MIPRO*, Opatija, Croatia, 2012, pp. 1725–1730.

[4] G. Bradski and A. Kaehler, *Learning OpenCV*, 1st ed. Bejing, Cambridge, etc.: O’Reilly, 2008.

[5] F. Masci, “Line detection by Hough transformation,” California Institute of Technology, (MN), United States, Tech. Rep., 2009.