Real3D-Aug: Point Cloud Augmentation by Placing Real Objects with Occlusion Handling for 3D Detection and Segmentation

Petr Šebek*, Šimon Pokorný*, Patrik Vacek, Tomáš Svoboda

Dept. of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague

Abstract—Object detection and semantic segmentation with the 3D lidar point cloud data require expensive annotation. We propose a data augmentation method that takes advantage of already annotated data multiple times. We propose an augmentation framework that reuses real data, automatically finds suitable placements in the scene to be augmented, and handles occlusions explicitly. Due to the usage of the real data, the scan points of newly inserted objects in augmentation sustain the physical characteristics of the lidar, such as intensity and raydrop. The pipeline proves competitive in training top-performing models for 3D object detection and semantic segmentation. The new augmentation provides a significant performance gain in rare and essential classes, notably 6.65% average precision gain for "Hard" pedestrian class in KITTI object detection or 2.14 mean IoU gain in the SemanticKITTI segmentation challenge over the state of the art.

I. INTRODUCTION

Accurate detection and scene segmentation are integral to any autonomous robotic pipeline. Perception and understanding is possible thanks to various sensors, such as RGB cameras, radars, and lidars. These sensors produce structural data and must be interpreted for the proper function of critical safety systems. We focus on lidars. Recently, the most promising way to process lidar data is to train deep neural networks [24], [31], [23] with full supervision, which require a large amount of annotated data.

The manual annotation process is very time and resource consuming. For example, to perform semantic segmentation on lidar point clouds, one needs to accurately label all the points in the scene as a specific object class. As a result, there is not enough annotated data to train large neural networks. Data augmentation is a way to effectively decrease the need for more annotated data by enriching the training set with computed variations of the data. This type of augmentation is usually achieved with geometrical transformations, such as translation, rotation, and rescale applied to the already labeled samples [14], [27], [29], [7].

In general, 3D point cloud augmentations [14], [12] have been much less researched than image augmentation techniques [27], [7], [5], [15]. For example, the aforementioned 3D point cloud augmentations only enrich geometrical features of the training samples, but do not create new scenarios with the previously unseen layout of objects. The lack of modeling realistic class population of the scenes is still a bottleneck of augmentation techniques. This problem can be addressed by augmentation that uses simulated virtual data and scene
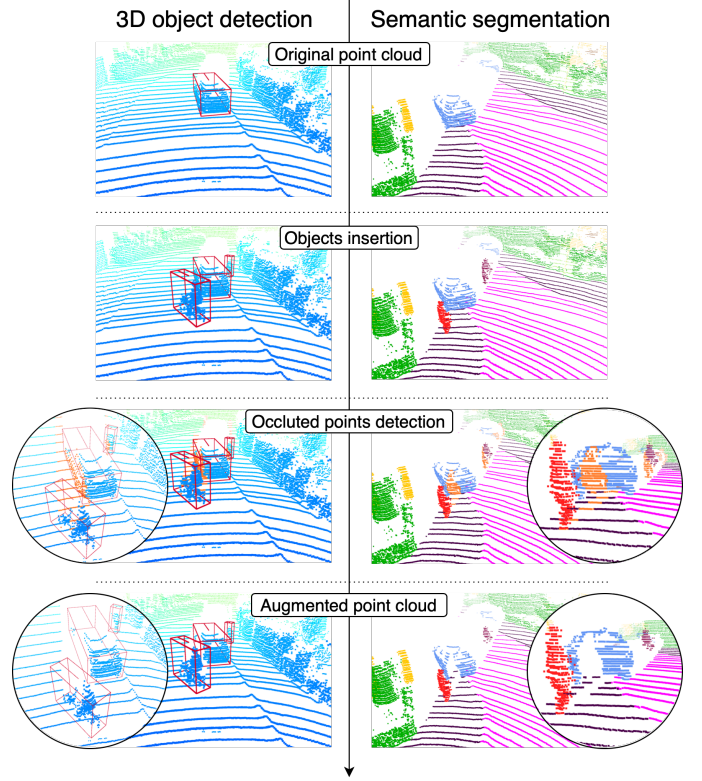

Fig. 1: We show examples of our augmentation method in 3D object detection and semantic segmentation. First, we insert objects one by one and then simulate their visibility to model realistic occlusions. Note the details of the scene (circled) and the detection of occluded orange points. After removal, we see the final augmented version of the point cloud in the last row

configurations. However, the effect of such data on training is low due to nonrealistic physical and visual features compared to real data.

We focus on improving the learning of 3D perception networks by enhancing the lidar data in autonomous driving scenarios with data augmentation. Depth information allows for per-object manipulation when augmenting the point clouds [12]. We take advantage of the spatial position of annotated objects and place them in different scenes, while handling occlusions and class-specific inhabitancy.

Our method segments the road and sidewalks for class-specific insertion. Next, the method exploits the bounding boxes of objects to avoid collisions. Compared to state-of-the-art LiDAR-Aug [12], which is suitable only for object

*Both authors contributed equally.
Corresponding author: vacekpa2@fel.cvut.cz

detection, our bounding box generation allows augmenting the semantic segmentation datasets and simulates realistic occlusions throughout the spherical projection. The inserted augmentations come from the same dataset and are placed to the same distance, ensuring natural reflection values and point distribution, including ray dropouts. We evaluate the proposed method on tasks of 3D object detection and semantic segmentation. Our contribution is twofold:

- We present a new augmentation framework suitable for *both* 3D object detection and semantic segmentation.
- We propose a novel way to *model occlusions* and physically consistent insertion of objects for augmentation.

We demonstrate the usefulness of our method on autonomous driving benchmarks and show improvement especially in rarely represented classes. The codes for our method will be publicly available.

II. RELATED WORK

A. *Data Augmentation*

One of the first approaches to augmenting lidar data was GT-Aug, which was published within the 3D detection model SECOND [28]. GT-Aug adds samples from the ground-truth database, which is precomputed before the training phase. The samples are randomly selected and inserted into the scene as is. If a collision occurs, the added object is simply removed. The visibility and occlusion handling of added scan points or the inserting strategy is not taken into account. Global data augmentations (Gl-Aug) [14] such as rotation, flip, and scale are commonly used in 3D point-cloud neural networks. These augmentations provide a different geometrical perspective, which supports the neural network with more diversity of training samples. An attempt to automate the augmentation strategy was proposed in [9], which narrows the search space based on previous training iterations. The state-of-the-art LiDAR-Aug [12] enriches the training data to improve the performance of the 3D detectors. Additional objects are rendered on the basis of CAD models. Simulations of intensity and raydrops are not discussed in the article. LiDAR-Aug [12] also simulates occlusion between additional objects and the rest of the scene, unlike GT-Aug [28].

B. *Data Simulators*

The recent progress in computer vision brought large neural networks with a large number of learnable parameters, often unable to reach saturation point with the size of current training sets. These models require training on a very large number of annotated examples. Commonly used solutions include synthetically generated data [25] or using game simulators such as Grand Theft Auto V, which was used to generate images for the semantic segmentation of ground truth [18]. Some simulators built on Unreal Engine, for example, Carla [11] are also used in autonomous driving research. However, the gap between real and synthetic data remains a great challenge [25]. One of the approaches to deal with the difference and portability to the real world is [19], [20], which can produce more realistic lidar data from simulation by learning GAN models.

C. *3D perception tasks*

Learning in the lidar point cloud domain poses challenges such as low point density in remote areas, unordered structure of the data, and sparsity due to the sensor resolution. Three common approaches to aggregation and learning the lidar features are voxel-based models [30], [28], re-projection of data into 2D structure [26], [16], and point cloud-based models [31], [23]. To show the ability to generalize, we evaluate our proposed method based on different model feature extractors, as well as on two different tasks of 3D object detection and semantic segmentation task.

One of the key aspects of our approach is placing the object in a realistic position by estimating the road for vehicle and cyclist insertions and the sidewalk for pedestrian insertion. Work [4] designed a fast fully convolutional neural network, which can predict the road from the bird's eye view projection of the scene. However, this approach does not handle occlusions. It does not predict the road behind obstacles, e.g. vehicles. Approaches [10], [2] can separate ground from non-ground points, which can even be improved by usage of The Jump-Convolution-Process [21]. However, this method filters out all ground points on roads and sidewalks. We need to distinguish them.

III. METHOD

Our augmentation method places additional objects into an already captured point cloud. The objects must be placed in adequate locations; therefore, the road and pedestrian area must be estimated (in Subsection III-A). The method avoids collisions between additional objects and objects that are in the original point cloud. We analyze overlapping bounding boxes. Therefore, we need to create bounding boxes for semantic datasets that come without object boxes (in Subsection III-B). More details on placing additional objects are given in Subsection III-C. Lastly, the method handles realistic occlusions between objects (in Subsection III-D). The overview of the proposed method is visualized in figure 4.

A. *Road estimation*

To place the new objects, we need to know where they realistically appear in the scene. This information may be given by HD maps [3], [6], if included in datasets; however, KITTI dataset does not provide them. We estimate valid roads and sidewalk areas for both tasks according to the pipeline described in Fig. 2. First, we pseudo-label 3D points by Cylinder3D [31], a state-of-the-art semantic segmentation neural network, which was pre-trained on the SemanticKITTI dataset [1]. The resulting predictions are then projected onto the 2D lidar (x, y) ground plane, discretized with a cell size resolution of 1×1 meter. Then we divide the space in the scene for the road (cyclist placement) and the sidewalk (pedestrian placement) as follows:

Road. To obtain a continuous *road area*, a morphological closing is used on the projection. We use a disk seed with dimension of three.

Pedestrian area. The estimate is based on the assumption that pedestrians are supposed to walk along the road border.

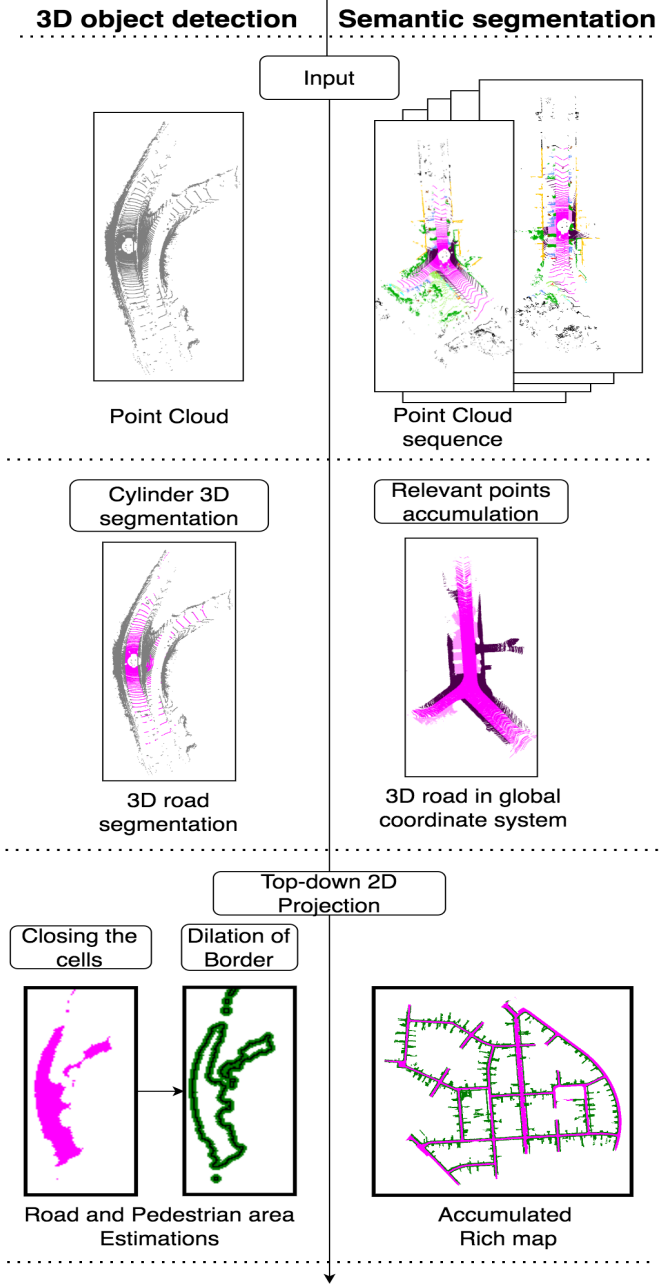**3D object detection** | **Semantic segmentation**

Fig. 2: Rich map generating. Road maps are created from points position and labels. Semantic datasets already contains labels of each road point, in case of detection dataset labels are pseudo-labeled by neural network [31]. We then project segmented points into 2D bird's eye view and acquire road and sidewalk map by morphological operations on the 2D projection, namely closing for the road and dilation of road boundary for the sidewalk–pedestrian area.

Cells closer than two pixels from the border of the road estimate are processed and subsequently dilated. We use disk seed with dimension of two.

SemanticKITTI contains poses of each point cloud in sequence. Therefore, road and sidewalk labels can be transformed into a global coordinate system and accumulated in space. The accumulated sequence of road and sidewalk labels
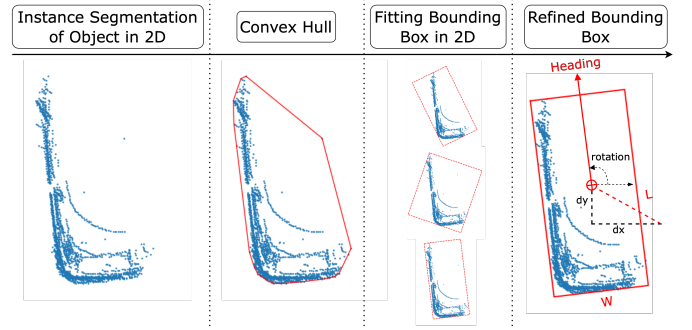

Fig. 3: Creation of the bounding box in Bird's Eye View around the car. First, convex hull is constructed around points, then we fit bounding box to estimate position x, y, dimensions *length*, *width*, *height*, and orientation *yaw*. The z is estimated as if the object touches the road without intersecting it.

leads to a more accurate estimation of the placement areas in the 2D lidar (x, y) ground plane projection. Accumulating multiple scans in one frame densifies the lidar point cloud and naturally reduces the need for morphological operations.

B. Creating of bounding boxes

For a collision-free placement of objects, the bounding boxes are required. The bounding box is parameterized by the center coordinates (x, y, z), size dimensions (l, w, h) and heading angle (yaw). For object detection in KITTI dataset, the bounding boxes are already provided as ground-truth labels. However, the SemanticKITTI dataset contains only the semantic label of the class with the instance of the object (each object in one frame has a different instance). We mitigate the absence of the bounding boxes by separating individual objects from the scene based on an instance and estimate bounding boxes, see Fig. 3. Modeling the bounding boxes is divided into three steps:

Wrapping. Object-labeled 3D Lidar points are projected to the ground plane. The 2D projected points are wrapped in a convex hull.

Smallest area. Assume the convex hull consists of n points. We construct $n - 1$ rectangles so that two neighboring points on the convex hull compose one side of the rectangle. The remaining sides of the rectangle are added to achieve the smallest area.

Refinement. Some objects may be represented by too few points. They are scanned at a great distance or are significantly occluded by closer objects. Bounding boxes may also be distorted by occlusions. We analyze the heights, widths and lengths of the bounding boxes in the KITTI dataset for classes "Car", "Pedestrian" and "Cyclists", which we use in Semantic KITTI. We obtain the distributions for each class and parameter. For each random variable, we calculate the lowest decile. The lowest decile values are the minimal threshold values of the bounding box. The maximal values of bounding boxes are set as the maximal values for the corresponding dimension that occurred in the KITTI data set.

For bicycle, motorcycle, motorcyclist and truck objects in the SemanticKITTI dataset, we do not have corresponding
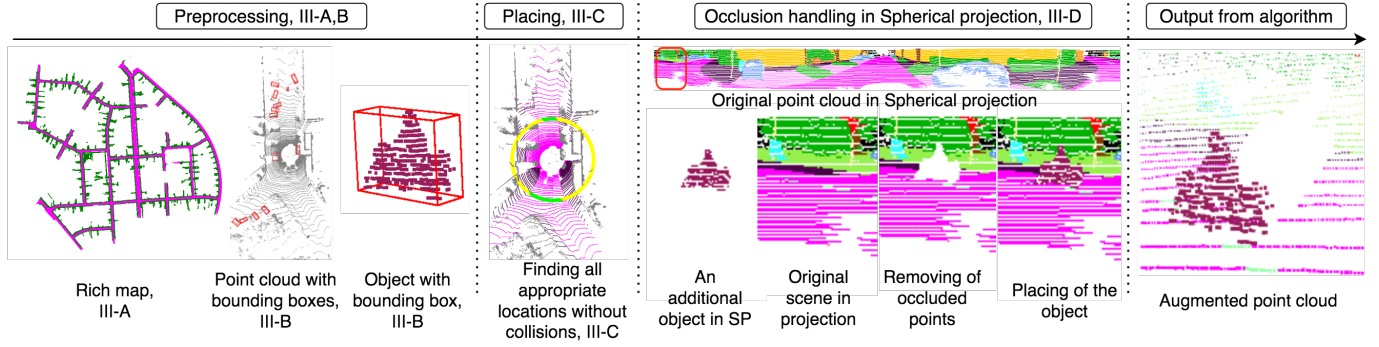
Fig. 4: Overview of proposed pipeline. We process the data in order to estimate all possible placements, all bounding boxes in the scene and augmenting objects from different frame. Possible placement of augmenting objects is a conjunction of the same depth as the cut out object (yellow circle) and suitable area from the map of possible insertions (green). Occlusion handling is performed in spherical projection. Result is re-projected to the scene to the 3D augmented point cloud.

statistics for bounding box dimensions, since they are not present in KITTI. Therefore, we do not apply any bounding box post-processing in this step for the aforementioned classes.

C. *Placing of objects*

For placing of one or multiple objects, we need their bounding box dimensions and yaw angles. Only points within the bounding boxes are used to augment different frames of the dataset. For the semantic segmentation datasets (task), these points are further filtered to have an appropriate label. In the case of the object detection datasets, points that are pseudo-labeled as road or sidewalk are removed to ensure that the cutout point cloud contains only the objects points.

To maintain the most realistic augmentation, our method places the object at the same distance with the same observation angle, which can be achieved by rotating its point cloud by the vertical z-axis of the frame origin. In this way, realistic object point density and lidar intensity are maintained due to the preserved range between the sensor and the object. It also keeps the same observation angle. Then, we consider the collision-free location of the insertion:

Location: Objects must be fully located on the appropriate surface. For vehicles and cyclists, it is on the road and for pedestrians, it is the pedestrian area. For each appropriate position, the z coordinate of the object is adjusted to ensure that the object touches the surface, according to the road prediction level.

Collision avoidance: At first, the sole bounding box belonging to the object is cut from the scene and placed in the augmented frame on the road level. For the insertion of vehicles and cyclists, the bounding box must not contain any point that is not predicted as road; same for pedestrians and the pedestrian area. Then, we check whether the inserted bounding box overlaps with each of the original boxes from the augmented scene and skip insertion when it does.

D. *Occlusion handling*

Algorithm 1 Occlusion handling

Input: Scene point-cloud, Scene projection, Object point-cloud, Object projection

Output: success, Scene point-cloud

1: sum_points ← 0
2: success ← False
3: **for** each pixel in object's spherical projection **do**
4:     **if** distance of object is smaller then in scene **then**
5:         Remove scene points in pixel (they are occluded)
6:         Add points projected to object s. p. pixel to scene
7:         sum_points ← sum_points + nbr of added points
8:     **end if**
9: **end for**
10: **if** sum_points > minimal point for class **then**
11:     success ← True
12: **end if**
13: **return** success, Scene

Data projection: The occlusion handling uses a spherical projection, similarly to [26], to solve realistic visibility after the additional object is placed. The spherical projection stores information about the minimal distance between the sensor and the points that were projected to the corresponding pixel. To correct the holes in the object, the projection is morphologically closed by a rectangular seed of dimension 5×3 (5 rows and 3 columns). The pixels closed by the seed are assigned the depth computed from the neighboring pixels as an average of the depths in that seed area. Morphological closing is computed separately for the scene and object.

Removing occluded points: The algorithm goes through every pixel in the spherical projection. Every pixel contains information about the distance of the point. All scene points more distant than the inserted point are removed, since they would be naturally occluded by the added object. as they are occluded by the placed object. Consequently, all object points, which were projected in the same pixel, are added to the scene point cloud. The procedure ends if a sufficient number of

points is added to the scene. A pseudocode of the algorithm is shown in Algorithm 1.

IV. EXPERIMENTS

A. Datasets and perception tasks

3D object detection: We use the KITTI 3D object detection benchmark. The data set consists of 7,481 training scenes and 7,518 testing scenes with three object classes: "car", "pedestrian", and "cyclist".

The test labels are not accessible, and access to the test server is limited. Therefore, we followed the methodology proposed by [12] and divided the training data set into training and validation parts, where the training set contains 3,712 and the validation 3,769 lidar samples [8]. The split of the dataset into training and validation was made consistent with standard KITTI format, i.e. with regard to avoid having similar frames and scenes in both sets. The evaluation was carried out on a validation set, where the labels are available, as was done in [12], [13].

For object detection, we consider only the two more rare and challenging classes, i.e. pedestrian and cyclist. From our point of view, the datasets provide a sufficient number of examples of cars for training.

A metric for conducting an evaluation is the standard average precision (AP) of 11 uniformly sampled recall values. We use the IoU threshold 50%, true positive predictions are considered bounding boxes with ground-truth overlaps greater than 50%. We denote AP for "Pedestrian" as $\mathbf{AP_{Ped}50}(\%)$ and $\mathbf{AP_{Cyc}50}(\%)$ for "Cyclist". The difficulties of the predictions are divided on the basis of the sizes of the bounding box, occlusion, and truncation into "Easy", "Moderate" and "Hard" as required by the [13] benchmark.

Semantic segmentation: We use the SemanticKITTI [1] benchmark. The dataset is an extension of the original KITTI [13] benchmark with dense point-wise annotations provided for each $360°$ field-of-view frame. In general, the dataset offers 23,201 3D scans for training and 20,351 for testing. The training data set was divided into training and validation parts with 19 annotated classes.

The intersection-over-union, $\mathrm{IoU} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP} + \mathrm{FN})$, was used for comparison. Performance is evaluated for each individual class, as well as the average (mIoU) for all classes.

B. 3D perception models

We tested the augmented data on two *3D object detection* models, each based on a different type of feature extractor backbone. *PV-RCNN* [22] is a 3D object detection model that combines a 3D voxel convolutional neural network with a pointnet-based set abstraction approach [17]. The second is PointPillar [24], which encodes the point cloud in vertical pillars. The pillars are later transformed into 3D pseudo-image features.

For *segmentation task*, we use the SPVNAS [23] multiclass detector. The neural network achieves significant computation reduction due to the sparse Point-Voxel convolution and holds

the fourth place on the competitive SemanticKITTI leaderboard and the second among those with available implementation.

Each neural network was set to the default parameters proposed by the authors of the architectures, with its performance reported on KITTI 3D benchmark and SemanticKITTI. We trained each neural network three times for object detection and five times for semantic segmentation. Average performance was considered as the final score of the method.

C. Augmentations

All augmentations were trained with the same hyperparameters to ensure a fair comparison between methods. The approach of GT-Aug was performed with information of the precomputed planes, which is an approximation of the ground from the KITTI dataset. This step should ensure that the inserted objects lie on the ground. For our proposed augmentation method, we choose to add objects with a zero-occlusion KITTI label only (Easy). Some cases are naturally transformed into other difficulties (Moderate and Hard) by newly created occlusions.

For global augmentation of the scenes, we used uniformly distributed scaling of the scene in the range $[0.95, 1.05]$, rotation around the z-axis (vertical axis) in the range $[-45°, 45°]$ and random flipping over the x-axis from the point cloud as in [14], [12].

The maximum number of added objects in semantic segmentation was set to 10 per scene, and object class is selected randomly each time of the insertion. In Table I we can see the number of objects added per scene and the minimal number of points for insertion.

TABLE I: Number of insertions per class on Semantic KITTI dataset

Class	Min number of points	Added objects per scene
Bicycle	10	1.67
Motorcycle	10	1.66
Truck	40	1.59
Person	20	1.56
Bicyclist	30	1.66
Motorcyclist	30	1.66

D. Evaluation

We compare our method (Real3D-Aug) with copy-and-paste augmentation (GT-Aug) [28] and with state-of-the-art LiDAR-Aug augmentation [12]. In the Real3D-Aug multiclass (mc),

TABLE II: Object detection results with PV-RCNN. Our method achieves the best results in categories "pedestrian" and "easy cyclists". (mc) abbreviates multiclass

| Method | $\mathbf{AP_{Ped}\,50}(\%)$ | | | $\mathbf{AP_{Cyc}\,50}(\%)$ | | |
	Easy	Mod	Hard	Easy	Mod	Hard
w/o Object-Aug	65.92	59.14	54.51	76.80	59.36	56.61
GT-Aug [28]	65.69	59.33	54.78	88.30	**72.55**	**67.79**
LiDAR-Aug [12]	65.05	58.90	55.52	N/A	N/A	N/A
Real3D-Aug (mc)	**73.57**	**66.55**	**62.17**	**92.69**	65.06	63.43

TABLE III: Semantic segmentation on SemanticKITTI. Comparison of our method with the global augmentation baseline. Both methods use SPVNAS [23] neural network. The reported results are averaged over five runs. The augmented categories are typeset in *italics*. We observe a performance gain in each of them except for one: trucks. Improvement is especially notable in the motorcyclist class, which contains only a few training examples in the dataset with only global augmentations.

	mIoU	car	*bicycle*	*motorcycle*	*truck*	other-vehicle	*person*	*bicyclist*	*motorcyclist*	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
w/o Obj-Aug	60.62	95.47	29.64	58.16	**64.22**	47.69	66.24	79.14	0.04	**93.06**	**48.52**	**80.20**	1.72	**89.75**	**58.67**	87.88	67.07	73.40	63.51	47.34
Real3D-Aug	**62.76**	**95.93**	**44.13**	**73.41**	49.24	**48.43**	**70.34**	**85.45**	**12.01**	92.84	45.66	79.66	**2.91**	89.36	56.96	**89.18**	**67.61**	**76.72**	**63.73**	**48.88**

we added 4.7 pedestrians and 6.7 cyclists on average per scene. All methods were trained with global augmentations [14] if not stated otherwise.

In Table II we show the results of LiDAR-Aug with PV-RCNN. The numbers are taken from the original paper due to the unpublished codes and the lack of technical details about their CAD model and ray-drop characteristic. In the original article, LiDAR-Aug was trained under unknown hyperparameters and was not applied to the cyclist category. Our method surpasses the LiDAR-Aug in pedestrian class by a large margin despite all the difficulties. Both GT-Aug and Real3D-Aug achieve a significant improvement in performance. Real3D-Aug achieves a significant improvement with PV-RCNN in the pedestrian class, where we achieve 15.4%, 10.96%, and 7.87% improvement in Easy, Moderate, and Hard difficulty, and GT-Aug achieves 7.52%, 3.74%, and 0.48% improvement compared to the model without (w/o) any object augmentations.

In the semantic segmentation task, we achieve an increase of 2.14 in mean IoU compared to the common augmentation technique [14], see Table III. We observe an increase in IoU of all classes added, except for the truck category. Our method also increases the car category (not augmented) since we add more negative examples to other similar classes. We are not comparing with GT-Aug [28] and LiDAR-Aug [12] in the semantic segmentation task. The aforementioned methods were not designed for segmentation, whereas our method allows for augmenting both tasks.

E. *Ablation study of object detection*

In Tables IV and V we show the influence of adding a single object to the scene in comparison to GT-Aug. Each configuration is named after the added class, and the lower index indicates the average number of objects added per scene. We can see that, in the case of PointPillar, adding only one class decreases performance on the other classes. We suspect that it is conditioned by similarities between classes. For example, pedestrians and bicycles are simultaneously present in the class "cyclist". Therefore, it is beneficial to add both classes simultaneously. In case of PV-RCNN, the addition of one class improves the performance of both.

V. CONCLUSION

We propose an object-centered point cloud augmentation technique for 3D detection and semantic segmentation tasks.

TABLE IV: Real3D-Aug Object detection results with Point-Pillar architecture based on number of inserted classes.

Augmentation	AP_{Ped} 50(%)			AP_{Cyc} 50(%)		
	Easy	Mod	Hard	Easy	Mod	Hard
GT-Aug	54.52	49.04	45.49	**77.64**	**61.30**	**58.15**
Real3D-Aug (Ped_1)	**55.72**	51.30	47.47	46.33	33.84	32.47
Real3D-Aug (Cyc_1)	46.87	44.17	41.77	72.65	52.71	49.04
Real3D-Aug (mc)	55.50	**52.00**	**49.03**	76.82	52.74	50.18

TABLE V: Real3D-Aug object detection results with PV-RCNN based on number of inserted classes.

Augmentation	AP_{Ped} 50(%)			AP_{Cyc} 50(%)		
	Easy	Mod	Hard	Easy	Mod	Hard
GT-Aug	65.69	59.33	54.78	88.30	**72.55**	**67.79**
Real3D-Aug (Ped_1)	70.96	66.63	61.14	78.97	63.47	57.31
Real3D-Aug (Cyc_1)	65.63	59.14	57.47	82.79	63.69	62.39
Real3D-Aug (mc)	**73.57**	**66.55**	**62.17**	**92.69**	65.06	63.43

Our method improves performance on important and rarely occurring classes, e.g. pedestrian, cyclist, motorcyclist, and others. The method outperforms state-of-the-art on well-established autonomous driving benchmarks. Our method is self-contained and requires only 3D data. All augmentations can be preprocessed, so it does not increase time during training. One way to further improve the method is to incorporate more informative selection of placements based on the uncertainty of the detection model.

REFERENCES

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019. 2, 5

[2] Igor Bogoslavskyi and Cyrill Stachniss. Efficient Online Segmentation for Sparse 3D Laser Scans. *Photogrammetrie - Fernerkundung - Geoinformation*, 85:41–52, 12 2016. 2

[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 2

[4] Luca Caltagirone, Samuel Scheidegger, Lennart Svensson, and Mattias Wahde. Fast LIDAR-based road detection using fully convolutional neural networks. In *2017 ieee intelligent vehicles symposium (iv)*, pages 1019–1024. IEEE, 2017. 2

[5] Nino Cauli and Diego Reforgiato Recupero. Survey on videos data augmentation for deep learning models. *Future Internet*, 14(3), 2022. 1

[6] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3D Tracking and Forecasting With Rich Maps. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8740–8749, 2019. 2

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020. 1

[8] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals using stereo imagery for accurate object class detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(5):1259–1272, 2017. 5

[9] Shuyang Cheng, Zhaoqi Leng, Ekin Dogus Cubuk, Barret Zoph, Chunyan Bai, Jiquan Ngiam, Yang Song, Benjamin Caine, Vijay Vasudevan, Congcong Li, Quoc V. Le, Jonathon Shlens, and Dragomir Anguelov. Improving 3D Object Detection through Progressive Population Based Augmentation. In *ECCV*, 2020. 2

[10] Phuong Chu, Seoungjae Cho, Simon Fong, and Kyungeun Cho. Enhanced ground segmentation method for Lidar point clouds in human-centric autonomous robot systems. *Human-centric Computing and Information Sciences*, 9, 12 2019. 2

[11] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 2

[12] Jin Fang, Xinxin Zuo, Dingfu Zhou, Shengze Jin, Sen Wang, and Liangjun Zhang. LiDAR-Aug: A General Rendering-based Augmentation Framework for 3D Object Detection. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4708–4718, 2021. 1, 2, 5, 6

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 5

[14] Martin Hahner, Dengxin Dai, Alexander Liniger, and Luc Van Gool. Quantifying Data Augmentation for LiDAR based 3D Object Detection. *arXiv:2004.01643*, 1:1–7, 2020. 1, 2, 5, 6

[15] Yen-Cheng Liu, Chih-Yao Ma, Zijian He, Chia-Wen Kuo, Kan Chen, Peizhao Zhang, Bichen Wu, Zsolt Kira, and Peter Vajda. Unbiased Teacher for Semi-Supervised Object Detection. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 1

[16] Andres Milioto, Ignacio Vizzo, Jens Behley, and C. Stachniss. Rangenet ++: Fast and accurate lidar semantic segmentation. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220, 2019. 2

[17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 5

[18] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for Data: Ground Truth from Computer Games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, Cham, 2016. Springer International Publishing. 2

[19] Ahmad El Sallab, Ibrahim Sobh, Mohamed Zahran, and Nader Essam. LiDAR sensor modeling and data augmentation with GANs for autonomous driving. *arXiv preprint arXiv:1905.07290*, 2019. 2

[20] Ahmad El Sallab, Ibrahim Sobh, Mohamed Zahran, and Mohamed Shawky. Unsupervised neural sensor models for synthetic lidar data augmentation. *arXiv preprint arXiv:1911.10575*, 2019. 2

[21] Zhihao Shen, Huawei Liang, Linglong Lin, Zhiling Wang, Weixin Huang, and Jie Yu. Fast Ground Segmentation for 3D LiDAR Point Cloud Based on Jump-Convolution-Process. *Remote Sensing*, 13(16), 2021. 2

[22] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 5

[23] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision*, pages 685–702. Springer, 2020. 1, 2, 5, 6

[24] Jiayin Tu, Ping Wang, and Fuqiang Liu. PP-RCNN: Point-Pillars Feature Set Abstraction for 3D Real-time Object Detection. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021. 1, 5

[25] Patrik Vacek, Otakar Jašek, Karel Zimmermann, and Tomáš Svoboda. Learning to Predict Lidar Intensities. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–9, 2021. 2

[26] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *European Conference on Computer Vision*, pages 1–19. Springer, 2020. 2, 4

[27] Xianfa Xu, Zhe Chen, and Fuliang Yin. Cutresize: Improved data augmentation method for rgb-d object recognition. *IEEE Robotics and Automation Letters*, 7(1):183–190, 2022. 2

[28] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2, 5, 6

[29] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d: Self-training for unsupervised domain adaptation on 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10368–10378, June 2021. 1

[30] Y. Zhou and O. Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society. 2

[31] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9934–9943, 2021. 1, 2, 3