Beltrán, J., Cortés, I., Barrera, A., Urdiales, J., Guindel, C., García, F. & de la Escalera, A. (27-30 October, 2019). *A method for synthetic LiDAR generation to create annotated datasets for autonomous vehicles perception* [Proceedings]. 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, pp. 1091-1096.

DOI: 10.1109/ITSC.2019.8917176

A Method for Synthetic LiDAR Generation to Create Annotated Datasets for Autonomous Vehicles Perception

Jorge Beltrán, Irene Cortés, Alejandro Barrera, Jesús Urdiales,
Carlos Guindel, Fernando García and Arturo de la Escalera
Intelligent Systems Laboratory (LSI) Research Group
Universidad Carlos III de Madrid, Leganés, Madrid, Spain
{jbeltran, irecorte, alebarre, jurdiale, cguindel, fegarcia, escalera}@ing.uc3m.es

***Abstract*— LiDAR devices have become a key sensor for autonomous vehicles perception due to their ability to capture reliable geometry information. Indeed, approaches processing LiDAR data have shown an impressive accuracy for 3D object detection tasks, outperforming methods solely based on image inputs. However, the wide diversity of on-board sensor configurations makes the deployment of published algorithms into real platforms a hard task, due to the scarcity of annotated datasets containing laser scans. We present a method to generate new point clouds datasets as captured by a real LiDAR device. The proposed pipeline makes use of multiple frames to perform an accurate 3D reconstruction of the scene in the spherical coordinates system that enables the simulation of the sweeps of a virtual LiDAR sensor, configurable both in location and inner specifications. The similarity between real data and the generated synthetic clouds is assessed through a set of experiments performed using KITTI Depth and Object Benchmarks.**

I. INTRODUCTION

Perception technologies for autonomous vehicles have seen rapid progress in the past few decades driven by the advances in computer vision and, mainly, machine learning. Today, the perception stack is closer than ever before to achieve the degree of robustness and reliability required for safe autonomous operation in the variety of real-world driving scenarios.

High-level planning and control modules require precise identification of other road participants, such as vehicles and VRUs (Vulnerable Road Users). On this regard, research on object detection for autonomous cars has moved over the last years from object detection in images to complete 3D detection. Today, available frameworks are able to provide the location and pose of the objects in the scene. These methods usually incorporate additional sources of data, enabling spatial reasoning. Among the sensors that can provide geometrical information, 360° LiDAR rangefinders have become the most popular choice in autonomous vehicles.

The use of different data modalities within the same framework entails specific approaches to appropriately fuse the information conveyed by the different types of data and, ultimately, provide the 3D detection result. A myriad of methods have been proposed recently to that end, such as AVOD [1] and SECOND [2]. All of them are based on modern deep learning techniques, particularly convolutional neural networks (CNNs), whose evolution has pushed the boundaries of sensor data processing.

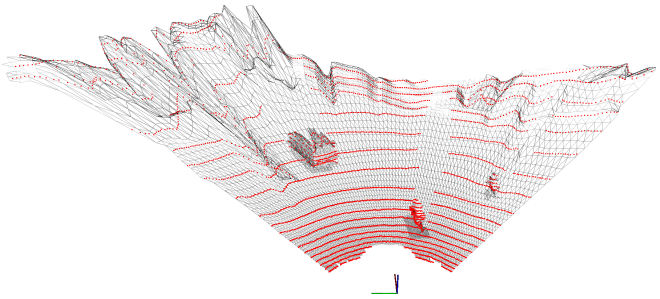


Fig. 1. Results of a reconstructed 3D mesh of the scene (gray), and a simulated LiDAR point cloud (red) using a virtual Velodyne HDL32E for a random sample from KITTI dataset.

Deep learning methods have brought numerous benefits, notably a significant improvement of the accuracy. However, the gap in performance comes at a cost: an unprecedented amount of labeled data are required to fit the models properly. Advances in CNN-based computer vision would not have been possible without the extensive publicly available datasets introduced in the last decade. Likewise, the viability of 3D detection multi-modal methods is subject to the availability of adequate datasets and benchmarks. Nevertheless, when dealing with autonomous driving, alternatives are narrowed down to a few options, such as KITTI [3] and, very recently, nuScenes [4].

The lack of a sufficient number of multi-modal datasets for training and evaluation may curb the widespread development and deployment of these methods on real platforms, especially considering their sensitivity against changes in the sensor configuration; namely, the relative position between sensors and the characteristics of the LiDAR model [5].

To address this problem, we propose a method that builds a 3D representation of the scene using real LiDAR information as input and produces synthetic point clouds as if generated by another sensor configuration, chosen by the user (Fig. 1). Changes can include the location of the LiDAR sensor and its intrinsic characteristics so that different models can be simulated. The approach aims to create new datasets from the existing ones avoiding the burden of labeling, with a focus on training and validation of 3D object detection algorithms.

Our work seeks to close the gap between synthetic and real LiDAR data by generating realistic inputs for the training of CNN models.

The rest of this paper is organized as follows. In Section II, a brief review of the point-cloud-based 3D object detection field is provided, covering the most relevant state-of-the-art methods, and an introduction to recent synthetic LiDAR generation approaches devised to enlarge the amount of annotated datasets available. In Section III, a detailed description of the proposed algorithm is presented. Experimental results are reported in Section IV, and the conclusions are included in Section V.

II. RELATED WORK

Our approach is geared towards LiDAR-based 3D detection frameworks. The work by Chen et al. [6], where image and LiDAR data were combined in a CNN-based framework to perform 3D object detection, paved the way for a myriad of recent developments devoted to the same goal. In that work, LiDAR data was represented in a Bird's Eye View (BEV) that has been leveraged by many others, such as AVOD [1] and BirdNet [7]. Although the BEV is a useful representation of LiDAR data, information is lost in the process. Another typical description, the front view, aims to represent the totality of the points in a 2D map through spherical projection and has also been adopted by several approaches [6], [8]. A different group of methods deals with the raw 3D points instead; for instance, F-PointNet [9] analyzes all the points lying within a pre-computed 2D detection to determine the location and 3D bounding box of the object. When the search space is not reduced beforehand, efficient representation techniques, such as the discretization of the raw point cloud into 3D voxels, is critical. Thus, VoxelNet [10] and SECOND [2] employ voxel-wise feature encoding layers to express the information contained in each partition.

Depth estimations from monocular images can replace LiDAR data on 3D detectors. Hence, some works have exploited this information to build a LiDAR-like point cloud and perform detection on it [11]. Wang et al. [12] proved that the representation of depth information as a 3D point cloud yields much better results than embedding the 2D depth map into the detection pipeline, thus highlighting the paramount importance of data representation. The point cloud obtained through this approach is, in general, noisy and needs further postprocessing [13].

Training samples for LiDAR-based algorithms can also be synthetic data obtained through simulation. Yue et al. [14] generate synthetic LiDAR data from a video game to create new automatically labeled samples to be put together with the real KITTI frames. The potential of synthetic data to complement the KITTI samples has been explored in other works [15], [16]. On the other hand, CARLA [17] is a simulator that allows generating fully synthetic data, including LiDAR samples. However, the validity of synthetic data to train models intended to be deployed in real-world applications is still an open question, especially when images are involved.

Alternatively, 3D reconstruction from single frames acquired from a moving platform is an issue that has been often discussed in the literature [18], [19]. These methods often rely on meshing techniques to convert the discrete LiDAR samples into a continuous surface [20].

III. PROPOSED APPROACH

The presented method for LiDAR generation is divided into two phases. First, a reconstruction of the scene is built by approximating a mesh of triangles to the real LiDAR point cloud. In a second step, a ray tracing algorithm is used to obtain the synthetic laser signal. Although this is an end-to-end pipeline, a prior pre-processing of the inputs has shown to improve the quality of the generated clouds.

A. Pre-processing

In order to enhance the outcome of the 3D reconstruction phase, data acquired at different frames can be aggregated. As a result, the sparsity of the LiDAR point cloud is reduced, and the geometry of temporarily occluded areas can be retrieved. However, this is not a straightforward process when performed in driving environments due to the presence of non-static objects, which may lead to inaccurate scene representation over time. To address this problem, a set of pre-processing tasks are performed on raw point clouds captured by the laser scanner.

For each of the sweeps to be accumulated, points belonging to dynamic elements (foreground) have to be handled separately from those corresponding to the static structure (background) of the scene. To this end, annotated object parameters $\{\mathbf{t}, \mathbf{s}, \theta\}$ –being $\mathbf{t}$ the translation vector (t_x, t_y, t_z), $\mathbf{s}$ the dimensions (s_x, s_y, s_z), and θ the yaw of the obstacle– can be used to subtract non-static readings from the cloud.

Once the LiDAR clouds are only composed of background points, information from multiple scans can be aligned. For this purpose, all frames must share a common coordinates system $\{S\}$ while preserving their relative position given by the motion of the ego-vehicle. To do so, each point cloud is transformed from their local coordinates system $\{L\}$ to $\{S\}$ following $\mathcal{C}_i^S = (R|t)\ \mathcal{C}_i^L$, being $\mathcal{C}_i^L$ the background point cloud at a certain frame i, $\mathcal{C}_i^S$ the points transformed into the shared axis, and R and t the rotation and translation matrix from the $\{L\}$ to $\{S\}$.

Similarly, data belonging to labeled objects can be accumulated over time to obtain a better representation of their geometry. As for background processing, instant LiDAR sweeps must be aligned in a single axis which, for simplicity, corresponds to the object's coordinates system $\{O\}$. To this end, the inverse transformation of the given annotated 3D bounding box is applied to the object's points at each frame, $\mathcal{P}_i^O = (R|t)^{-1}\ \mathcal{P}_i^L$, being $\mathcal{P}_i^L$ and $\mathcal{P}_i^O$ the points of the object at a frame i in local and object axes, respectively, and $R|t$ the transformation between the coordinates systems.

Finally, the dense clouds of the foreground elements can be placed back into their original poses at $\{L\}$ by undoing their corresponding transformation, $\mathcal{P}_i^L = (R|t)\ \mathcal{P}_i^O$.

B. 3D Reconstruction

After the pre-processing of the LiDAR data, the point clouds are prepared to be used as input for the next phase of the pipeline. The purpose at this stage is the reconstruction of a 3D mesh that accurately models the shape of the scene at a given time. Henceforward, the described steps are applied to both the object and background point clouds independently.

The designed meshing algorithm is divided into two stages and is based on [20]. Due to the nature of LiDAR data, the presented method is tailored to work in spherical axes instead of traditional Cartesian space. Consequently, each point $P_c = (x, y, z)$ has to be transformed into spherical coordinates $P_s = (\theta, \phi, r)$ following Eq. 1, where θ stands for the horizontal angle of the beam, ϕ for the vertical inclination and r for the range.

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= \arctan(y/x) \\ \phi &= \arcsin(z/r) \end{aligned} \tag{1}$$

In this manner, it is assumed that there cannot exist two points in the world for the same (θ, ϕ) pair. However, this constraint does not affect the reconstruction capabilities of the method, as it mimics the behavior of laser scanning.

At the first phase, a coarse mesh is approximated to the scene geometry. After the transformation between coordinates systems is performed, a rectangular grid with a resolution of $\Delta\theta \times \Delta\phi$ is built to divide the spherical space containing LiDAR information into square patches (see Fig. 2). Then, a depth map is built by setting the r value of the vertices of each cell to the average range of all points falling inside. As a result, at every vertex $V = (\theta, \phi)$ in the original grid, there may be up to four points, one for each surrounding patch. Finally, the mean r value of those corners is assigned to its corresponding vertex in the grid, thus dividing the squared patches into triangles and connecting the mesh. Triangles containing no points are removed.

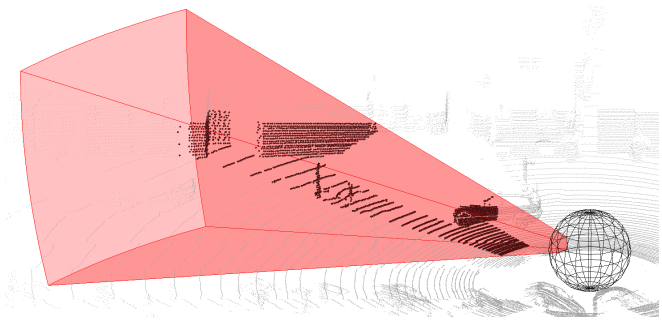


Fig. 2. Spherical grid generation represented in Cartesian coordinates.

In a second step, the resulting structure is converted into a multi-resolution grid, where every vertex may have between four and eight triangle neighbors, except those located in the borders. This kind of mesh enables the recursive subdivision of the cells to scale the resolution at patches where LiDAR data presents high variance. Thus, increasing the degree of adaptability of the faces to fit their corresponding points more accurately.

This procedure is conducted in a breadth-first fashion, evaluating the candidacy of every cell at a given resolution level to be subdivided. This way, mesh remains in a valid state at every step of the refinement process.

For a triangle to be considered a candidate, the error between the face plane and the points falling inside the patch must be greater than a threshold δ. The fitting error is calculated by means of the bidirectional Hausdorff distance (Eq. 2) between the triangle vertices $\mathcal{V}_i$ and its corresponding point cloud set $\mathcal{P}_i$. The one-sided distance is defined in Eq. 3.

$$d_H(\mathcal{V}, \mathcal{P}) = \max(\widetilde{d}_H(\mathcal{V}, \mathcal{P}), \widetilde{d}_H(\mathcal{P}, \mathcal{V})) \tag{2}$$

$$\widetilde{d}_H(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\| \tag{3}$$

A triangle may be divided by its altitude segment, creating two sibling faces. Whenever a candidate is divided, so it does the triangle that meets its hypotenuse (or bottom neighbor), if any. For a *split* operation to be performed, two conditions have to be fulfilled. First, the triangle to be split and its bottom neighbor must have equal size. Second, the newly created children faces must contain at least one point, except if any of the resulting triangles would not have a bottom neighbor. In this case, the empty child is removed. If both requirements are met, the candidate is divided. The refinement process stops when for every triangle in the mesh $d_H(V_i, P_i) < \delta$, or no triangle can be split. Fig. 3 shows a feasible configuration for a refined mesh.

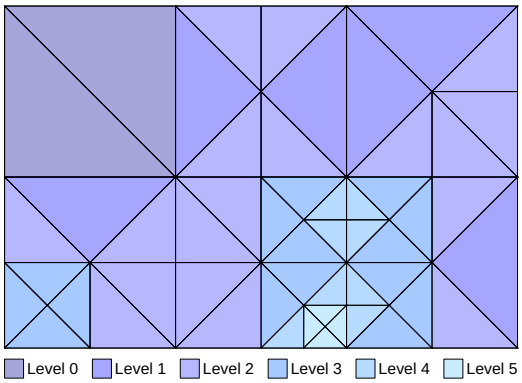


Fig. 3. Grid division process.

To compute the range coordinate of the final vertices, the same approach previously used for the coarse meshing procedure is followed, so that all the triangles are connected composing a single surface.

As mentioned before, computing background and foreground reconstructions separately allows different mesh resolutions to be used, as well as different fitting thresholds. This enables defining unlike refinement degrees for each of the meshes. A sample can be observed in Fig. 4.

Once the vertices of the triangles composing the mesh are obtained, they are converted back into Cartesian space

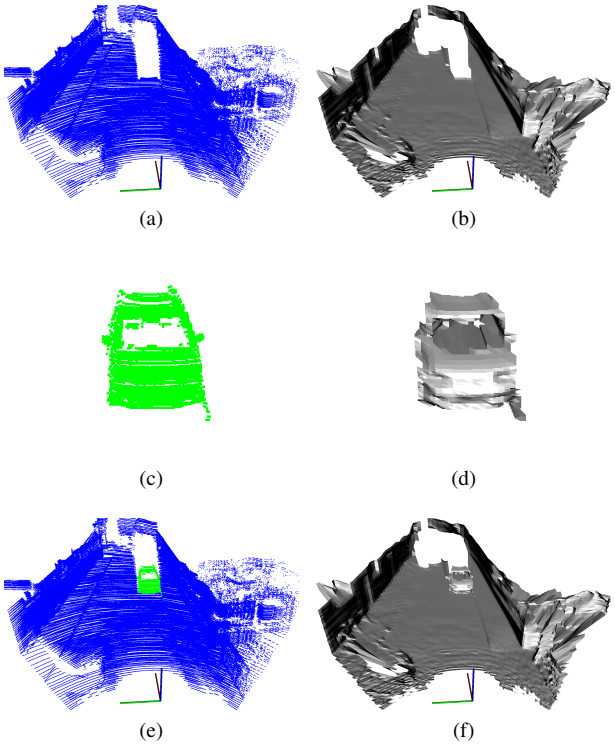


Fig. 4. Meshing process: (a) Background points. (b) Background mesh. (c) Foreground points. (d) Foreground mesh. (e) All points. (f) Final mesh.

by computing the (x, y, z) coordinates of each vertex in the cloud reversing the Cartesian-to-spherical transformation performed before.

Lastly, background and foreground meshes are merged to reproduce the original layout of the geometry of the environment, as shown in Fig. 4f. The effects of refinement can be observed in Fig. 5.

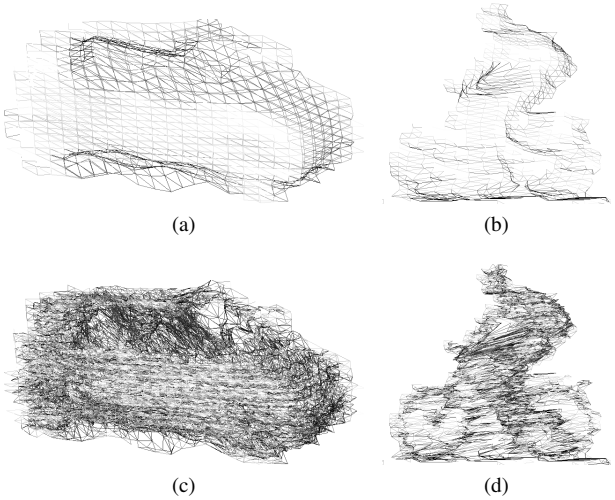


Fig. 5. Differences between coarse (a, b) and fine (c, d) reconstruction of obstacles.

C. LiDAR scans simulation

The last stage of the presented approach addresses the generation of realistic LiDAR signals from the previously reconstructed scene. This step enables the creation of new laser datasets by defining the characteristics of the simulated device, as well as its 3D pose. Hence, the output cloud depends on two variables: the position and orientation of the desired virtual scanner, and the intrinsic parameters of the simulated device, which can be described by the inclination angle of the rings, the horizontal resolution or azimuth, and the minimum and maximum range distances.

Regarding the laser scanner simulation, a ray tracing approach is followed. For each of the inclination angles defined by the rings of the sensor, a whole turn around the Z-axis is performed in steps of size θ. At every position, a LiDAR beam is traced from the origin of the sensor in the corresponding direction so that intersections between the ray and the mesh faces can be checked. To this end, the well-known Möller-Trumbore [21] algorithm is used, which allows the computation of the intersection point between a ray and a triangle without solving the plane equation.

Despite the possibility of having occluded surfaces due to the separate generation of object and background meshes, each of the simulated LiDAR beams generates a single 3D point, as only the collision with the closest triangle of the mesh is considered.

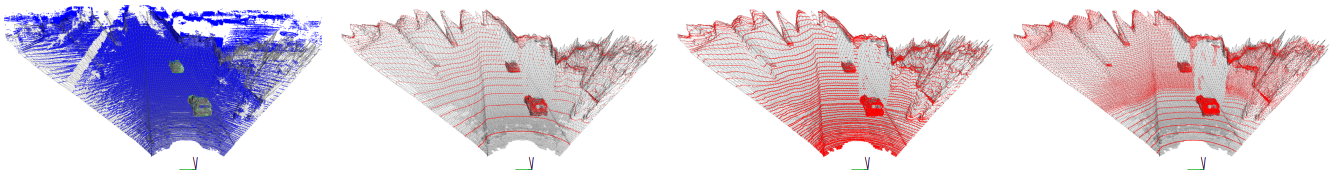
Finally, to increase the similarity of the generated cloud and the one obtained by the equivalent real device, a noise signal is added to the output based on the specification of the real device, which is usually defined by a random process with a normal distribution. As an example, in Fig. 6 the output for different models of Velodyne LiDARs is shown.

IV. EXPERIMENTS

In order to assess the quality of the LiDAR signals generated using the proposed approach, a thorough analysis is presented here. Virtual scans fidelity is evaluated in two manners. On the one hand, the resulting point clouds are compared to a manually annotated depth ground truth. On the other hand, three state-of-the-art 3D object detection networks trained using the synthetic laser scans produced by the presented algorithm. The selected works process LiDAR information differently. First, in [9], the raw points are used without any pre-processing. SECOND, in [1] the laser data is projected into a Bird's Eye View image before being fed into a 2D CNN detection network. Finally, in [2], the point cloud is voxelized, and 3D convolutions are applied to the resulting grid.

A. Cloud quality evaluation

For the first analysis, the KITTI Depth Prediction Evaluation Benchmark [22], composed of labeled depth maps, has been used. The semi-dense ground truth provided in this dataset allows measuring the depth error of the generated clouds by projecting them into the camera plane. Then, the percentage of correct pixels can be computed by determining the number of points where $\left|D_i^c - D_i^{gt}\right| < \epsilon$ fulfills, being



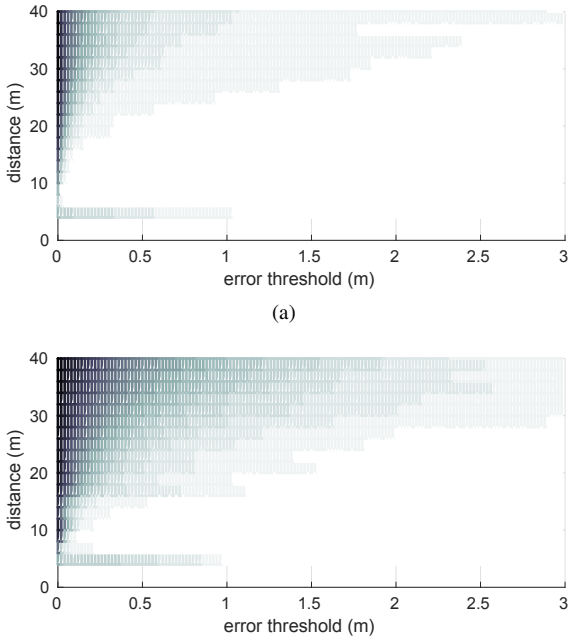
(a) (b) (c) (d)

Fig. 6. Outputs for different LiDAR devices based on the same input mesh. (a) Accumulated LiDAR points from 30 consecutive frames. (b) Simulated points for a Velodyne VLP32C model. (c) Simulated points for a Velodyne HDL64E model. (d) Simulated points for a Velodyne VLS128 model.

D_i^c the value of the Z coordinate of point i in camera coordinates, D_i^{gt} its corresponding ground truth depth, and ϵ the error threshold.

The validation of the method has been performed over a random sample of 2000 frames, accumulating both the previous and next 15 consecutive clouds. For the background mesh generation, the spherical grid is created with a resolution of $\Delta\theta = 1$, $\Delta\phi = 2$, while $\delta = 0.1$ is used as the fitting threshold. For objects reconstruction, mesh parameters are set to $\Delta\theta = 1$, $\Delta\phi = 2$, while $\delta = 0.01$. These parameters have been empirically obtained.

As shown in Fig. 7, the annotated ground truth is not perfect and presents significant errors for depths higher than 20 meters, when compared to the original LiDAR signal. Besides, the quality of the synthetic cloud for near distances falls under the real device error margins, even though the reconstructed mesh approximates scene geometry from partial data. For further depths, signal degradation increases proportionally. This can be explained by the selected window size for frames accumulation, which determines the number of points available at any distance. For longer distances, the number of frames used during the 3D reconstruction phase becomes a key factor, as the sparsity of the cloud is dramatically reduced as the ego-vehicle moves forward. Therefore, for larger window sizes, the density of the point cloud would increase and, as a result, a higher resolution at distant areas would be achieved during the refinement step, reducing the reconstruction error.



(a)

(b)

Fig. 7. Percentage of correct points for increasing error thresholds at different distance ranges. White color means 100%, and black means 0%. (a) Original Velodyne HDL64E. (b) Virtual Velodyne HDL64E.

B. Detection benchmark

The second set of conducted experiments aims to compare the performance of the models mentioned above when the LiDAR data used for training is produced by a real and a simulated device. In order to ensure a fair comparison, only real point clouds are used as inputs at the validation stage. Thus, the degree of similarity between the real and synthetic signals can be assessed.

To this end, the well-known KITTI Object Benchmark has been used. Table I presents the average precision (AP) in 3D and Bird's Eye View (BEV) for the detection and classification of cars, pedestrians, and cyclists in three levels of difficulty. The intersection over union (IoU) thresholds used for computing the AP for each class are 0.7, 0.5, and 0.5, respectively.

As can be seen, the results for car detection in the three studied approaches are very similar to those obtained using real data, thus proving the suitability of the generated clouds to be used as training data for detecting these kinds of obstacles. For cyclists, the achieved performance is lower than when using the original data. This is mainly due to the fact that cyclists' shape variability is higher than vehicles, and here the pre-processing does not consider the movements of the rider or changes in the shape of the bike over time. It is noteworthy to mention that the results using real data also suffer this effect. For pedestrian 3D detection, the trend is similar to the one found in cyclist class. However, results on BEV are comparable to those produced by a model trained on real data, with the exception of AVOD. In fact, a slightly better AP BEV is obtained with SECOND when training with synthetic point clouds.

V. CONCLUSIONS

In this paper, we have presented a method for reconstructing the geometry of the scene using LiDAR information from different instants, which permits the generation of realistic synthetic point clouds as if they were captured by real devices. Both the pose and the intrinsic parameters of the

TABLE I

KITTI 3D OBJECT DETECTION RESULTS ON THE VALIDATION SET FOR MODELS TRAINED WITH REAL AND SYNTHETIC INPUTS

Class	Method	AP 3D (real)			AP BEV (real)			AP 3D (synthetic)			AP BEV (synthetic)		
		Easy	Moder.	Hard	Easy	Moder.	Hard	Easy	Moder.	Hard	Easy	Moder.	Hard
Car	Frustum	81.96	68.76	60.84	86.93	81.27	72.91	69.39	49.99	42.57	82.27	72.57	63.98
	AVOD	77.17	68.20	67.73	89.03	79.84	79.41	75.36	64.99	63.68	87.37	78.28	77.79
	SECOND	87.66	77.57	75.61	89.83	86.93	79.42	84.95	67.14	60.02	89.47	79.07	78.38
Ped.	Frustum	63.07	54.59	47.41	70,21	60.79	53.10	52.46	44.69	39.25	64.42	55.77	49.38
	AVOD	44.99	44.73	39.26	53.47	48.44	42.50	32.24	29.08	26.96	36.21	32.62	30.65
	SECOND	60.01	52.72	50.20	62.33	59.330	53.21	61.87	53.67	46.76	65.20	56.42	54.61
Cyc.	Frustum	74.98	54.19	49.84	78.66	57.51	53.69	57.98	42.43	40.34	64.63	48.32	45.79
	AVOD	63.00	39.57	38.72	64.58	40.41	39.86	40.57	28.67	25.93	44.51	29.94	26.89
	SECOND	79.01	60.29	55.11	80.58	62.23	60.80	65.99	52.10	46.94	67.64	53.77	48.54

virtual LiDAR device are configurable.

Conducted tests proved the utility of the approach. Although some of the information provided by the devices may not yet be available, e.g., multiple beams and intensity, the proposed solution is a step forward on the standardization of the current databases for any laser sensor and may facilitate the adoption of upcoming devices in the incipient and fast-changing market of LiDARs.

In future work, local information from the real LiDAR scans will be used to estimate an intensity value for the points of the synthetic clouds. Later, methods to allow the alignment of point clouds from multiple frames for deformable objects such as pedestrians will be studied. Moreover, depth prediction networks from monocular images might be explored, so that the presented approach can be used to add LiDAR modality to existing image-only datasets.

ACKNOWLEDGMENT

Research supported by the Spanish Government through the CICYT projects (TRA2016-78886-C3-1-R and RTI2018-096036-B-C21), and the Comunidad de Madrid through SEGVAUTO-4.0-CM (P2018/EMT-4362). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

REFERENCES

[1] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3D Proposal Generation and Object Detection from View Aggregation," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 5750–5757.

[2] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[3] 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.

[4] 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," *arXiv preprint arXiv:1903.11027*, 2019.

[5] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A Survey on 3D Object Detection Methods for Autonomous Driving Applications," *IEEE Transactions on Intelligent Transportation Systems (In press)*, 2019.

[6] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-View 3D Object Detection Network for Autonomous Driving," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6526–6534. [Online]. Available: http://arxiv.org/abs/1611.07759

[7] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. de la Escalera, "BirdNet: a 3D Object Detection Framework from LiDAR information," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Maui, Hawaii, USA: IEEE, 2018, pp. 3517–3523.

[8] B. Li, T. Zhang, and T. Xia, "Vehicle Detection from 3D Lidar Using Fully Convolutional Network," in *Robotics Science and Systems*, 2016.

[9] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 918–927.

[10] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4490–4499.

[11] B. Xu and Z. Chen, "Multi-level Fusion Based 3D Object Detection from Monocular Images," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, pp. 2345–2353.

[12] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Weinberger, "Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving," *arXiv:1812.07179 [cs.CV]*, 2018.

[13] X. Weng and K. Kitani, "Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud," *arXiv:1903.09847 [cs.CV]*, 2019.

[14] 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 International Conference on Multimedia Retrieval*, 2018, pp. 458–464.

[15] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual Worlds as Proxy for Multi-Object Tracking Analysis," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4340–4349.

[16] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented Reality Meets Computer Vision," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 961–972, 2018.

[17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proc. Conference on Robot Learning (CoRL)*, 2017, pp. 1–16.

[18] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3d reconstruction in real-time," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 963–968.

[19] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. S. Torr, "Urban 3D semantic modelling using stereo vision," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 580–585.

[20] C. Hoppe and S. Krömker, "Adaptive meshing and detail-reduction of 3d-point clouds from laser scans," *International archives of photogrammetry, remote sensing and spatial information sciences*, vol. 38, p. 5, 2009.

[21] T. Möller and B. Trumbore, "Fast, minimum storage ray/triangle intersection," in *ACM SIGGRAPH 2005 Courses*. ACM, 2005, p. 7.

[22] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *International Conference on 3D Vision (3DV)*, 2017.