Multimodal 3D Object Detection from Simulated Pretraining

Åsmund Brekke, Fredrik Vatsendvik(✉), and Frank Lindseth

Norwegian University of Science and Technology, Trondheim, Norway
{aasmunhb,fredrva}@stud.ntnu.no, frankl@ntnu.no

Abstract. The need for simulated data in autonomous driving applications has become increasingly important, both for validation of pretrained models and for training new models. In order for these models to generalize to real-world applications, it is critical that the underlying dataset contains a variety of driving scenarios and that simulated sensor readings closely mimics real-world sensors. We present the Carla Automated Dataset Extraction Tool (CADET), a novel tool for generating training data from the CARLA simulator to be used in autonomous driving research. The tool is able to export high-quality, synchronized LIDAR and camera data with object annotations, and offers configuration to accurately reflect a real-life sensor array. Furthermore, we use this tool to generate a dataset consisting of 10 000 samples and use this dataset in order to train the 3D object detection network AVOD-FPN, with finetuning on the KITTI dataset in order to evaluate the potential for effective pretraining. We also present two novel LIDAR feature map configurations in Bird's Eye View for use with AVOD-FPN that can be easily modified. These configurations are tested on the KITTI and CADET datasets in order to evaluate their performance as well as the usability of the simulated dataset for pretraining. Although insufficient to fully replace the use of real world data, and generally not able to exceed the performance of systems fully trained on real data, our results indicate that simulated data can considerably reduce the amount of training on real data required to achieve satisfactory levels of accuracy.

Keywords: Autonomous driving · Simulated data · 3D object detection · CARLA · KITTI · AVOD-FPN · LIDAR · Sensor fusion

1 Introduction

Machine learning models are becoming increasingly complex, with deeper architectures and a rapid increase in the number of parameters. The expressive power of such models allow for more possibilities than ever before, but require large amounts of labeled data to properly train. Labeling of data in the autonomous driving domain requires extensive amounts of manual labour, either in the form of actively producing annotations such as class labels, bounding boxes and semantic segmentation by hand, or by supervising and adjusting automated generation of these using a pretrained ensemble of models from previously labeled data. For the use of modern sensors such as LIDAR, not many sizable labeled datasets exists, and those that do generally offer little variation in terms of environments or weather conditions to properly allow for generalization to real world conditions. Popular datasets such as KITTI [2] offers a large array of sensors, but with largely unchanging weather conditions and lighting, while the larger and more diverse BDD100K [3] dataset does not include multimodal sensor data, only offering camera and GPS/IMU. The possibility of new sensors being introduced that greatly impact autonomous driving also carry the risk of invalidating the use of existing datasets for training state-of-the-art solutions.

1.1 Simulated Data for Autonomous Driving

With the advances in recent years in the field of computer graphics, both in terms of photorealism and accelerated computation, simulation has been a vital method of validating autonomous models in unseen environments due to the efficiency of generating different scenarios [13]. More recently there has been added interest in the use of modern simulators for also generating the data used to train models for autonomous vehicles, both for perception and end-to-end reinforcement learning [10–12]. There are several advantages in generating training data through simulation. Large datasets, with diverse conditions can be quickly generated provided enough computational resources, while labeling can be fully automated with little need for supervision. Specific, difficult scenarios can be more easily constructed and advanced sensors can be added provided they have been accurately modeled. Systems such as the NVIDIA Drive Constellation [5] are pushing the boundaries for photorealistic simulation for autonomous driving using clusters of powerful NVIDIA GPUs, but is currently only available to automakers, startups and selected research institutions using NVIDIAs Drive Pegasus AI car computer, and only offers validation of models, not data generation for training. However, open source solutions based on state-of-the-art game engines such as Unreal Engine 4 and Unity, are currently in active development and offer a range of features enabling anyone to generate high quality simulations for autonomous driving. Notable examples include CARLA [1] and AirSim [4], the former of which was used for this research.

2 Simulation Toolkit

In order to facilitate the training and validation of machine learning models for autonomous driving using simulated data, the authors introduce the Carla Automated Dataset Extraction Tool (CADET) [8], an open-source tool for generating labeled data for autonomous driving models, compatible with Carla 0.8. The tool supports various functionality including LIDAR to camera projection (Fig. 1), generation of 2D and 3D bounding box labels for cars and pedestrians (Fig. 2), detection of partially occluded objects (Fig. 3), and generation of sensor data including LIDAR, camera and ground plane estimation, as well as sensor calibration matrices. All labels and calibration matrices are stored in the data format defined by Geiger et al. [2], which makes it compatible with a number of existing models for object detection and segmentation. As a varied dataset is crucial for a machine learning model to generalize from a simulated environment to real-life scenarios, the data generation tool includes a number of measures to ensure variety. Most importantly, the tool resets the environment after a fixed number of samples generated. Here, a sample is defined as the tuple containing a reading from each sensor, corresponding ground truth labels and calibration data. Resetting the environment entails randomization of vehicle models, spawn positions, weather conditions and maps, and ensures a uniform distribution of weather types, agent models for both pedestrians and cars and starting positions of all vehicles. The LIDAR and camera sensors are positioned identically and synchronized such that a full LIDAR rotation exists for each image[1]. The raw sensor data is projected to a unified coordinate system used in Unreal Engine 4 before determining visible objects in the scene, and projecting to the relative coordinate spaces used in KITTI. As the initial LIDAR configuration in CARLA ignores the pitch and roll of the vehicle it is attached to, additional transformations are applied after projection such that the sensor data is properly aligned. One challenge when generating object labels is determining the visible objects in the current scene. In order to detect occluded objects, the CARLA *depth map* is utilized. A vertex is defined as occluded if the value of one of its neighbouring pixels in the depth map is closer than the vertex distance to the camera. An object is defined as occluded if at least four out of its eight bounding box vertices are occluded. This occlusion detection performs satisfactory and much faster than tracing the whole object, even when objects are localized behind see-through objects such as chain-link fences, shown in Fig. 3. A more robust occlusion detection can be performed by using the semantic segmentation of the scene, but this is not implemented as of yet.

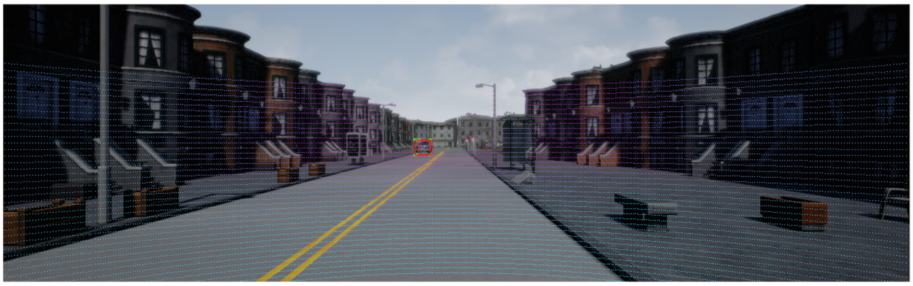

Fig. 1. LIDAR point cloud projected to image space. The color of each LIDAR point is determined by its depth value (Color figure online).

[1] Note that this only implies an approximate correspondance between points from the camera and LIDAR sensors.

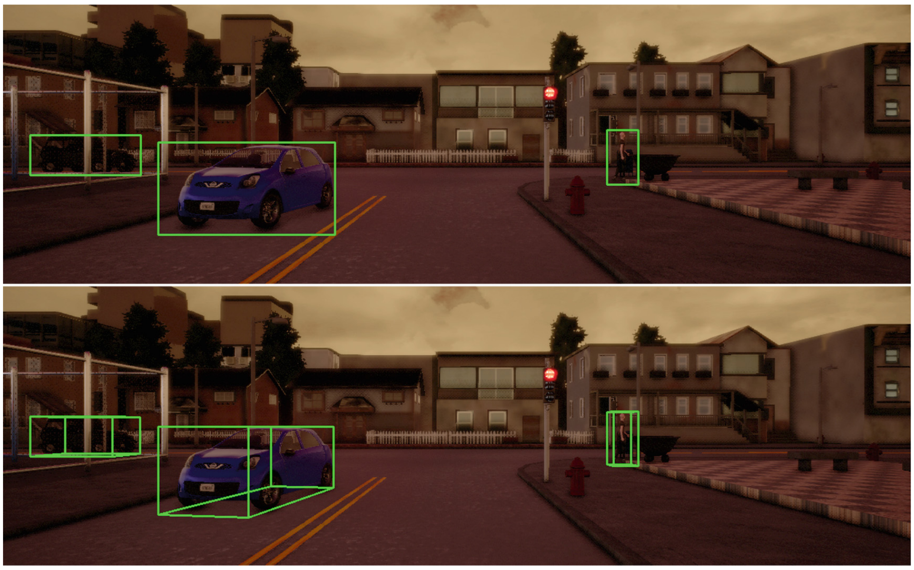

Fig. 2. 2D (top) and 3D (bottom) bounding boxes as generated by CADET. Class labels are omitted.

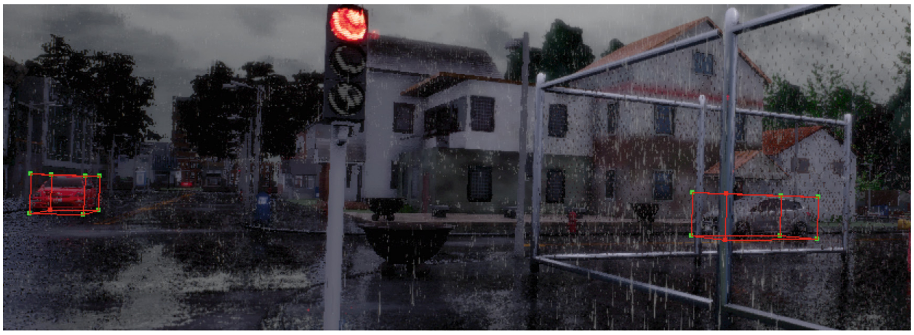

Fig. 3. Occluded vertices behind a chain fence. Note that both cars are visible, and thus have a bounding box drawn around them. Occluded and visible vertices are drawn with red and green, respectively (Color figure online).

3 Generated Dataset

Using CADET we generate the CADET dataset, consisting of 10 000 samples. In total, there are 13989 cars and 4895 pedestrians in the dataset, averaging about 1.9 labeled objects per image. The dataset contains 2D and 3D bounding box annotations of the classes Car and Pedestrian, and contains both LIDAR and camera sensor data, as well as ground plane estimation and generation of sensor calibration matrices. The environment is generated from two maps, namely *Town01* and *Town02* in the CARLA simulator, which are both suburban environments. The distribution of objects in each image is shown in Fig. 8. In comparison with the KITTI dataset, the CADET dataset has less cars and pedestrians per image, which is mostly due to city-environment of KITTI, where cars are frequently parked at the side of the road and pedestrians are present in a higher degree. The orientation of each labeled object is shown in Fig. 9. We observe that the distribution of orientation have a sharp multimodal distribution with three peaks, namely for objects seen from the front, behind or sideways. Note that the pedestrians in the dataset generally have a smaller bounding box than cars, shown in Fig. 7, making them harder to detect.

4 Training on Simulated Data

In order to evaluate the use of the simulated CADET dataset, as well experiment with LIDAR feature map representations, several configurations were used of the AVOD-FPN [6] architecture for 3D object detection using camera and LIDAR point cloud. The AVOD-FPN source code has been altered to allow for customized configurations by specifying the features wanted for two groups, slice maps and cloud maps. Slice maps refer to feature maps taken from each vertical slice the point cloud is split into, as specified in the configuration files, while the cloud maps consider the whole point cloud. Following the approach described in [6], two networks were used for detecting cars and pedestrians separately, repeating the process for each configuration. As multiclass detection might also produce more unstable results when evaluating per class, this was considered the better option. All models used a feature pyramid network to extract features from images and LIDAR, with early fusion of the extracted camera and LIDAR features. Training data is augmented using flipping and jitter, with the only differences between models of the same class being the respective representations of LIDAR feature maps in Bird's Eye View (BEV) as described in Sect. 4.1. All configurations used are available in the source code [9].

4.1 Model Configurations

AVOD-FPN uses a simplified feature extractor based on the VGG-16 architecture [14] to produce feature maps from camera view as well as LIDAR projected to BEV, allowing the LIDAR to be processed by a Convolutional Neural Network (CNN) designed for 2D images. These separate feature maps are fused together using trainable weights, allowing the model to learn how to best combine multimodal information. In addition to what will be referred to as the default BEV configuration, as proposed in [6], two additional novel configurations are proposed for which experimental results either show faster inference with similar accuracy, or better accuracy with similar inference speed. In all cases the BEV is discretized horizontally into cells at a resolution of 0.1m. The default configuration creates 5 equally sized vertical slices within a specified height range, taking the highest point in each cell normalized by the slice height. A separate image for the density of the entire point cloud is generated from the number of points N in each cell following Eq. 1, as used in [6] and [7], though normalized by $log(64)$ in the latter. We propose a simplified structure, taking the global maximum height, minimum height and density of each cell over the entire point cloud, avoiding the use of slices and halving the amount of BEV maps. We argue that this is sufficient to determine which points belong to large objects and which are outliers, and that it sufficiently defines box dimensions. For classes that occupy less space, we argue that taking three slices vertically using the maximum height and density for each slice can perform better with less susceptibility to noise, as the network could potentially learn to distinguish whether the maximum height value of a slice belongs to the object or not depending on the slice density. All configurations are visualized in Figs. 4, 5 and 6.

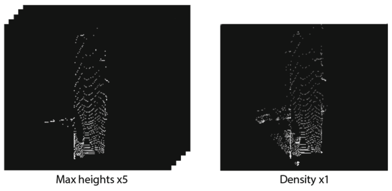

Fig. 4. Visualization of default BEV configuration, taking the maximum height within 5 vertical slices as well as the density of the full point cloud.

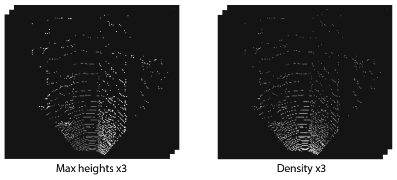

Fig. 5. Visualization of first custom BEV configuration, taking the maximum height and density within 3 vertical slices.

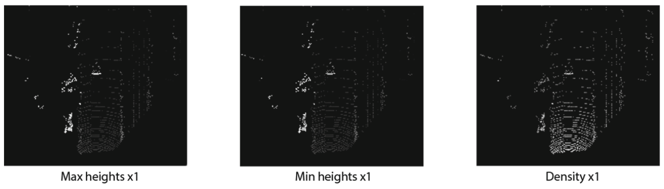

Fig. 6. Visualization of second custom BEV configuration, taking the maximum height, minimum height and density of the full point cloud.

$$min(1.0, \frac{log(N+1)}{log(16)}) \tag{1}$$

4.2 Results

In order to gather qualitative results each model trained for a total of 120k steps on the respective datasets, with a batch size of 1, as described in [6]. Checkpoints were stored at every 2k steps, of which the last 20 were selected for evaluation.

Tables 1 and 2 show generated results on the KITTI dataset, for the Car and Pedestrian classes respectively, selecting the best performing checkpoint for each of the 3 BEV configurations. To measure inference speed, each model performs inference on the first 2000 images of the validation set, with learning deactivated, using a NVIDIA GTX 1080 graphics card. The mean inference time is rounded up to the nearest millisecond and presented in the tables.

Table 1. KITTI-trained model evaluated on the KITTI dataset for the Car class

Method	Runtime (ms)	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
		Easy	Moderate	Hard	Easy	Moderate	Hard
Default	119	**83.46**	73.94	67.81	89.37	86.44	78.64
Max*3, Density*3	120	83.16	**73.97**	**67.98**	**89.84**	**86.62**	**79.85**
Max, Min, Density	**114**	82.98	73.92	67.84	89.62	86.61	79.68

Table 2. KITTI-trained model evaluated on the KITTI dataset for the Pedestrian class

Method	Runtime (ms)	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
		Easy	Moderate	Hard	Easy	Moderate	Hard
Default	122	41.05	37.00	32.00	44.12	39.54	38.11
Max*3, Density*3	122	**45.61**	**42.66**	**38.06**	**49.16**	**45.99**	**44.53**
Max, Min, Density	**117**	27.85	27.17	24.54	33.39	33.10	29.78

Following evaluation on the KITTI dataset, all configurations were trained from scratch on the generated CADET dataset following the exact same process. Results from evaluation on the validation set of the CADET dataset can be seen in Tables 3 and 4. Note that as dynamic occlusion and truncation measurements are not included in the dataset (these are only used for post training evaluation in KITTI), evaluation does not follow the regular easy, moderate, hard categories used in KITTI. Instead objects are categorized as large or small, following the minimum height requirements for the bounding boxes of 40 pixels for easy and 25 pixels for moderate and hard. These models were additionally evaluated directly on the KITTI validation set, with results summarized in Tables 5 and 6.

Table 3. CADET-trained model evaluated on the CADET dataset for the Car class

Method	$AP_{3D}(\%)$		$AP_{BEV}(\%)$	
	Large	Small	Large	Small
Default	70.86	69.37	**80.13**	**71.32**
Max*3, Density*3	**70.96**	**69.59**	79.81	71.28
Max, Min, Density	68.79	60.87	78.75	70.72

Table 4. CADET-trained model evaluated on the CADET dataset for the Pedestrian class

Method	$AP_{3D}(\%)$		$AP_{BEV}(\%)$	
	Large	Small	Large	Small
Default	75.43	**73.89**	75.43	**73.91**
Max*3, Density*3	**76.13**	72.99	**80.24**	73.41
Max, Min, Density	75.49	71.82	79.73	72.37

Table 5. CADET-trained model evaluated on the KITTI dataset for the Car class

Method	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Default	29.85	20.29	18.40	50.58	37.81	30.77
Max*3, Density*3	**35.85**	**29.40**	**24.99**	**57.32**	**49.63**	**43.25**
Max, Min, Density	30.34	24.28	20.25	45.22	37.56	31.17

Table 6. CADET-trained model evaluated on the KITTI dataset for the Pedestrian class

Method	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Default	**9.09**	**9.09**	**9.09**	9.38	**9.33**	**9.38**
Max*3, Density*3	2.27	2.27	2.27	2.27	2.27	2.27
Max, Min, Density	**9.09**	**9.09**	**9.09**	**9.78**	9.09	9.09

Table 7. CADET-trained model, fine-tuned and evaluated on the KITTI dataset for the Car class

Method	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Default	**83.84**	68.67	**67.40**	**89.41**	79.77	**78.86**
Max*3, Density*3	76.85	**72.44**	66.55	88.20	**85.18**	78.71
Max, Min, Density	81.00	66.95	65.88	88.76	79.36	78.41

The CADET-trained models were subsequently restored from their checkpoints at step 90k and modified for further training on the KITTI dataset. Training was resumed until step 150k, meaning the models received 60k steps of training on the KITTI training set as opposed to 120k originally. Other than increasing the amount of steps and switching the target datasets,

Table 8. CADET-trained model, fine-tuned and evaluated on the KITTI dataset for the Pedestrian class

Method	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Default	**40.26**	**38.55**	33.93	**46.96**	**44.80**	**40.73**
Max*3, Density*3	39.19	38.02	**34.15**	46.14	43.54	40.44
Max, Min, Density	37.32	34.34	32.60	45.71	42.43	37.62

the configuration files were not altered from when training on the CADET dataset. Tables 7 and 8 show results from the top performing checkpoint of each model.

5 Discussion

For the fully KITTI-trained models, results on the Car class are very similar for all configurations, where the largest loss in amounts to only 0.5% 3D AP on the easy category from the default configuration to our configuration using half as many layers in the BEV map. Larger differences are apparent for the Pedestrian class, where 3 layers is not sufficient to compete with the default configuration. However, the use of 3 slices of maximum heights and density, totalling 6 layers as with the default configuration, shows noticeably better results across the board suggesting a more robust behaviour.

Evaluation of the CADET-trained models on the CADET validation set shows similar relative performance between the models, however with the simpler custom configuration showing a dip in accuracy for the moderate category on the Car class. With regards to pedestrians, differences are much smaller than what could have been expected. Also considering the unremarkable and rather inconsistent performance on the Pedestrian class of the KITTI dataset, we can likely accredit this to overly simplified representation of the physical collision of pedestrians visible in the simulated LIDAR point cloud. The CADET-trained models do perform better on the Car class however, suggesting better and more consistent generalization for this task.

The fine-tuned models show performance on the Car class mostly similar to the fully KITTI-trained models, however with each model showing a noticeable drop in performance on either the easy or moderate category. Results on the Pedestrian class are a bit more interesting. The default configuration sees a slight increase in accuracy on the moderate and hard category, with a slight decrease for easy. The max/density configuration sees a significant decrease in performance on all categories, where as the less complex max/min/density configuration, although still being the weakest performer, sees a significant increase in performance compared to when only trained on the KITTI dataset. The reason for the rather inconsistent results when compared to the KITTI-trained models are not thoroughly investigated, but can in part be due to somewhat unstable

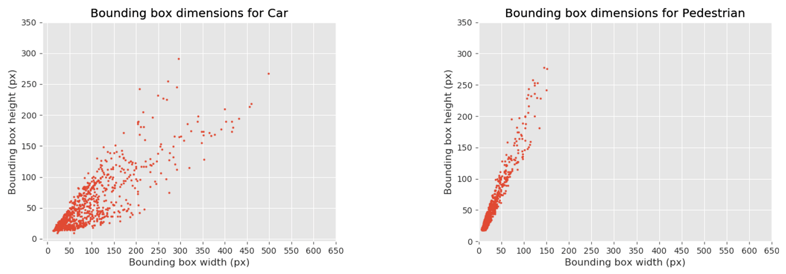

Fig. 7. Dimension of 2D bounding boxes for the classes in the CADET dataset.

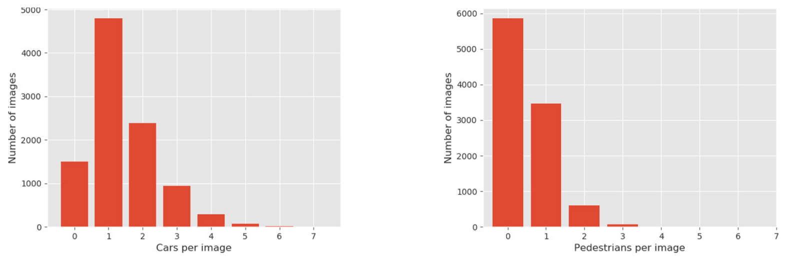

Fig. 8. Number of annotations for each class per image in the CADET dataset.

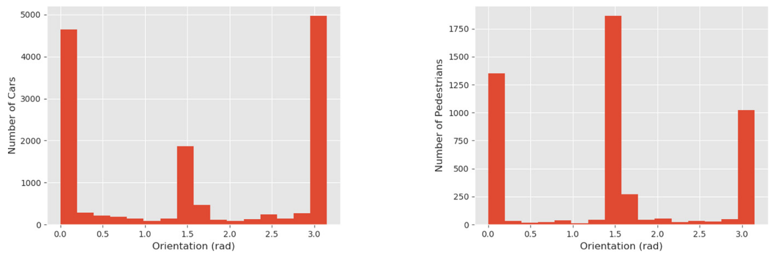

Fig. 9. Distribution of orientation per class in the CADET dataset.

gradients not producing fully reliable results. The CARLA generated LIDAR point cloud does not feature accurate geometry due to simplified collision of all dynamic objects. As such the different capabilities of the configurations may not be exploited during the pretraining on the CADET dataset, impacting overall results. The simplest configuration significantly closing the gap on the Pedestrian class may be a testament to this, as the simplified pedestrian representation is more easily recognizable.

While results from simulated and partly simulated training do not generally exceed the performance of direct training on the dataset, there is a clear indication that the use of simulated data can achieve closely matched performance with less training on actual data. The ease of generation and expandabililty in terms of sensors, scenarios, environments and conditions makes tools such as CADET very useful for training and evaluating models for autonomous driving, although improvements are needed before they are sufficient for training real world solutions.

6 Conclusion

In recent years, the use of synthetic data for training machine learning models has gained in popularity due to the costs associated with gathering real-life data. This is especially true with regards to autonomous driving because of the strict demand of generalizability to a diverse number of driving scenarios. In this study, we have described CADET - a tool for generating large amounts of training data for perception in autonomous driving, and the resulting dataset. We have demonstrated that this dataset, while not sufficient to directly train systems for use in the real world, is useful in lowering the amount of real-life data required to train machine learning models to reasonably high levels of accuracy. We have also suggested and evaluated two novel BEV representations, easily configurable before training, with potential for better detection of smaller objects and reduced complexity for detection of larger objects respectively. The CADET toolkit, while still requiring improved physical models in LIDAR modelling, is currently able to generate datasets for training and validation of virtually any model designed for the KITTI object detection task.

References

1. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: an open urban driving simulator. In: Conference on Robot Learning, pp. 1–16 (2017)
2. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the KITTI dataset. Int. J. Robot. Res. **32**, 1231–1237 (2013)
3. Yu, F., et al.: BDD100K: A diverse driving video database with scalable annotation tooling. arXiv preprint arXiv:1805.04687 (2018)
4. Shah, S., Dey, D., Lovett, C., Kapoor, A.: AirSim: high-fidelity visual and physical simulation for autonomous vehicles. In: Hutter, M., Siegwart, R. (eds.) Field and Service Robotics. SPAR, vol. 5, pp. 621–635. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67361-5_40
5. NVIDIA Drive Constellation Homepage (2019). https://www.nvidia.com/en-us/self-driving-cars/drive-constellation. Accessed 8 Apr 2019
6. Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.L.: Joint 3D proposal generation and object detection from view aggregation. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1–8. IEEE (2018)
7. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3D object detection network for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1907–1915 (2017)
8. Brekke, Å., Vatsendvik, F.: CARLA data export tool (2019). https://github.com/Ozzyz/carla-data-export/

9. Modified AVOD architecture (2019). https://github.com/Fredrik00/avod. Accessed 8 Apr 2019
10. Kiran, B.R., Roldão, L., Irastorza, B., Verastegui, R., Süss, S., Yogamani, S., Talpaert, V., Lepoutre, A., Trehard, G.: Real-time dynamic object detection for autonomous driving using prior 3D-maps. In: Leal-Taixé, L., Roth, S. (eds.) ECCV 2018. LNCS, vol. 11133, pp. 567–582. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11021-5_35
11. Codevilla, F., et al.: End-to-end driving via conditional imitation learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–9. IEEE (2018)
12. Gaidon, A., Lopez, A., Perronnin, F.: The reasonable effectiveness of synthetic visual data. Int. J. Comput. Vision **126**(9), 899–901 (2018)
13. Schöner, H.P.: Simulation in development and testing of autonomous vehicles. In: Bargende, M., Reuss, H.C., Wiedemann, J. (eds.) Internationales Stuttgarter Symposium. P, pp. 1083–1095. Springer, Wiesbaden (2018). https://doi.org/10.1007/978-3-658-21194-3_82
14. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)