Augmented LiDAR Simulator for Autonomous Driving

Jin Fang, Dingfu Zhou, Feilong Yan, Tongtong Zhao, Feihu Zhang, Yu Ma, Liang Wang, and Ruigang Yang

Abstract—In Autonomous Driving (AD), detection and tracking of obstacles on the roads is a critical task. Deep-learning based methods using annotated LiDAR data have been the most widely adopted approach for this. Unfortunately, annotating 3D point cloud is a very challenging, time- and money-consuming task. In this letter, we propose a novel LiDAR simulator that augments real point cloud with synthetic obstacles (e.g., vehicles, pedestrians, and other movable objects). Unlike previous simulators that entirely rely on CG (Computer Graphics) models and game engines, our augmented simulator bypasses the requirement to create high-fidelity background CAD (Computer Aided Design) models. Instead, we can deploy a vehicle with a LiDAR scanner to sweep the street of interests to obtain the background points cloud, based on which annotated point cloud can be automatically generated. This “scan-and-simulate” capability makes our approach scalable and practical, ready for large-scale industrial applications. In this letter, we describe our simulator in detail, in particular the placement of obstacles that is critical for performance enhancement. We show that detectors with our simulated LiDAR point cloud alone can perform comparably (within two percentage points) with these trained with real data. Mixing real and simulated data can achieve over 95% accuracy.

Index Terms—Simulation and animation, computer vision for automation, object detection, segmentation and categorization.

I. INTRODUCTION

LiDAR devices have been widely used in robotics and in particular autonomous driving. They provide robust and precise depth measurements of their surroundings, making them usually the first choice for environmental sensing. Typically the raw point cloud from LiDAR is sent to a computer vision system to find obstacles and other relevant driving information. Currently, high-performance vision systems usually are based on deep learning techniques. Deep neural networks (DNNs) have proven to be a powerful tool for many vision tasks [3], [25].

The success of DNNs rely most on the quality and quantity of labeled training data. Compared to image data, 3D point cloud from LiDAR is much harder to be labeled manually [11], [22]. This is particularly true for real-time LiDAR scanners, such as the points from Velodyne. These devices, predominantly used in autonomous vehicles (AV), generate sparse point cloud that is difficult to interpret. Labeling the huge amount of point cloud data needed for the safety of AV quickly becomes prohibitively expensive.

There have been work to leverage computer graphics techniques to generate synthetic labeled data (e.g., [20], [17], [16], [13], [23], [24]). While these simulated data are shown to be useful to improve DNNs’ performance, there remain a few unsolved problems. First the CG environment is mostly manually crafted with limited scale and complexity. As we will show in our experiments, the fidelity of background plays a significant role in perception accuracy. Based on our survey, creating photo-realistic scenes, with a price tag of over 10 K USD per kilometer, simply does not scale. The price can be decreased by half if we drop out the textures and materials. With the help of scene parsing, the cost of our data annotation is only 200 USD per Km and less than one day. Secondly, the obstacle placement and movement are mostly based on heuristics, which often do not reflect the diversity of our real world. Thirdly in the scope of LiDAR simulation [29], [24], existing methods simply render the scene depth, without considering the physical characteristics of LiDAR, leading to obvious artifacts. As such detector trained with only synthetic data performed poorly (e.g., around 30% accuracy) on real data [29].

In this letter, we present a novel hybrid point cloud generation framework for automatically producing high-fidelity annotated 3D point data, aimed to be immediately used for training DNNs models. To both enhance the realism of our simulation and reduce the cost, we have made the following design choices. First, we take advantage of mobile LiDAR scanners, which are usually used in land surveys, to directly take 3D scan of road scenes as our virtual environment, which naturally retains the complexity and diversity of real-world geometry, therefore bypassing completely the need for environmental model creation. Secondly we develop novel data-driven approaches to determine obstacles’ poses (position and orientation) and shapes (CAD model). More specifically, we extract from real traffic scenes the distributions
of obstacle models and their poses. The learned obstacle distribution is used to synthesize the placement and type of obstacles to be placed in the synthetic background. Note that the learning distribution does not have to be aligned with the captured background. Different combinations of obstacle distribution and background provide a much richer set of data without any additional data acquisition or labeling cost. Thirdly we develop a novel LiDAR render system that takes into considerations both the physical model and real statistics from the corresponding hardware. Combing all these together, we have developed a simulation system that is realistic, efficient, and scalable.

The main contributions of this letter include the following.

- We present a LiDAR point cloud simulation framework that can generate the annotated data for autonomous driving perception, the resultant data have achieved comparable performance with real point cloud. Our simulator provides the realism from real data, with the same amount of flexibility that was previously available only in VR-based simulation, such as regeneration of traffic patterns and change of sensor parameters.
- We combine real-world background models, which are acquired by LiDAR scanners, with realistic obstacle placement that is learned from real traffic scenes. Our approach does not require the costly background modeling process, or heuristics-based rules. It is both efficient and scalable.
- We demonstrate that the model trained with synthetic point data alone can achieve competitive performance in terms of 3D obstacle detection and semantic segmentation. Mixing real data and simulated data can easily outperform the model trained with the real data alone.

II. RELATED WORK

As deep learning becomes prevalent, increasing effort has been invested to alleviate the lack of annotated data for training DNNs. In this section, we mainly review the recent work on data simulation and synthesis for autonomous driving.

To liberate the power of DNNs from limited training data, [18] introduces SYNTHIA, a dataset of a big volume of synthetic images and associated annotation of urban scenes. [6] imitates the tracking video data of KITTI [7] in virtual scenes to create a synthetic copy, and then augments it by simulating different lighting and weather conditions. [6] and [17] generate a comprehensive dataset with pixel level labels and instance level annotation, aiming to provide a benchmark supporting both low-level and high-level vision tasks. [4] added CG characters to existing street view images for pedestrian detection. However the problem of existing pedestrians in the image is not discussed. [8] synthesizes annotated images for vehicle detection, and shows an encouraging result that it is possible for state-of-the-art DNNs models trained with purely synthetic data to beat the one trained on real data, when the amount of synthetic data is sufficiently large. Similarly, our target is to obtain the comparable perception capability from purely simulated point data by virtue of their diversity. To address the scarcity of realism of virtual scenes, [37] and [1] proposes to augment the dataset of real-world images by inserting rendered vehicles into those images, thus inheriting the realism from the background and taking advantage of the variation of the foreground. [21] presents a method for data synthesis based on procedural world modeling and more complicated image rendering techniques. Our work also benefits from real background data and physically-based sensor simulation.

While most of previous works are devoted to image synthesis, only a few focus on the generation and usage of synthetic LiDAR points cloud, albeit they play an even more important role for autonomous driving. [36] proposes to creating a ground truth dataset through simulator to verify algorithms. Recently, [35] uses terrain primitives from semantic parsing to compose arbitrary scenes but the precision is lower than their baseline. [29] collects the calibrated images and point cloud using the APIs provided by the video game engine, and apply these data for vehicle detection in their later work [23]. CARLA [5] and AutonoVi-Sim [2] also furnish the function to simulate LiDAR points data from the virtual world. However their primary target is to provide a platform for testing algorithms of learning and control for autonomous vehicles.

In summary, in the domain of augmenting data, our method is the first to focus on LiDAR points. In addition, we can change the parameters of LiDAR, in terms of placement and the number of lines, arbitrarily. The change of camera parameters has not been reported in image-augmentation methods.

III. METHODOLOGY

In general, we simulate the data acquisition process of the LiDAR sensor mounted on the autonomous driving vehicle in the real traffic environment. The whole process is composed of several modules: static background construction, movable foreground obstacles generation and placement, LiDAR points cloud simulation and the final verification stage. A general overview of our proposed framework is described in Fig. 1 and more details of each module will be described in the following context.

A. Static Background Generation

Different from other simulation frameworks [5], [29], which generate both the foreground and background point cloud from an artificial virtual world, we follow [34] to generate the static background with the help of a professional 3D scanner RIEGL.\(^1\)

The RIEGL is a high speed, high performance dual scanner mobile mapping system which provides dense, accurate, and feature-rich data at highway speeds. The resolution of the point cloud from the RIEGL scanner is about 3 cm within a range of 100 meters. In the real application, a certain traffic scene will be repeatedly scanned several rounds (e.g., 5 rounds) in our experiments. Thanks for the high precise inertial navigation system and differential GPS, the point cloud registration can reach centimeter level for different loops. With multiple scanning, the resolution of the point cloud can reach about 1 cm. An example of the scanned point cloud is displayed in Fig. 2. By using this scanner, the structure details can be well obtained. Theoretically,

\(^1\)http://www.rie gl.com
we can simulate any other type of LiDAR points cloud whose point distance is larger than 1 cm. For example, the resolution of common used Velodyne HDL-64E S3 is about 1.5 cm within a range of 10 meters.

To obtain a clean background, both the dynamic and static movable obstacles should be removed away. To improve the efficiency, state-of-the-art semantic segmentation approaches are employed to obtain an initial labeling results roughly and then annotators correct these wrong parts manually. Here, a modified PointNet++ [14] network is used for semantic segmentation and the average precision can reach about 98.0%. Based on the semantic information, holes on the ground plane are filled are filled simply by using Poisson surface reconstruction [31] in PCL.²

### B. Movable Obstacle Generation

After obtaining the static background, we need to consider how to add movable obstacles in the environment. Particularly, we find that the position of obstacles has a great influence on the final detection and segmentation results. However, this has been rarely mentioned by other simulation approaches. Different with [32] and [33], we propose a simpler data-driven-based method to generalize the obstacle’s pose based on their distribution in the real dataset.

1) **Probability Map for Obstacle Placement:** First of all, a Probability Map (PM) [4] will be constructed based on the obstacles distribution in the labeled dataset from different scenarios. In the PM, the position with a higher value will be selected to place an obstacle with a bigger chance. This map is built based on some labeled data or predictions from other modern detectors. First of all, a local area $G$ is divided into $M \times N$ grids. Corresponding to these grids, a weight matrix $W$ and direction angle matrix $\theta$ are assigned to them with initial values of zero. Then all the objects are transformed into the global world coordinate and their location indexes in $G$ are recorded too. At the same time, both the $W$ and $\theta$ are updated accordingly if there is an object located in a certain grid. To well consider the uncertainty of object’s location during the detection process, the object center is modeled as a Gaussian distribution matrix $T$ by considering the surrounding grids.

Finally, given a certain pose, the position and direction of the obstacles can be sampled by weighted random sampling with the help of normalized $W$ and $\theta$. A summary of building PM can be found in Algorithm 1. Particularly, we have built different PMs for different classes. Given the semantics of background, we could easily generalize the PM to other areas in a texture-synthesis fashion. Furthermore, the trajectories of dynamic obstacles can be obtained with rule-based or data-driven-based traffic flow simulation methods.

2) **Model Selection:** Similar to the obstacle’s position, a data-driven-based strategy has been employed to determine the occurrence frequency of different obstacle categories. Based on the labeled dataset, prior occurrence frequency information of different types can be easily obtained. During the simulation

---

²https://velodynelidar.com/hdl-64e.html
³http://pointclouds.org
In order to apply

\[ E_{return} = E_{emit} \star R_{rel} \star R_{ia} \star R_{atm}, \]

where \( E_{return} \) denotes the energy of a returned laser pulse and \( E_{emit} \) is the energy of original laser pulse, \( R_{rel} \) represents the reflectivity of the surface material, \( R_{ia} \) denotes the reflection rate w.r.t the laser incident angle, \( R_{atm} \) is the air attenuation rate because the laser beam is absorbed and reflected when traveling in the air, \( \sigma_{air} \) is a constant number (e.g., 0.004) in our implementation, and \( D \) denotes the distance from LiDAR center to target.

Given the basic principle above, the common used multi-beam LiDAR sensor (e.g., Velodyne HDL-64E S3) for autonomous vehicles can be simulated. It emits 64 laser beams in different vertical angles ranging from \(-24.33^\circ\) to \(+2^\circ\), as shown in Fig. 4. These beams can be assumed to be emitted from the center of the LiDAR. During data acquisition, HDL-64E S3 rotates around its own upright direction and shoots laser beams at a predefined rate to accomplish \(360^\circ\) coverage of the scenes.

Theoretically, 5 parameters of the beam should be considered for generating the point cloud, including the vertical and azimuth angles and their angular noises, as well as the distance measurement noise. Ideally, these parameters should keep constant, however, we found that different devices have different vertical angles and noise. To be closer to reality, we obtain these values from real point clouds statistically.

Specifically, we collect real point clouds of these HDL-64E S3 sensors atop parked vehicles, guaranteeing the point curves generated by different laser beams to be smooth. The points of each laser beam are then marked manually and fitted by a cone with the apex located in the LiDAR center. The half-angle of the cone minus \(\pi/2\) forms the real vertical angle while the noise variance is figured out from the deviation of lines constructed by the cone apex and the points from the cone surface.

The real vertical angles usually differ from the ideal ones by 1–3°. In our implementation, we approximate aforementioned noises using Standard Gaussian Distribution, setting distance noise variance to 0.5 cm and the azimuth angular noise variance 0.05°.

2) Point Cloud Rendering: In order to generate a point cloud, we have to compute intersections of laser beams and the virtual scene, for this we propose a cube map based method to handle the hybrid data of virtual scenes, points and meshes. Instead of computing intersections of beams and the hybrid data, we compute the intersection with the projected maps (e.g., depth map) of scenes which offer the equivalent information but much easier.

To do this, we first perspectively project the scene onto 6 faces of a cube centered at the LiDAR origin to form the cube maps as in Fig. 5. The key to make cube maps usable is to obtain the smooth and holeless projection of the scene, with presence
of environment points. Therefore we render the environment point cloud using surface splatting [30], while rendering obstacle models using the regular method. We synergize these two parts in the same rendering pipeline, yielding in a complete image with both the environment and obstacles. In this way, we get 3 types of cube maps: depth, normal, and material which are used in Eq. (1).

Next, we simulate the laser beams according to the geometric model of HDL-64E S3, and for each beam we look for the distance, normal and material of the target sample it hit, with which we generate a point for this beam. Note that some beams are likely discarded, if its returned energy computed with Eq. 1 is too low or it hit a empty area in the cube face, which indicates the sky.

Finally, we automatically generate the tight oriented bounding box (OBB) for each obstacle by simply adjusting its original CAD’ OBB to points of the obstacle, or follow KITTI to generate an amodal bounding box. The ground truth annotation is customizable.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

The whole simulation framework is a complex system. Direct comparison of different simulation system is a difficult task and it is also not the key point of this letter. The ultimate objective of our work is to boost DNNs’ perception performance by introducing free auto-labeled simulation ground truth. Therefore, the comparison of the different simulators can be transferred by comparing the point cloud generated by different ones. Here, we choose an indirect way of evaluation by comparing the DNNs’ performance trained with different simulation point cloud. To highlight the superiority of our proposed framework, we plan to verify it on two types of dataset: public and the self-collected point cloud. Due to the popularity of Velodyne HDL-64E in the field of AD, we set all the model parameters based on this type of LiDAR in our simulation process and all the following experiments are executed based on this type of LiDAR.

A. Evaluation on Public Dataset

Currently, CARLA, as an open-source simulation platform, has been widely used for different kinds of simulation purposes (e.g., [10], [15], [19], [28]). Similar to most typical simulation frameworks, both the foreground and background CG models have to be built in advance. As we have mentioned before, the proposed framework only need the CG models of foreground and the background can be directly obtained by a laser scanner. Here, we take CARLA as a representative of traditional methods for comparison.

1) Dataset: Simulation Data: we use CARLA and the proposed method to produce two groups of simulation data. For both simulators, we choose similar traffic scenarios for generating the background point cloud. For a fair comparison, similar number of foreground objects have been added for both. In addition, for better generalization, we first generate 100,000 point cloud frames and then randomly select 8,000 frames for training and 2,000 frames for validation. As in KITTI benchmark, ground truth amodal bounding box is generated for each foreground object. Three common categories of objects have been added into the point cloud as “car,” “pedestrian” and “cyclist”. Due to the lack of the camera image, the objects have not been categorized into different hard levels (e.g., easy, moderate and hard). The point cloud is generated in 360° field-of-view and only the front 90° field-of-view is used for our training.

Real Data: All the evaluations are executed on the third-party public KITTI object detection benchmark. This data has been divided into training and testing two subsets. Since the ground truth for the test set is not available, we subdivide the training data into a training set and a validation set as described in [3], [26]. Finally, we obtained 3,712 data samples for training and 3,769 data samples for validation. On the KITTI benchmark, the objects have been categorized into “easy,” “moderate” and “hard” based on their height in the image and occlusion ratio etc. Here, we merge them together for evaluation because they are equally important for the real AD application. In addition, the intensity attribute has been removed in our experiments.

2) Evaluation Methods: Two popular perception tasks in AD application have been evaluated here including instance segmentation and 3D object detection. For each task, one state-of-the-art approach is used for evaluation here. We choose SECOND [27] for object detection evaluation. While for instance segmentation, an accelerated real-time version of MV3D [3] from ApolloAuto5 project is used here.

For the 3D object detection, we take the AP-70 (Average Precision) for comparison. For instance segmentation, we take the mean of metric of mean bounding box (Bb)/mask AP proposed in etc. Here, we merge them together for evaluation because they are equally important for the real AD application. In addition, the intensity attribute has been removed in our experiments.

3) Experimental Results and Analysis: Two kinds of way of using the simulation data are advocated here. One is to train a model purely using simulation data and the other is to train a model on the simulation data first and then fine-tuned with some real data. The experimental results are shown in Table II. Compared with CARLA, the model trained purely by the proposed simulation point cloud achieves better performances on both instance segmentation and object detection. For instance segmentation, it gives more than 20 points improvements for mean AP and Mask AP. While for object detection, the AP is also improved by a big margin. With the help of simulation data,
Table II

| Methods          | Instance Segmentation mean AP | Mask AP | 3D Detection (AP 70) | Easy | Med | Hard |
|------------------|-------------------------------|---------|----------------------|------|-----|------|
| CARLA            | 10.55                         | 23.98   | 33.46                | 28.61| 24.64|
| Proposed         | 33.28                         | 44.68   | 48.88                | 44.71| 40.61|
| Real KITTI       | 40.22                         | 48.62   | 87.66                | 77.35| 75.57|
| CARLA + Real     | 40.97                         | 49.31   | 87.20                | 77.59| 76.53|
| Proposed + Real  | 45.51                         | 51.23   | 87.99                | 78.25| 77.36|

Table III

| Methods          | Vehicle | Cyclist | Pedestrian | Mean AP% | Testing Data   |
|------------------|---------|---------|------------|----------|----------------|
| Proposed + CAD BG| 78.44   | 55.18   | 11.03      | 41.88    | Self-collected |
| Proposed + Scanned BG | 90.05 | 59.52   | 32.48      | 60.68    | Self-collected |
| CARLA + CAD BG   | 54.17   | 17.35   | 12.26      | 27.99    | KITTI          |
| Proposed + Scanned BG | 66.60 | 21.35   | 18.96      | 35.64    | KITTI          |

B. Simulation for Real Application

As we have mentioned, the proposed framework can easily solve the domain problem happening in the traditional simulators by collecting the similar environment by a profession laser scanner. For building a city-level large-scale background, one week is sufficient enough. Then, we can use this background to generate sufficient ground truth data.

To further prove the effectiveness of our proposed framework, we test it on a large self-collected dataset. The data is collected by Velodyne HDL-64E laser scanner from different cities. Totally, we have labeled 120,000 frames of point cloud, including 80,000 frames for training, 20,000 frames for validation and the rest is for testing. Specifically, the training and testing datasets come from different roads. Different from KITTI, our data is labeled for full 360° view with six types of obstacles, including “small cars,” “big motors” (e.g., truck and bus), “cyclists,” “pedestrians,” “traffic cones” and “other” unknown obstacles. The following experiments are executed based on this dataset.

1) Results on Self-Collected Dataset: First, we collect all background point cloud from different roads to build a large background database. In piratically, these roads only appear in training datasets. Then we generate sufficient simulation point cloud by placing different objects into the background. Finally, we randomly select 100 K frame of simulation point cloud for our experiments. The experimental results are shown in Table IV. From the first big row of the table, we can find that the model trained with 100 K simulation data gives comparable result with the model trained with 16 K real data, with only 2 points of gap. More important, the detection rate can achieve 91.02% which is relatively high enough for the real application. Furthermore, by mixing 1.6 K real data with the 100 K simulation data, the mean AP reaches 94.10 which outperforms 16 K real data.

From the second row of the table, we can see that 16 K real data together with 100 K simulation can achieve comparable performance with 100 K real data, which can save more than 80% of the money for annotation. Furthermore, we can obviously find that model trained with real data can be boosted by adding more simulation data even we have a large number of labeled real data.

C. Ablation Studies

The results in Table IV, show the promising performances of the proposed framework. The whole framework is a complex system and the final perception results are depended on different steps of the system. Therefore, a set of experiments have been designed to analyze the effectiveness of different
TABLE V
EVALUATIONS WITH DIFFERENT BACKGROUND FOR INSTANCE SEGMANETATION, WHERE “SIM,” “BG” AND “FG” REPRESENT “SIMULATION,” “BACKGROUND” AND “FOREGROUND” FOR SHORT

| Methods                | Mask AP 50 | Mask AP 70 | mean Mask AP |
|------------------------|------------|------------|--------------|
| No BG + Sim FG         | 1.86       | 1.41       | 1.23         |
| Scan BG + Sim FG       | 88.60      | 86.80      | 83.38        |
| Real BG + Sim FG       | 88.79      | 87.19      | 84.22        |
| Real BG + Real FG      | 90.40      | 89.45      | 86.33        |

Fig. 6. Models trained for 3D instance segmentation with different levels of background. We keep all other settings the same.

TABLE VI
EVALUATIONS FOR INSTANCE SEGMENTATION WITH DIFFERENT OBSTACLE POSES

| Methods                  | Mask AP 50 | Mask AP 70 | mean Mask AP |
|--------------------------|------------|------------|--------------|
| Random on road           | 73.03      | 69.23      | 65.95        |
| Rule-based               | 81.37      | 78.13      | 74.32        |
| PM w/o Gaussian weight   | 85.64      | 83.42      | 81.49        |
| Proposed PM w Gaussian weight | 86.57 | 84.55 | 82.47 |
| Augmented PM w Gaussian weight | 87.80 | 84.19 | 83.07 |
| Real Pose                | 88.90      | 86.80      | 83.38        |

TABLE VII
EVALUATIONS FOR INSTANCE SEGMENTATION WITH OR WITHOUT RANDOM DROPOUT

| Methods                  | Mask AP 50 | Mask AP 70 | mean Mask AP |
|--------------------------|------------|------------|--------------|
| w/o dropout              | 88.60      | 86.80      | 83.38        |
| w dropout                | 89.43      | 88.75      | 85.53        |
| Real Data                | 90.40      | 89.45      | 86.33        |

TABLE VIII
POINT CLOUD SIMULATION WITH OR WITHOUT NOISE PERTURBATION FOR INSTANCE SEGMENTATION. ALL THE OTHER SETTINGS ARE THE SAME

| Background              | Mask AP 50 | Mask AP 70 | mean Mask AP |
|-------------------------|------------|------------|--------------|
| w/o noise perturbation  | 86.19      | 83.77      | 80.40        |
| w noise perturbation    | 88.60      | 86.80      | 83.38        |

Fig. 7. An simulation example of VLS-128 with same LiDAR and obstacle poses, where (a) is the simulated and (b) is the real.

3) Random Dropout and Point Perturbation: Another difference between the real and simulated data is the number of the point in each frame. For real Velodyne HDL-64E, usually about 102,000 points will be returned each frame, while the number is about 117,000 in the simulated point cloud. This depends on different reasons: textures, material, surface reflectivity and even the color of objects. These properties can be easily obtained for man-made foreground obstacle, however, it is not a trivial task for backgrounds. Inspired by the dropout strategy in DNNs, we randomly drop a certain ratio of points during our model training process. The results are given in Table VII. Surprising, it can steadily improve the performance by 2 points for mean Mask AP in 3D instance segmentation.

In addition, during the points cloud rendering process, a random perturbation noise is also added for each point using a Standard Gaussian Distribution, by setting distance noise variance to 0.5 cm and the azimuth angular noise variance 0.05°. The results with or without perturbation noise are given in Table VIII. The results verify the effectiveness of the perturbation noise.

4) Extension to Different Sensors: Beyond the Velodyne HDL-64E LiDAR, the proposed framework can be easily generalized to other types of sensors such as Velodyne Alpha Puck (VLS-128). For our implementation, a friendly interface has been designed. With a slight modification of configure file, we can achieve different types of LiDAR points clouds. In the configure file, the specific LiDAR properties can be modified such as the channels number, range, horizontal and vertical field-of-view, angular and vertical resolution etc. Fig. 7 gives a simulation example of VLS-128.

V. CONCLUSIONS AND FUTURE WORKS

This letter presents an augmented LiDAR simulation system to automatically produce annotated point cloud used for
3D obstacle perception in AD scenario. Our approach is data-driven, with scanned background, obstacle poses and types that are statistically similar to that from real traffic data, and a general LiDAR renderer that takes into considerations of physical/statics properties of the actual devices. The most significant benefits of our approach are realism and scalability. We demonstrated realism by showing that the performance gap between detectors trained with real or simulated data is within two percentage points. The scalability of our system is by design, there is no manual labeling, and the combination of different backgrounds and different obstacle placement provides abundant labeled data with virtually no cost except computation. More importantly, it can rapidly simulate a scene of interest by simply scanning that area.

Looking into the future, we want to investigate the use of low-fidelity LiDAR. Our current hardware system can produce very high quality and dense 3D point cloud, which can be re-sampled to simulate any LiDAR type. By using a low-fidelity one, we think it may limit the LiDAR type our system can simulate, this could be a design choice between cost and versatility. Another area to address is to simulate the LiDAR intensity value for the foreground object. It involves the modeling of material reflectance properties under near infrared (NIR) illumination, which is doable but quite tedious.

REFERENCES

[1] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, “Augmented reality meets computer vision: Efficient data generation for urban driving scenes,” Int. J. Comput. Vis., vol. 126, no. 9, pp. 961–972, 2018.

[2] A. Best, S. Narang, L. Pasqualin, D. Barber, and D. Manocha, “Autonovisim: Autonomous vehicle simulation platform with weather, sensing, and traffic control,” in Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. Workshops, 2018, pp. 1048–1056.

[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3D object detection framework for autonomous driving,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 1907–1915.

[4] E. Cheung, A. Wong, A. Bera, and D. Manocha, “Mixpedests: Pedestrian detection in unannotated videos using synthetically generated human agents for training,” in Proc. 32 AAAI Conf. Artif. Intell., 2018, pp. 6738–6747.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in Proc. Conf. Robot Learn., 2017, pp. 1–16.

[6] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, “Virtual worlds as proxy for multi-object tracking analysis,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 4340–4349.

[7] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2012, pp. 3354–3361.

[8] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, “Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?” in Proc. IEEE Int. Conf. Robot. Autom., 2017, pp. 746–753.

[9] S. Kim, I. Lee, and Y. J. Kwon, “Simulation of a geiger-mode imaging lidar system for performance assessment,” Sensors, vol. 13, no. 7, pp. 8461–8489, 2013.

[10] B. R. Kiran et al., “Real-time dynamic object detection for autonomous driving using prior 3d-maps,” in Proc. Euro. Conf. Comput. Vis., 2018, pp. 567–582.

[11] R. Kopper, F. Bacim, and D. A. Bowman, “Rapid and accurate 3d selection by progressive refinement,” in Proc. IEEE Symp. 3D User Interfaces, 2011, pp. 67–74.

[12] T.-Y. Lin et al., “Microsoft COCO: Common objects in context,” in Proc. Eur. Conf. Comput. Vis., 2014, pp. 740–755.

[13] M. Mueller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, “Sim4cv: A photo-realistic simulator for computer vision applications,” Int. J. Comput. Vis., vol. 126, no. 9, pp. 902–919, 2018.

[14] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5099–5108.

[15] J. Qiu et al., “Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 3313–3322.

[16] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, “Playing for data: Ground truth from computer games,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 102–118.

[17] S. R. Richter, Z. Hayden, and V. Koltun, “Playing for benchmarks,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, vol. 2, pp. 2232–2241.

[18] 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 Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 3234–3243.

[19] I. Soh et al., “End-to-end multi-modal sensors fusion system for urban automated driving,” in Proc. Conf. Neural Inf. Process. Syst. Workshop Mach. Learn. Intell. Transp. Syst., 2018, pp. 1–9.

[20] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, “Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 2686–2694.

[21] A. Tsirikoglou, J. Kronander, M. Wrenninge, and J. Unger, “Procedural modeling and physically based rendering for synthetic data generation in automotive applications,” CoRR, vol. abs/1710.06270, 2017, pp. 1–13.

[22] P. Welinder and P. Perona, “Online crowdsourcing: Rating annotators and obtaining cost-effective labels,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, 2010, pp. 25–32.

[23] B. Wu, A. Wan, X. Yue, and K. Keutzer, “SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR points cloud,” in Proc. IEEE Int. Conf. Robot. Autom., 2018, pp. 1887–1893.

[24] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, “SqueezeSegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a LiDAR points cloud,” in Proc. IEEE Int. Conf. Robot. Autom., 2018, pp. 4376–4382.

[25] D. Xu, D. Anguelov, and A. Jain, “Pointfusion: Deep sensor fusion for 3D bounding box estimation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 244–253.

[26] D. Zhou et al., “IoU loss for 2D/3D object detection,” in Proc. Int. Conf. 3D Vis., 2019, pp. 85–94.

[27] Y. Yan, Y. Mao, and B. Li, “Second: Sparsely embedded convolutional detection,” Sensors, vol. 18, no. 10, 2018, Art. no. 3337.

[28] D. J. Yoon, Y. T. Yang, and T. D. Barfoot, “Mapless online detection of dynamic objects in 3D LiDAR,” in Proc. 16th Conf. Comput. Robot Vision (CRV), 2019, pp. 113–120.

[29] X. Yue, B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli, “A lidar point cloud generator: from a virtual world to autonomous driving,” in Proc. ACM Int. Conf. Multimedia Retrieval, 2018, pp. 458–464.

[30] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, “Surface splatting,” in Proc. ACM 28th Annua. Conf. Comput. Graph. Interactive Techn., 2001, pp. 371–378.

[31] M. Kathizan, M. Bolitho, and H. Hoppe, “Poison surface reconstruction,” in Proc. 4th Eurographics Symp. Geometry Process., vol. 7, 2006, pp. 61–70.

[32] A. Prakash et al., “Structured domain randomization: Bridging the reality gap by context-aware synthetic data,” in Proc. IEEE Int. Conf. Robot. Autom., 2019, pp. 7248–7255.

[33] A. Kar et al., “Meta-sim: Learning to generate synthetic datasets,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 4551–4560.

[34] P. Wang, X. Huang, X. Cheng, D. Zhou, Qi. Geng, and R. Yang, “The apollobbox open dataset for autonomous driving and its application,” IEEE Trans. Pattern Anal. Mach. Intell., to be published.

[35] A. Tallavajhula, Ç. Meriçli, and A. Kelly, “Off-road LiDAR simulation with data-driven terrain primitives,” in Proc. IEEE Int. Conf. Robot. Autom., 2018, pp. 7470–7477.

[36] S. Lee et al., “LiDAR simulation method for low-cost repetitive validation,” in Advances in computer science and ubiquitous computing, Berlin, Germany: Springer, 2015, pp. 237–242.

[37] F. Debevec, “Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography,” in Proc. ACM SIGGRAPH Classes, 2008, pp. 1–10.