AD PerDevKit: An Autonomous Driving Perception Development Kit using CARLA simulator and ROS

Javier de la Peña[1], Luis M. Bergasa[1], Miguel Antunes[1], Felipe Arango[1], Carlos Gómez-Huélamo[1], Elena López-Guillén[1]

***Abstract*— Every now and then a new dataset appears that aims to help training computer vision models for autonomous vehicles (AVs), being KITTI [1] the most famous, but others like Waymo [2] or nuScenes [3] can be found that try to innovate by adding more sensors, giving more data about the ego-vehicle surroundings or offering new scenarios. These approaches tend to produce a high quality dataset, but at the cost of time and money. The key difficulty is to create datasets with enough data to generalize multiple road scenarios and hazard use-cases in the training process of perception models for AVs.**

To address this concern, we propose AD PerDevKit, a perception development kit for autonomous driving based on CARLA simulator [4] able to generate an infinite set of annotated data referred to the ego-vehicle by recording virtual car sensors (camera, LiDAR, Radar) in virtual environments. Our ground-truth generator provides real-time information of the objects surrounding the vehicle that are visible for each sensor by publishing them on a ROS [5] topic. In addition, a dataset generated with this tool is also presented for some specific challenging scenarios and an evaluation of its contribution in the improvement of 2D object detection methods on real data is included.

The code of our tool and its database is available in the following git repository: `https://github.com/Javier-DlaP/ad_perdevkit`.

keywords: autonomous vehicles, dataset, simulation, ROS, CARLA.

I. INTRODUCTION

In recent years, thanks to improvements in sensors, computational processing, Deep Learning (DL), and communication techniques, we are in a race to get the first fully autonomous vehicle in the market. Automotive and technology companies such as ArgoAI, Audi, Baidu, Cruise, Mercedes-Benz, Tesla, Uber, Waymo and others are investing huge amounts of money in the development of these technologies.

To achieve autonomous driving systems (ADS), a good knowledge of the surroundings is required. In order to build a robust driving system, multiple sensors are used to provide information to the vehicle, such as cameras, LiDAR or Radar. ADS/ADAS systems require a good understanding of the surroundings in order to work properly, so it is necessary to add several sensors along the vehicle that allow to obtain as much information as possible.

Different multi-sensors techniques have been reported in the design of perception systems on-board a vehicle. In last years, DL models, which require large amounts of data in their learning process, have gained weight.

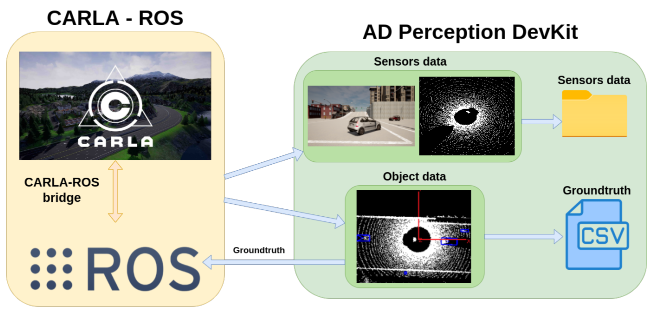

Fig. 1: AD PerDevKit pipeline.

The use of perceptual datasets consisting of real sensor data for training DL models is widespread. With the appearance of KITTI in 2012, this dataset began to be standardized for model training and evaluation, since it was the pioneer in the integration of multiple sensors and multiple evaluation systems within the same dataset. This breakthrough led to the creation of other datasets such as Argoverse [6], Lyft L5 [7], Waymo [2] or nuScenes [3] among others.

DL models, usually trained in a supervised way, need to use huge datasets with multiple scenarios and hazard situations to generalize traffic use cases that are found on the roads. These data need to be manually labeled, which is a time consuming and expensive activity. To alleviate this hard task, we propose to use a development kit using CARLA able to generate synthetic annotated data recorded by virtual sensors in virtual scenarios in simulation time using ROS [5]. This process is outlined in Fig. 1. With this development kit, almost unlimited data samples can be created offering the possibility to generate hazard use cases in a safety way and at a low cost.

In addition, a dataset with different challenging scenarios is provided as an example of what can be achieved with this tool. Furthermore, this dataset can be used in combination with other real data to increase the accuracy of the perception models to be built. As a concept proof of its potential we include an evaluation of the improvements obtained with a popular 2D object detection method.

This work has been funded in part from the Spanish MICINN/FEDER through the Artificial Intelligence based modular Architecture Implementation and Validation for Autonomous Driving (AIVATAR) project (PID2021-126623OB-I00) and from the RoboCity2030-DIH-CM project (P2018/NMT-4331), funded by Programas de actividades I+D (CAM), cofunded by EU Structural Funds and Scholarship for Introduction to Research activity by University of Alcalá

[1]Javier de la Peña, Luis M. Bergasa, Miguel Antunes, Felipe Arango, Carlos Gómez-Huélamo, Elena López-Guillén are with the Electronics Department, University of Alcalá (UAH), Spain. j.pena@edu.uah.es, luism.bergasa@uah.es, miguel.antunes@edu.uah.es, juanfelipe.arango@uah.es, carlos.gomezh@uah.es, elena.lopezg@uah.es

TABLE I: Comparison of cited datasets.

Dataset	Type of data	Cities	Annotated frames	Ground-truth 360º	Camera	LiDAR	Radar	Night/Rain	**GT generation tool**
KITTI	Real data	1	15k	✗	✓	✓	✗	✗	✗
nuScenes	Real data	2	40k	✓	✓	✓	✓	✓	✗
AIODrive	Synthetic data	8	100k	✓	✓	✓	✓	✓	✗
AD DevKit	Synthetic data	5	21k	✓	✓	✓	✓	✓	✓

II. RELATED WORK

In recent years, many datasets have been created to train and evaluate motion detection, tracking and forecasting models. Among all the autonomous driving datasets used, KITTI and nuScenes stand out as references in the creation of datasets with real data. On the other hand, AIODrive [8] is one of the main exponents of the synthetic data sets generated in the CARLA simulator.

Table I shows the main characteristics of all the above-mentioned datasets in order to understand more easily the differences among them. As we can see, our proposal is the only one that offers a free GT generation tool.

We would like highlight that the role of our dataset is not to replay real datasets. Instead, it can be used in combination with real data to pre-train detectors to improve detection performance on real data, as it will be shown in the dataset section.

A. KITTI

In 2012, the Karlsruhe Institute of Technology (KIT) provided KITTI, one of the first datasets specifically for autonomous driving, which offered front camera data in addition to point cloud data, with its corresponding ground-truth. As ground-truth, KITTI provides 2D and 3D annotated bounding boxes for use in detection and tracking tasks, in addition to location information for SLAM and visual odometry tasks. All this data was collected by a vehicle with 2 stereo cameras, 1 Velodyne HDL-64E LiDAR and a location system based on GPS, GNSS, IMU and RTK.

B. nuScenes

In 2019, nuTonomy company presented nuScenes, a dataset with a large number of different scenes. This dataset makes use of a vehicle with a 360-degree vision system using 6 cameras with 1600x900 resolution, a LiDAR of 32 beams at 20 Hz, a 5 Radar system with a maximum distance of 250 meters and a localization system composed of GPS, IMU, AHRS and an RTK positioning system. This dataset not only offers a wide variety of sensors but also provides the necessary ground-truth for detection, tracking, semantic segmentation and point cloud segmentation tasks, all annotated using 23 different categories. All this makes it one of the most complete dataset available today.

C. AIODrive

While the previous two datasets were performed in real-world environments, using synthetic data can be useful, as they allow the generation of data in a simpler way and with more complex and dangerous scenarios. Within synthetic datasets, AIODrive is the main one. This dataset built on CARLA uses 5 cameras to obtain a 360 degree view of the environment, 4 Radar, 3 Lidar and an IMU/GPS. The dataset is built with a large amount of data in a simple way, since all the necessary ground-truth is autonomously generated avoiding the costly manual labeling process.

The AIODrive dataset is open-source, but the tool used to create it has not been published yet. Because of this and the lack of documentation provided by the developers, it is difficult to manage correctly this dataset. On top of that, it does not solve the calculation of the visibility of objects from the vehicle, so it is not possible to distinguish whether the objects in the environment are visible from the vehicle itself or not.

D. Domain gap between synthetic and real data

Though synthetic data generation can be used to create a comprehensive dataset, one might argue that the domain gap between synthetic and real data is a weakness [8]. We believe that the usefulness of synthetic datasets is firmly predicated on many prior work [8],[9],[10],[11] that have shown, it can be used to enhance perception performance on real data when synthetic data is correctly used. In the same way that the success of prior works with synthetic data, we believe that the usefulness of our dataset in combination with real data is also undoubted, as will be proved in section VI.

III. CARLA AUTONOMOUS DRIVING SIMULATOR

The development of an autonomous vehicle, entails an exhaustive testing process in multiple simulated scenarios populated with vehicles and people to resemble the real situations that the vehicle may encounter, as a previous step for its validation in the real world. For this purpose, in recent years, several simulators have been designed. Among them we find Sim4CV [12] or Nvidia Drive [13]. These simulators are not open-source, so research community does not have free access to them, hindering a standard of comparison in the creation of validation scenarios, the modelling of virtual sensors or the evaluation of different weather conditions. CARLA, is an open-source project that offers a large number of facilities in the development of AVs. This is why we have based our development kit in this simulator, contributing to generate an standard in the AD validation process, despite of the fact that CARLA uses sensor models that are simplified to reduce its load in real time.

This simulator provides virtual environments to develop and validate AD techniques, by using the API programming in Python and C++. CARLA is based on the Unreal Engine [14] to run the world, and uses the OpenDRIVE [15] standard for the definition of roads and the environment.

It uses a client-server model in its architecture, so the server is responsible for the entire simulation: rendering sensors, calculating physics, updating the world, etc. On the

TABLE II: Format of the created ROS message.

Type	Name	Description
string	type	Object type (car, pedestrian, truck, etc)
uint32	object_id	Id assign to an object
float32	alpha	Observation angle of object
vision_msgs/BoundingBox2D	bounding_box_2D	Object 2D bounding box
geometry_msgs/Point	position	Center of the 3D object
geometry_msgs/Vector3	dimensions	Object dimensions (l, w, h)
float32	rotation_z	Object rotation along z axis
geometry_msgs/Vector3	velocity	Velocity of the object related to the ego vehicle
float32	truncated	Float from 0 (non-truncated) to 1 (truncated), where truncated refers to the object leaving image boundaries
uint8	occluded	Integer (0,1,2,3) indicating occlusion state: 0 = fully visible, 1 = partly occluded, 2 = largely occluded, 3 = not visible to the camera

other hand, the client controls the logic of the actors in the scene and changes the world options. The communication to the server is done through the API provided by CARLA, but in our case we use the CARLA-ROS bridge for the communication. With this configuration, it is possible to run our tool without the need to run CARLA simultaneously, since only ROS is needed.

IV. GROUND-TRUTH GENERATION TOOL

The main contribution of our AD PerDevKit is the creation of a ground-truth generation tool for the surrounding obstacles of the ego-vehicle using CARLA and ROS. For its implementation, as explained in section III, the CARLA-ROS brige is used so that the simultaneous execution of CARLA and this tool is not necessary, since the execution of both programs can be very demanding due to the simulator requirements. This way it is possible to record a rosbag (a file with all the ROS messages) with the GT information, so that only an area around the ego-vehicle is analyzed and not the whole obstacles in the CARLA runtime.

The messages created by CARLA contain the information of the different objects of the environment in relation to the map on which it is being used. However, to be used independently, the obstacles must be referenced to the ego-vehicle. Therefore, it is necessary to perform the different transformations to go from a coordinate system based on the map to a coordinate system based on the ego-vehicle.

A. *Method for obtaining 2D bounding boxes*

While the CARLA-ROS bridge gives access to the 3D bounding boxes of the objects in the environment, it does not provide the projected 2D bounding boxes of these 3D objects in the camera.

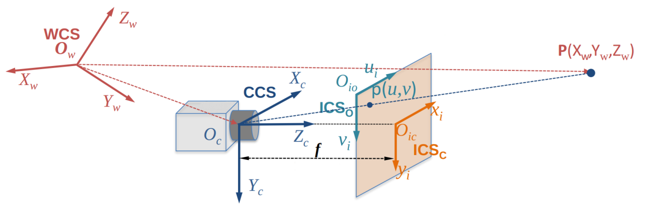

Fig. 2: Geometry transformation from world to camera.

A conversion from the world coordinates to the image ones is necessary to transform the eight vertices of a 3D bounding box into the vertices of a 2D bounding box. For this purpose, it is necessary to apply some geometry transformations as shown in Fig. 2.

Using homogeneous coordinates, the correspondence between a 3D point in the World Coordinate System (WCS) and its projected pixel in the Image Coordinate System at origin (ICS$_O$) is calculated applying the following equation:

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = M_{3x4} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

where $M_{(3x4)} = M_{int} M_{ext}$ is the camera projection matrix, formed by the product of intrinsic matrix and extrinsic matrix, defined as follow:

$$M_{int} = \begin{bmatrix} f/dx & 0 & u_0 \\ 0 & f/dy & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_{ext} = \begin{bmatrix} R_{3x3} & T_{3x1} \\ 0 & 1 \end{bmatrix}$$

being f focal distance, (dx, dy) physical size of the pixel, (u_0, v_0) centre of the image in pixels, T_{3x1} translation matrix and R_{3x3} rotation matrix from the world (WCS) to the camera (CCS).

B. *Calculation of object visibility*

One main issue for the GT calculation is that CARLA always render all the objects, even when they are not visible by the camera, LiDAR or Radar, which is a problem for any training process. There are many studies about ray tracing that solve this issue. These techniques [16] [17] are computationally very expensive so it is very difficult to implement them in real-time.

We propose a method for the calculation of visibility in CARLA and ROS using directly the point cloud calculated by CARLA. A vehicle will be considered as visible, as long as a point of the LiDAR point cloud is found inside an object, in the same way as it is done in the nuScenes dataset [3]. The steps to be performed are the following:

1) Remove objects furthest than the maximum LiDAR distance.
2) Deletion of the points of the point cloud with the height higher or lower than the objects in the surroundings.

3) Elimination of points outside the area in which the objects are located, taking into account all possible rotations.
4) Selection of the visible objects having at least one point in the point cloud taking into account the rotation of the different objects.

In order to obtain a much simplified and reduced GT to work with, the tool has been designed with sensor synchronization as the basis. In this way, GT does not have to be calculated every time data is obtained from a sensor, or in certain timestamps in which data is available. Because of this, it is necessary to activate the CARLA synchronization option for the correct functioning of the tool.

C. Data storage method

After having obtained the GT from the objects in the environment, data must be saved. Two options are offered: 1) Using ROS in simulation time. 2) Saving the data in a CSV file for later use.

For the first option, two ROS messages are created, which are composed of other standard ROS messages. The first message saves a list of the second message built, while the second message includes the most important information of each object as shown in Table II. The saved data of the objects are those necessary to perform the tasks of 2D detection, 3D detection and object tracking.

For the second option, a CSV file is provided in which the same information as in the previously described ROS messages is stored. As the data has a different format, and in order not to generate confusion, headers are added to simplify its use.

Therefore, this tool defines different object types for GT generation (pedestrian, car, cyclist), providing two use modes. All the settings to follow for a particular architecture are performed on a configuration file within the ROS node.

V. FAULTS FOUND IN CARLA SIMULATOR

During the development of our PerDevKit, multiple problems have been encountered due to diverse issues with CARLA, as this simulator is open-source and is continuously evolving with the contributions of the research community. We present issues detected using version 0.9.8 of the simulator.

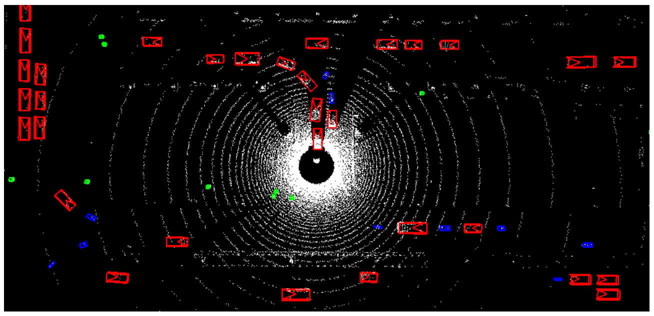
Fig. 3: Different classes of objects in BEV.

One of the problems observed when using the CARLA-ROS bridge, is the classes presented in ROS. While in the ROS message, up to 12 different types can be adopted, it has been found that even using CARLA's own object generation functions, only two of them are displayed: car and pedestrian. The solution adopted for the AD PerDevKit consists in the analysis of the width of the objects to distinguish more types of vehicles. This strategy works because the 3D boxes of motorcycles/bicycles are narrower than those of cars, and trucks have wider 3D boxes than cars. Fig. 3 shows the different classes detected by our tool, with pedestrians in green, motorcycles/bicycles in blue and cars/trucks in red.

Obtaining data from the sensors simulated by CARLA, is one way to understand the environment. Studying this data, several errors have been found in its simulation.

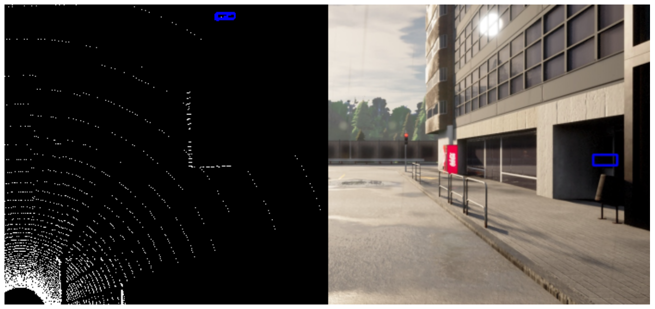
Fig. 4: Error in point clouds given by CARLA.

A good part of the maps provided by CARLA include buildings, so the LiDAR point cloud from the ego-vehicle should represent these static objects in the environment. However, certain buildings do not have any hit-boxes with respect to the LiDAR beams. Such issue not only worsens the given point clouds but also can produce failures in the vision system as seen in Fig. 4. In this particular case a vehicle is detected behind the building, because the point cloud does not collide with the building, it pass through it and the vehicle bounding box contains at least one of the LiDAR points, therefore the object is considered as visible. This failure is not a problem of the malfunction of CARLA, but an error in the creation of some of the buildings on the maps.

Finally, one of the most repeated flaws throughout the simulator is the definition of parked cars on the maps. While all spawned vehicles within the simulator are defined as different foreground objects, parked vehicles are defined as static background elements in the map, and for this reason they are not saved by the CARLA-ROS bridge. This would lead to situations where a model that is trained on CARLA data, detects a vehicle when it is visible to the sensor, but does not appear in the ground-truth. This object will be considered as a false positive, which negatively impact the model to be trained. That said, we are faced with a bad design decision, in which the position of parked vehicles is not allowed to be known since they are included as background in each map, so it is not really a bug in the simulator.

For all above reasons, it is recommendable to avoid areas that produce anomalous images, such as buildings without LiDAR hit-boxes or roads with parked vehicles defined as background in the maps, to provide correct visibility for

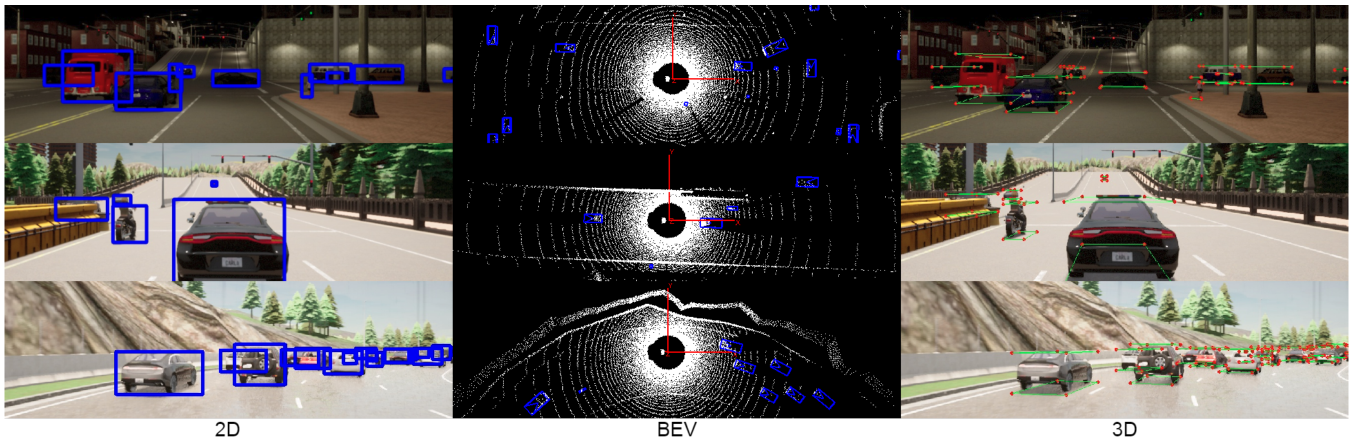

Fig. 5: Camera images and LiDAR point clouds with 2D and 3D bounding boxes.

all GT objects. Another alternative is the creation of maps from scratch with tools that allow editing under OpenDrive standard, such as: VectorZero's RoadRunner o VIRES ROD.

VI. AD PERDEVKIT DATASET

AD PerDevKit can be used for the development of control and decision-making modules involved in the design of AVs. In this case, perception layer can be substituted by the real-time GT provided by our tool in order to focus the study in the corresponding module, removing perception uncertainty. However, this paper is focused in the use of PerDevKit to help the training process of DL perception models. For this goal, virtual sensors data and their corresponding objects detection GT, recorded while a car is moving in virtual scenarios, are saved in a file in a synchronized way generating our dataset.

TABLE III: Dataset information

Town	Weather	Challenging scenarios	Frames	Objects per frames
03	Day	Intersection and roundabout	1708	29.11
	Night		1822	45.89
05	Day	Crowded intersections	3249	19.53
		Highway	2078	9.58
06	Day	Entrance to highway	1413	8.21
		Crowded highway	1673	16.1
	Rain		1487	15.86
07	Day	Small village	2738	10.89
	Rain		2560	9.78
10HD	Day	Crowded town	1587	77.9
	Night		1409	76.81

To train robust perception systems for challenging driving scenarios it is necessary to include a large number of out-of-distribution data in the dataset for training and testing.

Collecting such data in the real world is difficult because they rarely happen and can be dangerous since they contemplate hazard situations where accidents can happen. We leverage our kit to generate such rare data through 11 different scenes in different weather conditions, each containing between 1400 and 3200 frames with full set of annotations.

Data were recorded in 5 towns included in the CARLA simulator running the PerDevKit. The main characteristic of our dataset are described in Table III. As we can see it includes challenging crowded scenarios next to other daily traffic ones. Crowded scenarios have been included to train rare situations where interaction among agents are complex and collision may happen. We also provide other rare driving data as adverse weather and lighting (night, rainy, etc.). Fig 5 shows some examples of our driving scenes.

TABLE IV: Sensor data in the AD PerDevKit dataset.

Sensor	Brief Details
Camera	Front stereo camera at 20Hz, with a FoV of 85º and a resolution of 1280x720 generating RGB images.
LiDAR	360º of visibility, a maximum range of 120m, 64 beams at 20Hz and a vertical FoV of 2º to -24.9º, generating 1,300,000 points per second.
Radar	Frontal radar at 20Hz with a horizontal FoV of 90º, vertical FoV of 18º and a maximum range of 150m, generating 9,000 points per second.

For each agent (vehicle, pedestrian) we set a random behaviour and target destination to increase diversity and generate varied trajectories. After environment setup a vehicle is randomly selected to be our ego-vehicle and it is equipped to our sensor suite for data recording.

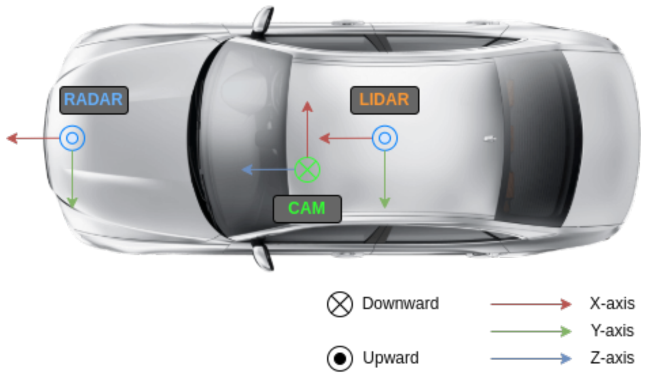

Fig. 6: Sensors setup and coordinate systems.

Our multi-sensor system consist of a front camera, a 360º LiDAR on the top of the vehicle and a front Radar. The characteristics of each sensor are explained in more detail in Table IV, and can be modified through the use of CARLA.

Each of the sensors uses a different coordinate axis as shown in Fig 6.

Our dataset is a useful tool for training DL based models, since it includes raw sensors data and object detection GT for each sensor through 2D/3D bounding boxes, temporal information of the objects as well as their visibility. Some scenes with their 2D and 3D bounding boxes are shown in Fig. 5. This dataset is freely available for researchers.

As a concept proof of the potential of our dataset we include an evaluation of the improvements obtained in a popular 2D detection method when we combine synthetic data with real data. In concrete, we will use YOLOv5 [18] in its versions "L" and "S", because this models achieves the best performance when compared to the rest of its versions.

TABLE V: Training results on CARLA and KITTI from scratch

Model	Train	Test	P	R	mAP
Yolov5S	Kitti 100%+ Carla	Kitti	0.884	0.853	0.925
		Carla	0.669	0.879	0.776
	Kitti 50%+ Carla	Kitti	0.925	0.873	0.901
		Carla	0.699	0.796	0.811
	Kitti 25%+ Carla	Kitti	0.831	0.755	0.841
		Carla	0.675	0.804	0.724
Yolov5L	Kitti 100%+ Carla	Kitti	0.933	0.874	**0.949**
		Carla	0.687	0.767	0.772
	Kitti 50%+ Carla	Kitti	0.910	0.841	**0.930**
		Carla	0.657	0.775	0.753
	Kitti 25%+ Carla	Kitti	0.873	0.794	**0.889**
		Carla	0.652	0.769	0.723

As training set we have chosen 3 different options, in which we have trained with a different percentage of this dataset combined with the KITTI dataset. As performance metric we use Precision (P), Recall (R) and mean Average Precision (mAP).

As we can see, performance get worse when % of real data decreases in the training. Real world performance is improved when the training is carried out with synthetic data. As baseline is considered the training with only 25% KITTI. For more information about this study we remit the readers to our publication in [19].

VII. CONCLUSIONS AND FUTURE WORKS

This paper presents a tool able to generate an infinite set of sensors data next to its corresponding 2D/3D objects detection GT using the CARLA simulator and ROS framework. This information is very useful to train DL models in a simple and cheap way in simulation, since it is not necessary to have a vehicle with the latest technology sensors to record data and manually annotate all the objects in the environment. Besides, a dataset using this tool while the vehicle is driving in some challenging scenarios has been presented. The potential of our tool in real world domain adaptation has been proved in a 2D object detection evaluation with YOLO using combinations of CARLA and KITTI data.

As future work we intend to generate more scenarios for wide our dataset. Finally, it is worth noting that this work is part of a larger project called AD DevKit, which plans an holistic evaluation tool for complete autonomous driving architectures in CARLA.

References

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

[2] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[5] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

[6] M. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," *CoRR*, vol. abs/1911.02620, 2019. [Online]. Available: http://arxiv.org/abs/1911.02620

[7] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," *CoRR*, vol. abs/2006.14480, 2020. [Online]. Available: https://arxiv.org/abs/2006.14480

[8] X. Weng, Y. Man, D. Cheng, J. Park, M. O'toole, and K. Kitani, "All-in-one drive: A large-scale comprehensive perception dataset with high-density long-range point clouds," 12 2020.

[9] B. Hurl, K. Czarnecki, and S. Waslander, "Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2522–2529.

[10] M. Maximov, K. Galim, and L. Leal-Taixé, "Focus on defocus: bridging the synthetic to real domain gap for depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1071–1080.

[11] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3234–3243.

[12] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Ue4sim: A photo-realistic simulator for computer vision applications," *CoRR*, vol. abs/1708.05869, 2017. [Online]. Available: http://arxiv.org/abs/1708.05869

[13] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016.

[14] Epic Games, "Unreal engine." [Online]. Available: https://www.unrealengine.com

[15] A. for Standarization of Automation and M. Systems, "Asam opendrive®." [Online]. Available: https://www.asam.net/standards/detail/opendrive/

[16] J. Knodt, J. Bartusek, S.-H. Baek, and F. Heide, "Neural ray-tracing: Learning surfaces and reflectance for relighting and view synthesis," 2021.

[17] S. Koksbang and S. Hannestad, "Studying the precision of ray tracing techniques with szekeres models," *Physical Review D*, vol. 92, no. 2, Jul 2015. [Online]. Available: http://dx.doi.org/10.1103/PhysRevD.92.023532

[18] G. Jocher, "ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements," https://github.com/ultralytics/yolov5, Oct. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4154370

[19] M. Antunes, L. M. Bergasa, J. Araluce, R. Gutiérrez, J. F. Arango, and M. Ocaña, "Including transfer learning and synthetic data in a training process of a 2d object detector for autonomous driving," Submitted to ITSC 2022.