Supplementary Information: Learning Vision Based Autonomous Lateral Vehicle Control without Supervision*

Qadeer Khan, Idil Sülö, Melis Öcal and Daniel Cremers

Computer Vision and Artificial Intelligence Group,
Technical University of Munich, Boltzmannstrasse 3, Garching, 85748,
Bayern, Germany.

*Corresponding author. E-mail(s): qadeer.khan@tum.de;
Contributing authors: idil.sulo@tum.de; melis.oecal@tum.de;
cremers@tum.de;

The supplementary material contains the following items:

- Code and the pre-trained models for performing inference. It contains the necessary scripts and information on how to execute these files. It can be found here[1]
- The code repository also contains a video depicting an online evaluation example on CARLA [1] of the various configurations described in the main paper. The video also shows synthesized images at different locations for 4 different scenes on both the CARLA and KITTI dataset.
- Qualitative results on the KITTI dataset for the following components of our framework: visual odometry, view synthesis and label generation using Model Predictive Control (MPC).
- Further details regarding configurations for MPC and the training of our network.
- Limitations of visual and learning components.
- Additional limitations of the supervised model.

*This work was supported by the Munich Center for Machine Learning

[1]https://github.com/idilsulo/nn-driving

1 Qualitative Results on the Real World KITTI dataset

Note that in Figure 1 of the main paper, our framework comprises of 4 components, namely: visual odometry, novel view synthesis, MPC and neural network. Only the neural network which predicts the steering command requires interaction with the environment for which evaluation is not possible with static images on real world datasets. Nevertheless, we can still report the qualitative results of the other 3 components. Figure 1 shows the results of running [2] as the visual odometry algorithm on Sequence 00 of the KITTI dataset [3]. Also, shown is the ground truth trajectory. It can be seen that the result of visual odometry closely follows the ground truth.

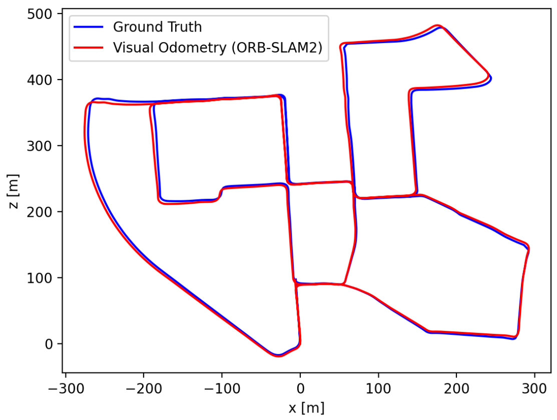

Fig. 1 Shows the KITTI trajectory generated by ORB-SLAM2.

One advantage of our framework is that the 4 components are independent of one another. Each component can be individually improved without affecting the performance of the other. Therefore, if an even better implementation of a visual odometry algorithm is available then one can replace their version with the better state of the art implementation.

Meanwhile, the video in the repository shows the view synthesis on images at 4 different locations on the KITTI sequence. The video additionally demonstrates image synthesis and corresponding labels generated by MPC for a subsection of the road.

2 MPC Configuration

There are several parameters that require to be configured while using MPC. In this section, we highlight their definitions and their usage.

Time difference (*dt*):
Note that the motion model equations in the main paper are discretized with a

timestep of dt. This parameter is the difference between two states at time t and $t+1$. For MPC optimizations, it is configured with the same value provided by *fixed delta seconds* parameter of the CARLA simulator. It is equal to 1/FPS for a simulator working at a certain frames-per-second (FPS). Our data was collected at 30 FPS.

Bounds:
Note that the car can be controlled by adjusting the throttle and steering command, which in turn influence the acceleration and steering angle respectively. For MPC optimization, the bounds are the constraints for these commands. The throttle is constrained to be in the range bound by CARLA of $[0,1]$. Additionally, we set the steering angle range based on the how much the front wheels of the vehicle can rotate with respect to the vehicle physics mentioned in the vehicle blueprint in the CARLA simulator. The steering values in CARLA range between -1 and 1, where 1 corresponds to $70°$ for the default vehicle [4].

Steering mechanism:
Note that in the bicycle model [5], the 2 front wheels and the 2 rear wheels are represented by a single front and single rear wheel respectively. However, in the real world, while the car is taking the turn, the 2 front wheels have different steering angles. This is because, when executing a turn, the outer front wheel needs to traverse a larger distance and hence would have a lower steering angle. The difference in the steering angles between the 2 front wheels is dependent on the curvature of the turn, the track-width and length of the wheelbase of the car [6]. This difference can be maintained using the Ackermann steering mechanism [7]. For our purpose, CARLA already compensates for this difference in its steering mechanism [4].

Goal State Search and Horizon:
The goal state search and horizon parameters are used jointly while optimizing for the steering commands in MPC. The goal state search is to determine how far from the ego-vehicle state we place the goal state on the reference trajectory. Meanwhile, the horizon describes for how far into the future we want to optimize for the steering commands. We do the optimization for 10 timesteps into the future, while the goal state is selected to be 10m ahead and dynamically adjusted to 5m when making turns. This is to ensure that the optimization does not cause the vehicle to cut corners, while simultaneously reducing the velocity and abiding by the no-slip condition.

3 Training Details

We had used the architecture from [8] for our network. Synthesized images and also those from the reference trajectory are used for training the network. The images are synthesized at their native resolution of 1200 x 600. However, image synthesis at locations farther distances from the source image leads to visible voids at the boundaries. This is because the field of view (FOV) of the source image does not capture the entire FOV of the synthesized images. We therefore, first center crop the image to 1000 x 600. This cropping, albeit reduces the FOV, considerably mitigates the void regions in the image that are irrelevant for the network. The image is then resized to 128 x 128 before being fed to the network for decision making. *video.mp4* shows this process for synthesized images at different locations for 4 different scenes. Trajectories starting at positions 52, 108, 152, 187, 208, 214 for Town01 of version 0.9.10 were used for training of our method. Whereas trajectories starting at positions 47, 178 were used as the testing trajectories for evaluating our approach. We made sure that there is no overlap between the training and testing trajectories.

Sampling:
It is also worth mentioning that the dataset contains mostly images wherein the vehicle is moving straight. A rare subset of images correspond to the vehicle taking turns. Therefore, training the network with a uniform random sampling will bias the prediction of steering commands towards going straight. Therefore, to compensate for this imbalance in the data, we create a histogram from the steering commands predicted by MPC for the reference trajectory. The histogram divides the steering values into bins. Next, weights from these counts are calculated as the inverse of the empirical priors on each class distribution. Basically for each bin i,

$$W_i = \frac{N}{N_i}, \tag{1}$$

where N_i is the number of samples in the bin and N is the total number of samples. Then, for each steering value we check which bin it falls into and associate it with the bin's respective weight W_i. The vector of weights obtained after this operation is then used for sampling. Hence, a sample falling into a bin with less data has a higher probability of being sampled at training time.

4 Limitations of Visual and Learning Components

As motivated in the main paper, one of the well known limitations of learning is encountering out-of-distribution data at inference time. In the context of self-driving, our paper deals with resolving a common problem of encountering anomalous off-course data. However, there are other situations where the learning model can also fail for e.g. changing weather conditions, which was tackled in [8]. Limitation of vision would include for e.g. impaired visibility caused by dust particles or rain droplets on the lens/windscreen. One solution could be to use LiDAR data [9]. But, they tend to be far more expensive [10] than RGB cameras, requiring more memory and a higher computational cost [11, 12] and can only produce sparse uncoloured point clouds.

5 Limitations of the Supervised model

We had already discussed some of the limitations of the supervised model in Section 5 of the main paper. Another issue with the supervised model is the requirement of having a dedicated driver to collect image-label pairs. It could be argued that a dedicated driver for data collection is not necessary. This is because most modern cars are equipped with the CAN bus from which steering labels can be acquired with some modifications. This can then be scaled to extracting data from modern vehicles driven by normal people. However, the issue with this is that normal people are not expected to have the requisite knowledge to modify their vehicles for data collection. Even if they could, such self-modifications may incur additional insurance premiums or void insurance claims in case of accidents. Moreover, in different places, the law holds the party making the modification liable to damage.

One solution is to have the car manufacture embed this feature of recording and retrieving the images and steering commands executed by the driver directly into the car. Even if the manufacturer can acquire this data, a far greater concern is how to collect off-course data, as this would possibly entail violation of traffic rules. As we had discussed in the main paper, this off-course data is critical in enhancing model performance. But no person would want to risk their license being revoked, car being damaged or most importantly injuring persons for the sake of collecting off-course data. This brings us back to using only a dedicated expert driver who has the necessary permits to perform such maneuvers under a controlled setting in order to collect such off-course data. Again, this is not a scalable solution.

Our approach on the other hand does not require dangerous maneuvers for acquiring off-course data during the data collection phase. Rather, we synthesize off-course images from a single on-course trajectory data which can be obtained from either a dedicated expert driver or even normal drivers. The steering labels are inferred using MPC rather than relying on potential car modifications to retrieve CAN bus data.

References

[1] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Conference on Robot Learning (CoRL) (2017)

[2] Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. IEEE Transactions on Robotics **33**(5), 1255–1262 (2017). https://doi.org/10.1109/TRO.2017.2705103

[3] Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)

[4] CARLA: CARLA Simulator documents: Measurements [Accessed on 07.03.2022]. CARLA

[5] Wang, D., Q, F.: Trajectory planning for a four-wheel-steering vehicle. In: IEEE International Conference on Robotics and Automation (ICRA) (2001)

[6] Rajamani, R.: Vehicle dynamics and control. In: Second Edition, Publisher: Springer (2012)

[7] Zhao, J.-S., Liu, Z.-J., Dai, J.: Design of an ackermann type steering mechanism. Journal of Mechanical Engineering Science **227** (2013). https://doi.org/10.1177/0954406213475980

[8] Wenzel, P., Khan, Q., Cremers, D., Leal-Taixé, L.: Modular vehicle control for transferring semantic information between weather conditions using GANs. In: Conference on Robot Learning (CoRL) (2018)

[9] Prakash, A., Chitta, K., Geiger, A.: Multi-modal fusion transformer for end-to-end autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7077–7087 (2021)

[10] Qian, R., Garg, D., Wang, Y., You, Y., Belongie, S., Hariharan, B., Campbell, M., Weinberger, K.Q., Chao, W.-L.: End-to-end pseudo-lidar for image-based 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5881–5890 (2020)

[11] Liu, Z., Tang, H., Lin, Y., Han, S.: Point-voxel cnn for efficient 3d deep learning. In: Advances in Neural Information Processing Systems (2019)

[12] Liu, Z., Amini, A., Zhu, S., Karaman, S., Han, S., Rus, D.: Efficient and Robust LiDAR-Based End-to-End Navigation. arXiv e-prints, 2105–09932 (2021) arXiv:2105.09932 [cs.RO]