Multiple View Geometry in Computer Vision: Assignment #1

Submitted by: Maksim Gusev

Atlantic Technological University

Abstract

The purpose of this article is to develop the new Lane Keeping detection system as a part of Multi View Geometry in Computer Vision. As an initial base was taken a part of CV open courses and conducted a few circles of improvements and fine-tuning of hyperparameters. The final result showed the significant improvements from the initial. However, some of the basic errors are still left. The author proposed some potential of improvement, based on machine learning and advanced computer vision algorithms.

Keywords: Computer Vision, Multiple View Geometry, Lane Keeping Assistent, Probabilistic Hough.

1 Introduction

This article is a part of an assignment for the Multiple View Geometry in Computer Vision module taken at the Atlantic Technological University. The purpose is to implement lane detection and investigate the efficiency of the various hyperparameters settings. As the test video was taken, video from Sligo, Ireland (test video).
The finalized version of the project, as well as the videos, could be founded in a GitHub repository [GusevPortfolio/Computer-Vision].
A lane keeping detection system (LKA) is a safety feature in vehicles that helps drivers stay within their lane while driving. The LKA doesn't work in all conditions and for the transparent assessment of the code the author found the limitations on the official website of Mercedes-Benz. The system may be impaired or may not function in the following situations. The first group is sensor set related situations: poor visibility, e.g. shade or in rain, snow, fog or heavy spray; glare; windshield or camera dirty, damaged or covered. The second group is computer vision algorithm related situations: no lane markings, or single lane markings; lane markings – worn out, unclear or covered; distance to the vehicle in front is too short;

2 Initial code and algorithm description

Initial code was taken from the "OpenCV Python Tutorial for Beginners 31 - Road Lane Line Detection with OpenCV". GitHub repository [pknowledge/detector.py].
The original pipeline of the video processing:

- Colour Space Conversions (original to gray)
  Colour space conversions are the processes of transforming the representation of colours from one colour space to another. RGB (Red Green Blue) is the most common colour space used for digital imaging and computer graphics. In this colour space, colours are represented as combinations of red, green, and blue light. Each colour channel is represented by a value ranging from 0 to 255.
- Canny Edge filter
  The Canny Edge filter is a popular image processing technique used for detecting edges in digital images. It was developed by John F. Canny in 1986 and is widely used in computer vision and image analysis applications. The Canny Edge filter works by detecting sharp changes in pixel values in an image.

- Mask Cropping Image (cropping, fillPoly and Bitwise Operation)
  Mask cropping is a technique used in image processing to selectively remove or retain certain portions of an image based on a predefined mask. The mask is a binary image that specifies which parts of the original image should be kept and which should be removed.
- Probabilistic Hough transforms (HoughLinesP)
  The Probabilistic Hough transform is an extension of the traditional Hough transform used for detecting lines and other geometric shapes in digital images. It was introduced by Matas et al. in 2000 and is widely used in computer vision and image processing applications. The Probabilistic Hough transform works by randomly selecting a subset of points from the input image and then fitting a line to these points using a least-squares approach.

3 Code iterations

The author chose the 4 photos of the hard places/scenarios from the test video, and each iteration was compared with their results. Each iteration description, the author started with an explanation of what exactly was changed and how it affected on the final result (video and 5 photos).

3.1 First iteration or result of initial code

The initial code has a sequence of serious problems. The most serious was a crash of a program if the algorithm has no lines in the picture (e.g. sharp turn or zones no lane marking). The second serious issue was triangle ROI (region of interest), that lead to ignoring lane marking if the vehicle goes uphill and downhill. And the last one is the low threshold hyperparameter in HoughLinesP function. It causes the high level of noise in image processing.

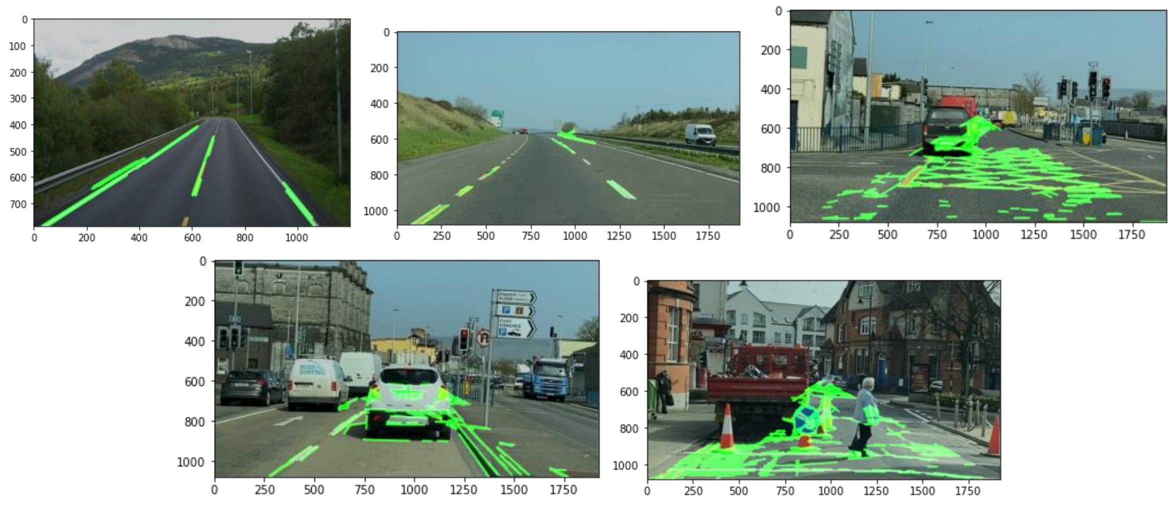

Figure 1: series of photos from the test video passed through the first iteration

3.2 Second iteration or yellow/white masks

The author started on with the crushing error. In the classical approach of the system, if the vehicle must return the control of the vehicle to a driver. However, for educational and training purposes, the author added "if lines is None: return image" function.
The second changing were ROI and masks reviewing. The ROI was changed from the triangle to the truncated pyramid. On the other hand, the mask of the white color was changed to a fusion of white and yellow masks

via the cv2.bitwise_and function. As well was added Blur function to reduce the possible noise (e.g. shadow and marks of road worn).

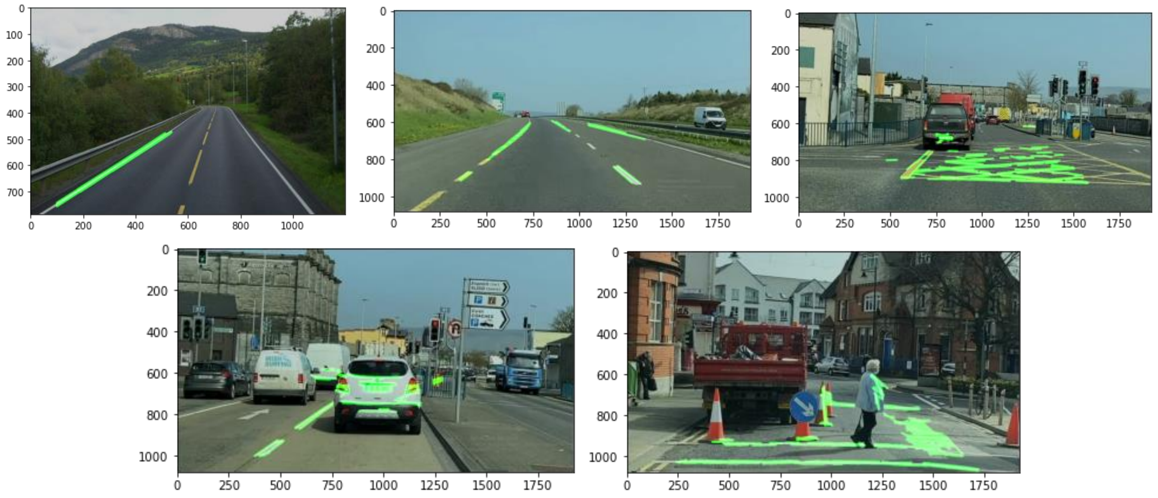
Figure 2: series of photos from the test video passed through the second iteration

3.3 Final iteration or hyperparameters

In the final iteration, the author tries to conduct the fine-tinning of hyperparameters of Canny filter and HoughLinesP function. In many iterations of tuning the author notices, if the masks are suitable enough and see only marking, the Canny filter is not helping and divided the closes line to a few separate lines and that lead to the additional noise. Decided to run the iteration without Blur and Canny filters. The iteration shows the same or little improvements in some cases.

The hyperparameters of HoughLinesP were set as (rho=2, theta=np.pi/180, threshold=125, lines=np.array([]), minLineLength=40, maxLineGap=100). The most iterations were concentrated on rho, threshold and minLineLength and maxLineGap.

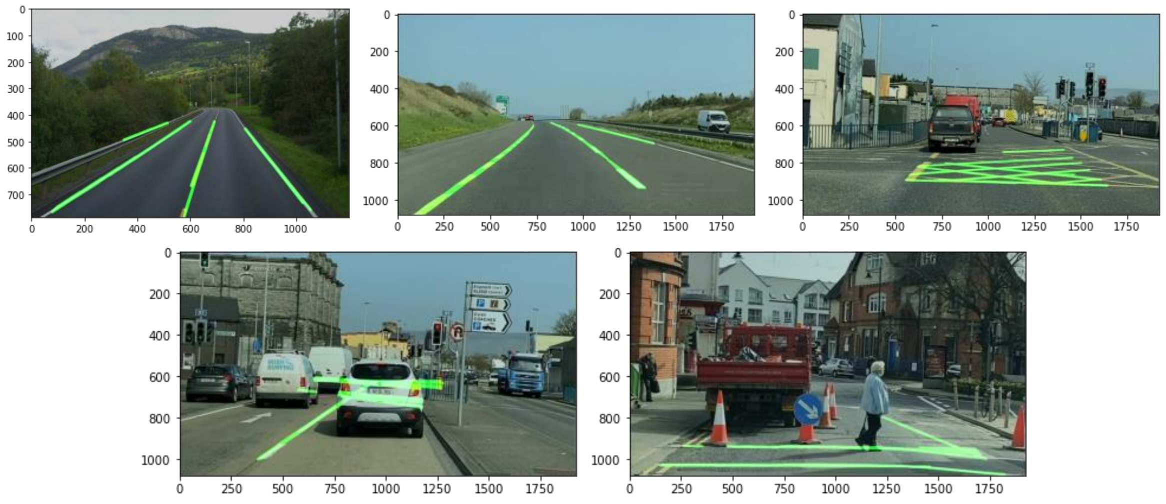
Figure 3: series of photos from the test video passed through the third iteration

4 Conclusions

The developed Lane Control System showed the overall improvement. However, this is not enough to consider this version as a minimal-value-product (MVP). The next possible iterations could include the next steps, as a potential for improvement.

1. Image segmentation - if the author added the semantic segmentation of the picture and extracted the road surface. It could solve the problem with "distance to the vehicle in front is too short" like in case #2. As well, improved the issues with high noise (e.g. fencing road barriers sometime recognized as a line marking).
2. Limitation of quantity of lines - the implementation of limitation of quantity of lines could help us in classical LKA (e.g. 2 lines). However, recognition of stop lines, yellow grating at the crossroads or other markings as well should be recognized as well.
3. Lane separation based on one visible line - the width of the road is strongly defined by the government. If the vehicle sees only the left/right line of the road, the LKA algorithm could complement the second line on the image.
4. RGB-D camera instead of RGB - stereo vision or RGB-D cameras could improve LKA and added/duplicate some functions of advanced cruise control system (ACC).

The implementation of these potential of improvement required more time and deep immersion into the computer vision and machine learning algorithms.

References

[Kim, 2017] Kim, J., & Yoon, K. J. (2017). Robust lane detection and tracking in challenging scenarios. IEEE Transactions on Intelligent Transportation Systems, 18(10), 2647-2660.

[Zhang, 2018] Zhang, L., Wang, Y., & Chen, X. (2018). A real-time lane detection algorithm based on deep learning. In 2018 IEEE International Conference on Mechatronics and Automation (ICMA) (pp. 1740-1745). IEEE.

[Huang, 2019] Huang, Y., Liao, H., & Chen, W. (2019). A fast and robust lane detection algorithm based on convolutional neural network and Hough transform. IEEE Access, 7, 119514-119525.

[Li, 2019] Li, Z., & Wang, M. (2019). Lane detection algorithm based on deep learning with a small dataset. In 2019 IEEE International Conference on Image Processing (ICIP) (pp. 3876-3880). IEEE.

[Lee, 2018] Lee, J. H., Kim, S. H., & Lee, S. (2018). Real-time lane detection using convolutional neural network and inverse perspective mapping. In 2018 IEEE International Conference on Consumer Electronics (ICCE) (pp. 1-2). IEEE.

[Gao, 2020] Gao, Y., & Liu, Y. (2020). A novel lane detection algorithm based on deep learning and adaptive thresholding. IEEE Access, 8, 12913-12923.