Lateral Ego-Vehicle Control without supervision using View Synthesis (Supplementary Material)

Anonymous ECCV submission

Paper ID 7752

The supplementary material contains the following items:

- A video depicting the an online evaluation example of the various configurations described in the main paper. Please refer to *video.mp4* for evaluations done on CARLA [4]. 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).
- Additional experiments/evaluations describing the implications of using visual odometry (VO) trajectory and synthesized images as opposed to ground truth trajectory and images.
- Additional experiments demonstrating that the true driving quality discerned from the online evaluation does not correlate with the offline metrics.
- Further details regarding configurations for MPC and the training of our network.
- Implications of when the assumption of the no-slip condition is broken.
- Limitations of visual and learning components
- Additional limitations of the supervised model
- Errata

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 [9] as the visual odometry algorithm on Sequence 00 of the KITTI dataset [5]. Also, shown is the ground truth trajectory. It can be seen that the result of visual odometry closely follows the ground truth.

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.

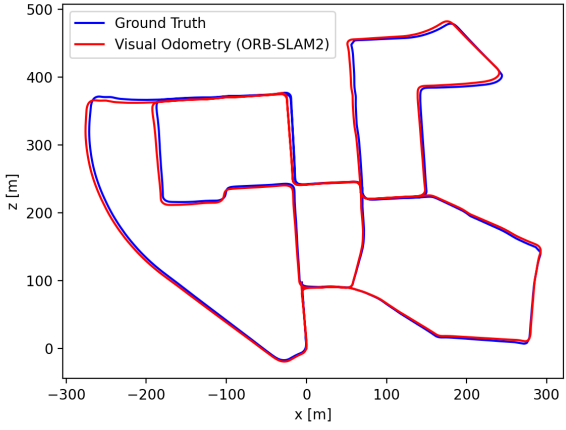

Fig. 1. Shows the KITTI trajectory generated by ORB-SLAM2.

Meanwhile, the video.mp4 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 Implications of using VO and Synthesized Images

This section is a further extension of the evaluation done in the main paper. Note that our method was trained with visual odometry trajectory and additional synthesized images. However, in the simulator we have access to the ground truth trajectory and can also collect additional ground truth images. Experimental results in the main paper showed that the performance of our method was on par with the model trained with ground truth trajectory and ground truth images. We investigate this further by evaluating 2 additional models. One is trained with ground truth images but trajectory obtained from visual odometry. The other is trained with the ground truth trajectory but synthesized images. The results of all 4 models are reported in Table 1.

As can be seen, the performance of the 4 models are similar. However, note that the ground truth trajectory and images will not necessarily be available in the real world. Therefore, this evaluation suggests that even without ground truth data we can achieve on par performance with our approach of using visual odometry trajectory and synthesized images. This also aligns with the observation from the main paper.

Synthesized vs. Ground Truth Images:
To investigate this further, Figure 2, shows the comparison between target

Table 1. Ratio of time the car remains within its driving lane for different model configurations

Method	Evaluation Result
VO + Synthesized [Ours]	0.9441
GT Trajectory + GT Images	0.9385
VO + GT Images	0.9408
GT Trajectory + Synthesized Images	0.9305

ground truth images and the images synthesized from the source at the desired target locations. Note that the synthesized images visually appear similar to the ground truth target images. In fact, the differences between the 2 images tend to occur beyond the drivable regions of the road at the object boundaries of buildings, fences, poles etc. Such differences can be attributed to occlusion or bleeding edge artifacts [19] occurring between boundaries of objects at different depths. This suggests that as far as the task of lateral vehicle control is concerned, minor differences between the synthesized and ground truth images outside the drivable regions do not influence driving behaviour. This could explain why the performance of the models trained with synthesized images or ground truth images are similar. This offers the possibility to utilize synthesized images in the real world where additional ground truth images are not available or are difficult to obtain.

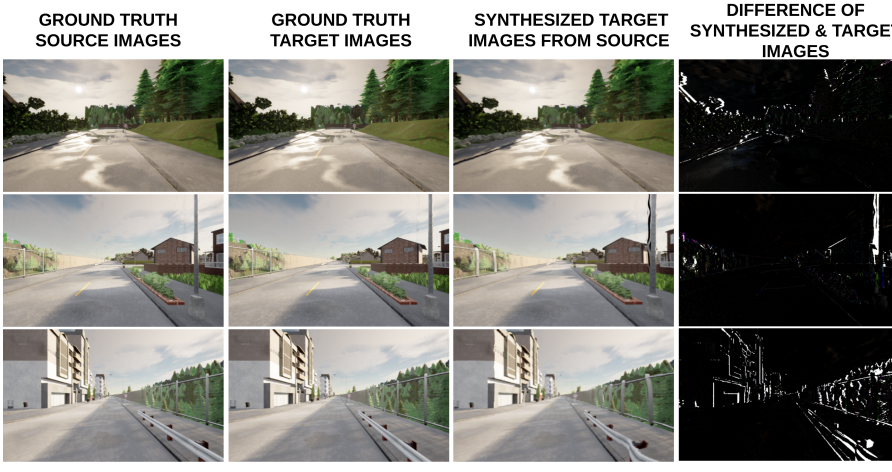

Fig. 2. Shows 3 examples of image synthesis on the CARLA dataset. The first column depicts the ground truth source images. The second column contains the ground truth target images. The third column are the images synthesized at the locations of the target images using the source images. The last column show the differences between the synthesized and target images. All images are of size 1000 x 600, and were center cropped from their native resolution of 1200 x 600.

Figure 3 shows a similar comparison on the KITTI dataset. Note that additional ground truth images are not available in the KITTI dataset but sequences from only a stereo pair. We therefore use our method to synthesize an image at the target location of the right image using the left image of the stereo pair as the source. We also observe here that the difference primarily occur at the object boundaries outside the drivable regions.

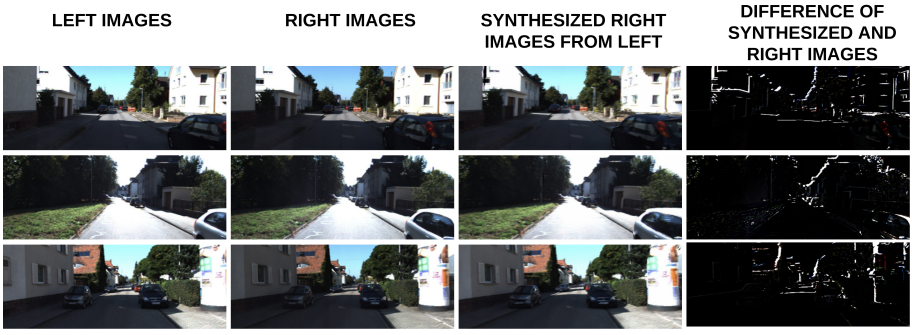

Fig. 3. Shows 3 examples of image synthesis on the KITTI dataset. The first column depicts the left images of the stereo pair. The second column contains the right images. The third column is the synthesized images at the locations of the right images using the left images as the source. The last column shows the differences between the synthesized and right images. All images are of size 1000 x 376, and were center cropped from their native resolution of 1241 x 376.

Visual Odometry vs. Ground Truth Trajectory:
Meanwhile, Figure 4 shows the comparison of the trajectory obtained by running visual odometry [9] with the ground truth trajectory for a small section of the road. As can be seen, the visual odometry trajectory aligns well with the ground truth until the turn. After the turn, it deviates slightly but remains parallel to the ground truth. However, note that when determining the target labels using MPC, only difference in state between the ego-vehicle and the goal state is used. Therefore, even though the two trajectories may not be aligned, the relative pose between 2 points on the 2 trajectories will nevertheless be very similar to each other. This seems to explain why the performance of our approach of training with visual odometry trajectory does not differ much from that trained with ground truth trajectory.

3 Offline Evaluation

[3] had conducted in depth studies on various offline metrics. They showed that the Mean Squared error (MSE) shows least correlation with the online evaluation. In fact, they demonstrated that 2 different models with the same MSE can

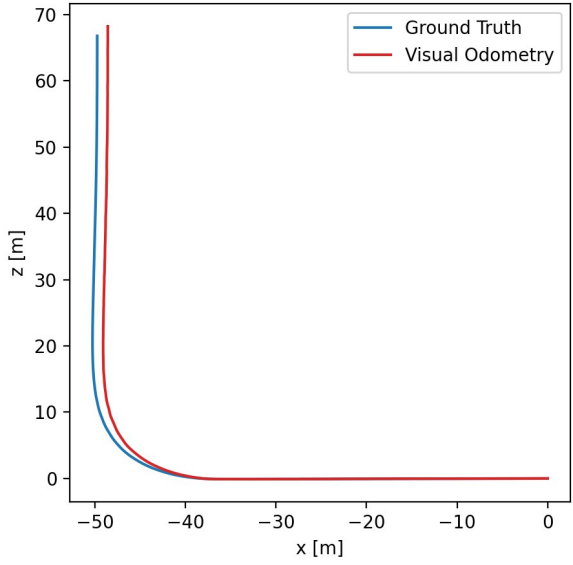

Fig. 4. Shows the trajectory generated by visual odometry. A comparison with the ground truth trajectory is also shown.

have drastically different driving behaviours. The closest correlation was shown by the Mean Absolute Error (MAE) but even this does not truly reflect the actual driving quality. Table 2 shows the MAE for our method and the single trajectory model when evaluated on the original on-course images.

Table 2. MAE metric for our method and the model trained with only a single trajectory. Lower is better.

	Train	Test
Our Method	0.1655	0.1813
Single Trajectory	0.1197	0.1017

Note that if the performance of the models were to be judged by the offline metrics then the single trajectory model would falsely purport to show better performance than our method. However, its true driving quality as measured by the online evaluation in Table 1 of the main paper is far inferior to our method.

As insinuated in the discussion section of the main paper (see Single Trajectory model), the reason for better offline metrics of the single trajectory model is because this model is trained only on the original on-course images. However, in an online evaluation, even a slight error in the prediction will cause the model to

diverge from the on-course trajectory causing it to be exposed to images outside of its training distribution. Hence, when such an out of distribution image is subsequently fed to the network, it makes further errors eventually causing the ego-vehicle to swerve off-course.

A better alternative would be to conduct the offline evaluation on not only the original trajectory data but also off-course data. Hence, when off-course images are fed to the single trajectory model, the network would give high errors in the offline evaluation and would correlate with the poor online performance. However, real world benchmarks do not contain off-course images as this would possibly involve violation of traffic rules during data collection. This is also the inspiration behind our framework on how to train a model with off course images synthesized from a single on-course trajectory.

Therefore, even if some of the popular real world datasets [5, 1, 13, 16, 15] provided steering commands, reporting the offline metrics for the various model configurations would be a meaningless comparison. This is because without an online evaluation, it is difficult to ascertain the true driving quality. The only real world dataset that we are aware of which provides steering labels is [6] but only for a single on-course trajectory. More importantly it does not provide a methodology to conduct an online evaluation on static image data.

4 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 [2].

Steering mechanism:
Note that in the bicycle model [14], 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 [12]. This difference can be maintained using the Ackermann steering mechanism [18]. For our purpose, CARLA already compensates for this difference in its steering mechanism [2].

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.

5 Training Details

We had used the architecture from [17] 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.

6 No-Slip condition

The slip (β) of a vehicle is the angle between its longitudinal axis and the velocity at its the center of gravity. If the slip is taken into consideration then the equations of motion describing the dynamics of a 4 wheel front drive vehicle having planar motion can be defined by[12]:

$$\dot{X} = V cos(\theta + \beta) \tag{2}$$

$$\dot{Y} = V sin(\theta + \beta) \tag{3}$$

$$\dot{\theta} = V \cos\beta \frac{\tan\delta}{L} \tag{4}$$

Where V, θ, X and Y describe the velocity, orientation and location coordinates of the vehicle.

Note that in the main paper, the equations of motion assumed the no-slip condition ($\beta = 0$). This holds true when the car is moving forwards or executing turns at low to moderate speed (5 meters per second). This is because at such speeds the lateral forces are low enough for the velocity vector for the tyres to be in alignment with their motion [12]. This causes the slip angle to be close to zero leading to the equations of motion described in the main paper.

Figure 5 shows the implications of increasing the throttle on the speed and driving performance of the ego-vehicle of our method when executing turns. Recall that the throttle values in CARLA can vary between [0,1]. A higher value implies a higher throttle. At a throttle of 0.6, the velocity of the vehicle falls within the value of 5 meters per second which abides by our no-slip assumption. In fact, the performance of our method is maintained to a velocity as high as 7.3 meters per second. This is a reasonable limit in urban environments wherein turns tend to be executed at such moderate speeds. However, if the velocity is increased any further by applying a greater throttle, the performance starts to deteriorate in proportion to the speed. Therefore, in order to cater for such scenarios it would be necessary to take the lateral forces exerted on the tyres and dynamics of the vehicle into consideration. Examining the lateral forces and dynamics is beyond the scope emphasized in this paper.

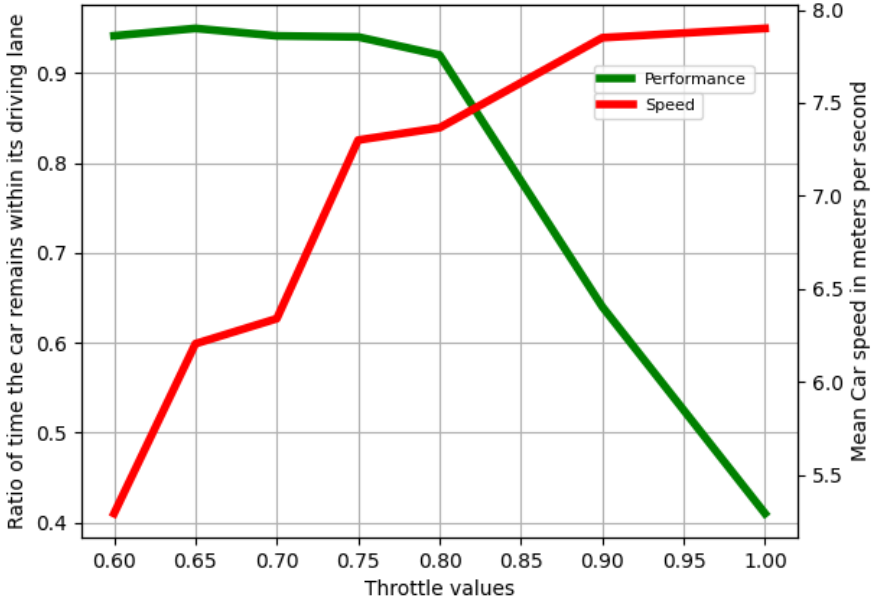

Fig. 5. Shows the implications of increasing the throttle on the mean speed of the car and the online performance of the our method. The online performance (left vertical axis) is reported as the ratio of time the car remains within its driving track. The speed of the car (right vertical axis) is reported in meters per second.

7 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 [17]. 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 [10]. But, they tend to be far more expensive [11] than RGB cameras, requiring more memory and a higher computational cost [8, 7] and can only produce sparse uncoloured point clouds.

8 Limitations of the Supervised model

We had already discussed some of the limitations of the supervised model in L543-L555 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 main paper, this off-course data is critical is 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.

9 Errata

There is a small typo error in the L165 of the main paper. The last reference should have been [15].

References

1. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 (2019)
2. CARLA: Carla simulator documents: Measurements [accessed on 07.03.2022]. https://carla.readthedocs.io/en/stable/measurements/
3. Codevilla, F., López, A.M., Koltun, V., Dosovitskiy, A.: On offline evaluation of vision-based driving models. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 236–251 (2018)
4. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Conference on Robot Learning (CoRL) (2017)
5. 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)
6. Geyer, J., Kassahun, Y., Mahmudi, M., Ricou, X., Durgesh, R., Chung, A.S., Hauswald, L., Pham, V.H., Mühlegg, M., Dorn, S., Fernandez, T., Jänicke, M., Mirashi, S., Savani, C., Sturm, M., Vorobiov, O., Oelker, M., Garreis, S., Schuberth, P.: A2D2: Audi Autonomous Driving Dataset (2020), https://www.a2d2.audi
7. Liu, Z., Amini, A., Zhu, S., Karaman, S., Han, S., Rus, D.: Efficient and Robust LiDAR-Based End-to-End Navigation. arXiv e-prints arXiv:2105.09932 (May 2021)
8. Liu, Z., Tang, H., Lin, Y., Han, S.: Point-voxel cnn for efficient 3d deep learning. In: Advances in Neural Information Processing Systems (2019)
9. 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
10. 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 (June 2021)
11. 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)
12. Rajamani, R.: Vehicle dynamics and control. In: Second Edition, Publisher: Springer (2012)
13. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
14. Wang, D., Q, F.: Trajectory planning for a four-wheel-steering vehicle. In: IEEE International Conference on Robotics and Automation (ICRA) (2001)
15. Wang, P., Huang, X., Cheng, X., Zhou, D., Geng, Q., Yang, R.: The apolloscape open dataset for autonomous driving and its application. IEEE transactions on pattern analysis and machine intelligence (2019)
16. Wenzel, P., Wang, R., Yang, N., Cheng, Q., Khan, Q., von Stumberg, L., Zeller, N., Cremers, D.: 4Seasons: A cross-season dataset for multi-weather SLAM in autonomous driving. In: Proceedings of the German Conference on Pattern Recognition (GCPR) (2020)

17. 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)
18. Zhao, J.S., Liu, Z.J., Dai, J.: Design of an ackermann type steering mechanism. Journal of Mechanical Engineering Science **227** (11 2013). https://doi.org/10.1177/0954406213475980
19. Zhu, S., Brazil, G., Liu, X.: The edge of depth: Explicit constraints between segmentation and depth. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)