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

Petr Šebek[1,†], Šimon Pokorný[1,†], Patrik Vacek[1,*] and Tomáš Svoboda[1]

[1] *Vision for Robotics and Autonomous Systems, 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.

Keywords

LiDAR, pointclouds, augmentation, semantic segmentation, object detection

1. Introduction

Accurate detection and scene segmentation are integral to any autonomous robotic pipeline. Perception and understanding are 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 [1, 2, 3] with full supervision, which requires 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 [4, 5, 6, 7].

In general, 3D point cloud augmentations [4, 8] have been much less researched than image augmentation techniques [5, 7, 9, 10]. For example, the aforementioned 3D point cloud augmentations only enrich the geometrical features of the training samples but do not create new scenarios with the previously unseen layout of objects. The lack of modeling a 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 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 LiDAR data in autonomous driving scenarios with data augmentation. Depth information allows for per-object manipulation when augmenting the point clouds [8]. We take advantage of the spatial position of annotated objects and place them in different scenes while handling occlusions and class-specific inhabitancy, see Figure 1.

26th Computer Vision Winter Workshop, Robert Sablatnig and Florian Kleber (eds.), Krems, Lower Austria, Austria, Feb. 15-17, 2023

*Corresponding author.

†These authors contributed equally.

sebekpe1@fel.cvut.cz (P. Šebek); pokorsi1@fel.cvut.cz (Pokorný); vacekpa2@fel.cvut.cz (P. Vacek); svobodat@fel.cvut.cz (T. Svoboda)
http://cmp.felk.cvut.cz/~vacekpa2/ (P. Vacek); http://cmp.felk.cvut.cz/~svoboda/ (T. Svoboda)
0000-0001-8587-5364 (P. Šebek); 0000-0002-7812-5634 (Pokorný); 0000-0003-3001-1483 (P. Vacek); 0000-0002-7184-1785 (T. Svoboda)

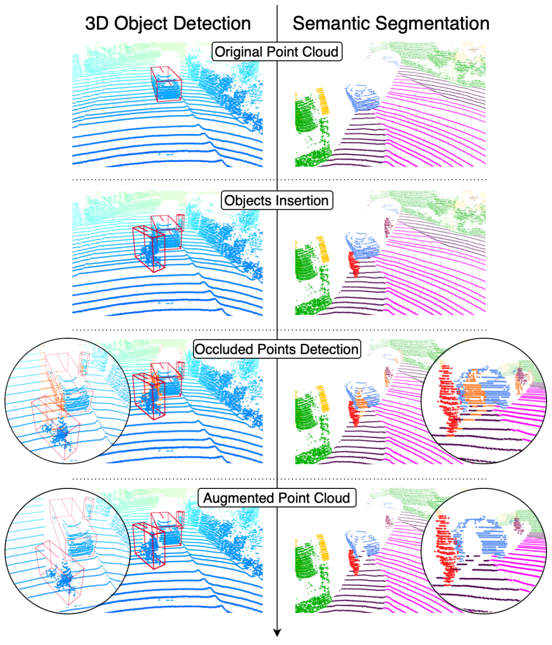

Figure 1: We show examples of our augmentation method in 3D object detection and semantic segmentation. First, we insert objects one by one and then simulate their visibility to model realistic occlusions. Note the details of the scene (circled) and the detection of occluded orange points. After removal,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

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 [8], which is suitable only for object 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 at 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 are publicly available[1].

2. Related Work

2.1. Data Augmentation

One of the first approaches to augmenting LiDAR data was GT-Aug, which was published within the 3D detection model SECOND [11]. 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) [4] 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 [12], which narrows the search space based on previous training iterations. The state-of-the-art LiDAR-Aug [8] 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 [8] also simulates occlusion between additional objects and the rest of the scene, unlike GT-Aug [11]. Recent method [13], similar to our one, also focuses on inserting objects into point clouds. The main difference between the methods is in the real visibility simulation. Approach [13] upsamples the number of points in the sample, which are then projected into a range image, where visible points are selected and then sparsed. From our point of view, this approach does not consider possible raydrop on objects located between the ego and the inserted sample. It can cause parts of the inserted sample to be falsely visible because some LiDAR beams could drop out from the obstacle and create holes in the range image.

2.2. Data Simulators

The recent progress in computer vision brought large neural networks with a large number of learnable parameters, often unable to reach a saturation point with the size of current training sets. These

[1] https://github.com/ctu-vras/pcl-augmentation

models require training on a very large number of annotated examples. Commonly used solutions include synthetically generated data [14] or using game simulators such as Grand Theft Auto V, which was used to generate images for the semantic segmentation of ground truth [15]. Some simulators built on Unreal Engine, for example, Carla [16] are also used in autonomous driving research. However, the gap between real and synthetic data remains a great challenge [14]. One of the approaches to deal with the difference and portability to the real world is [17, 18], which can produce more realistic LiDAR data from simulation by learning GAN models.

2.3. 3D Perception Tasks

Learning in the LiDAR point cloud domain poses challenges, such as low point density in regions at the far end of the FOV, the 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 [19, 11], re-projection of data into 2D structure [20, 21], and point cloud-based models [2, 3]. To show the ability to generalize, we evaluate our proposed method based on different model feature extractors and on two tasks of 3D object detection and semantic segmentation.

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. Recent research has shown, that a fast, fully convolutional neural network can predict the road from the bird's eye view projection of the scene [22]. However, this method does not handle occlusions, i.e. it does not predict the road behind obstacles, e.g. vehicles. Non-learnable methods proposed in [23, 24] can separate ground from non-ground points, which can be further improved by utilizing the Jump-Convolution-Process [25]. All these methods (and other established types like RANSAC, PCA, and height thresholding) filter out all ground points regardless of class road or sidewalk. In our setup, we need to distinguish them, so we rely on the segmentation network learned from the dataset.

3. 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 3.1). 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 3.2). More details on placing additional objects are given in Subsection 3.3. Lastly, the method handles realistic occlusions between objects (in Subsection 3.4). The overview of the proposed method is visualized in Figure 2.

3.1. 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 [26, 27] if included in datasets; however, KITTI dataset [28] does not provide them. We estimate valid roads and sidewalk areas for both tasks according to the pipeline described in Figure 3. First, we pseudo-label 3D points by Cylinder3D [2], a state-of-the-art semantic segmentation neural network, which was pre-trained on the SemanticKITTI dataset [29]. 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 a dimension of three.

Pedestrian area: The estimate is based on the assumption that pedestrians are supposed to walk along the road border. Cells closer than two pixels from the border of the road estimate are processed and subsequently dilated. We use a disk seed with a 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 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.

3.2. 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 the KITTI dataset, the

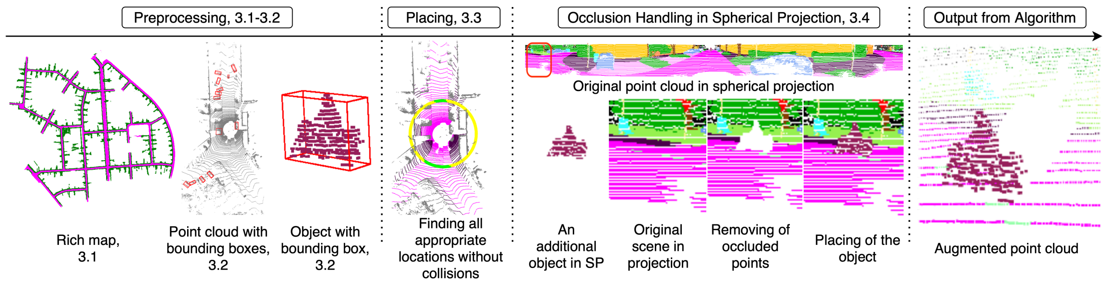
Preprocessing, 3.1-3.2
Placing, 3.3
Occlusion Handling in Spherical Projection, 3.4
Output from Algorithm
Rich map, 3.1
Point cloud with bounding boxes, 3.2
Object with bounding box, 3.2
Finding all appropriate locations without collisions
Original point cloud in spherical projection
An additional object in SP
Original scene in projection
Removing of occluded points
Placing of the object
Augmented point cloud

Figure 2: Overview of the proposed pipeline. We process the data in order to estimate all possible placements, all bounding boxes in the scene, and augmenting objects from different frames. The possible placement of augmenting objects is a conjunction of the same depth as the cut-out object (yellow circle) and a suitable area from the map of possible insertions (green). Occlusion handling is performed in spherical projection. The result is re-projected to the scene to the 3D augmented point cloud.

bounding boxes are already provided as ground-truth labels. However, the SemanticKITTI dataset contains only the semantic label of the class together 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 Figure 4. In case of the absence of instance labels, we would cluster the semantic segmentation points to get the instances via density-based clustering. In the case of close-by segmentation, more than one instance can be inserted without damaging the consistency of our approach.

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: Too few points may represent some objects. 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 minimum 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 statistics for bounding box dimensions since they are not present in KITTI. Therefore, the limits were hand-crafted from the first 100 generated samples from SemanticKitti. We also used the first decile, but with a 10% margin of safety.

3.3. Placing of Objects

Placing one or multiple objects requires knowing the 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 the road or sidewalk classes are removed to ensure that the cutout point cloud contains only the object points.

To maintain the most realistic augmentation, our method places the object at the same distance with the same observation angle. It can be achieved by rotating its point cloud by the vertical z-axis of the frame origin. 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. We place vehicles and cyclists on the streets and pedestrians on sidewalks. Thought pedestrians can move on the streets as well, we do not observe this occurance in the evaluation datasets and therefore do not consider it during

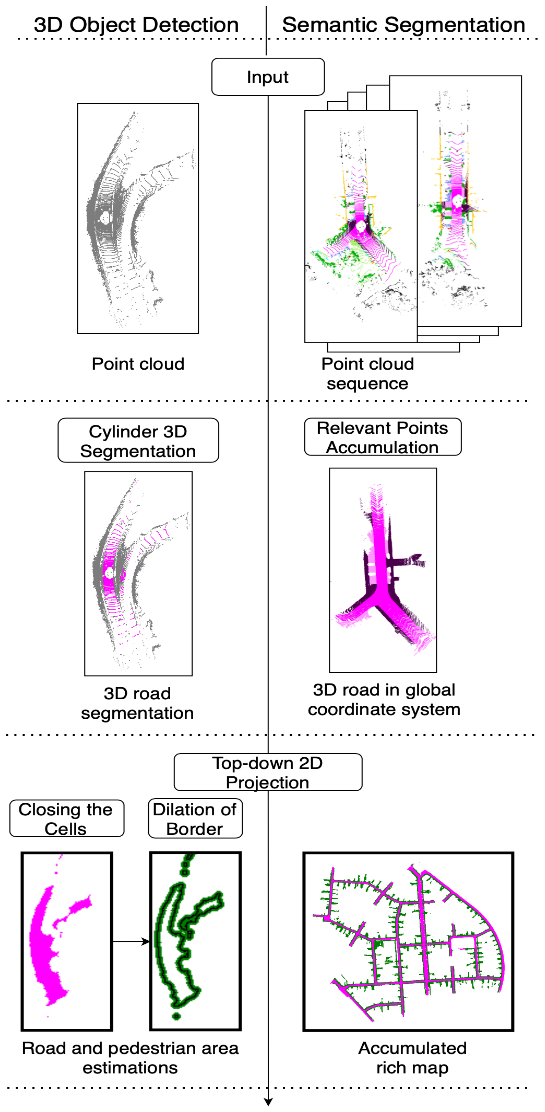

Figure 3: Rich map generating. Road maps are created from points' positions and labels. Semantic datasets already contain labels for each road point, in the case of the detection dataset, labels are pseudo-labeled by neural network [2]. We then project segmented points into a 2D bird's eye view and acquire road and sidewalk maps by morphological operations on the 2D projection, namely closing for the road and dilation of road boundary for the sidewalk–pedestrian area.

insertion. 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

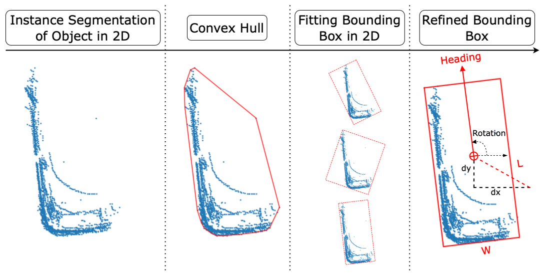
Instance Segmentation of Object in 2D
Convex Hull
Fitting Bounding Box in 2D
Refined Bounding Box
Heading
Rotation
dy
dx
L
W

Figure 4: Creation of the bounding box in Bird's Eye View around the car. First, a convex hull is constructed around points; then we fit a 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.

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 other than 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.

3.4. Occlusion Handling

By inserting objects into the scene, we model consistent occlusions in the point cloud from newly added points. We consider the occlusion of a newly inserted object by original points closer to the LiDAR sensor, as well as the occlusions caused by the inserted object itself.

Data projection: The occlusion handling uses a spherical projection, similarly to [20], to solve realistic visibility after the additional object is placed. The spherical projection stores the minimal distance between the sensor and the points 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 three 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.

Algorithm 1 Occlusion handling

```
Input: Scene point-cloud $\mathcal{P}$, Scene projection, Object point-cloud, Object projection
Output: success, Scene point-cloud
 1: point_counter ← 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:         point_counter ← point_counter + nbr of added points
 8:     end if
 9: end for
10: if point_counter > minimal point for class then
11:     success ← True
12: end if
13: return success, Scene
```

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 algorithm also returns boolean values, which represent if the number of added sample points exceeds the threshold for a given class. We used this to prevent super hard cases, with only, e.g., three visible points from the object. A pseudocode of the algorithm is shown in Algorithm 1.

4. Experiments

In this section, we show the experimental evaluation of our method on KITTI and SemanticKITTI datasets with comparison to other types of data augmentation such as Global Augmentation [4], Ground Truth insertion[11] and LiDAR-Aug [8]. We experiment with two neural networks for each task.

4.1. 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 [8] 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 [30]. The split of the dataset into training and validation was made consistent with the standard KITTI format, i.e., with regard to avoiding 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 [8, 28]. For object detection, we consider all possible classes, i.e., cars, pedestrians, and cyclists.

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% for pedestrians and cyclists. For cars, the 70% threshold was used. We denote AP for "Pedestrian" as $\mathbf{AP}_{\text{Ped}}\mathbf{50}(\%)$, $\mathbf{AP}_{\text{Cyc}}\mathbf{50}(\%)$ for "Cyclist" and $\mathbf{AP}_{\text{Car}}\mathbf{70}(\%)$ for "Cars". The difficulties of the predictions are divided based on the sizes of the bounding box, occlusion, and truncation into "Easy", "Moderate", and "Hard", as required by the [28] benchmark.

Semantic segmentation: We use the SemanticKITTI [29] benchmark. The dataset is an extension of the original KITTI [28] benchmark with dense point-wise annotations provided for each 360° field-of-view frame. The dataset generally 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.

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

4.2. 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* [31] is a 3D object detection model that combines a 3D voxel convolutional neural network with a pointnet-based set abstraction approach [32]. The second is PointPillar [1], which encodes the point cloud in vertical pillars. The pillars are later transformed into 3D pseudo-image features.

For *segmentation task*, we use Cylinder3D [2] and SPVNAS [3] multiclass detector. Cylinder3D [2] is the top-performing architecture on the Semantic KITTI dataset with public codes. SPVNAS [3] achieves significant computation reduction due to the sparse Point-Voxel convolution and holds the fourth place on the competitive SemanticKITTI leaderboard right behind Cylinder3D [2].

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

Table 1
Semantic segmentation on SemanticKITTI. Comparison of our method with the global augmentation baseline. Both methods are evaluated using SPVNAS [3] and Cylinder3D [2] architectures. The reported results are averaged over five runs for SPVNAS, and only one run was performed for Cylinder3D due to the large training time. The augmented categories are denoted by * for SPVNAS and by ** for Cylinder3D. 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
SPVNAS 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
SPVNAS w 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**
Cylinder3D w/o Obj-Aug	58.83	95.63	42.67	59.37	33.28	41.03	67.15	78.83	0.00	92.48	**42.24**	78.49	**0.02**	89.86	57.32	**87.43**	**67.23**	73.70	**65.03**	45.93
Cylinder3D w Real3D-Aug	**63.00**	**96.27**	**50.47**	**71.29**	**64.28**	**50.20**	**69.78**	**88.84**	**12.66**	**93.37**	35.43	**79.81**	0.00	**90.60**	**59.86**	87.42	59.02	**73.71**	64.83	**49.24**

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.

4.3. 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 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 [4, 8].

The maximum number of added objects in semantic segmentation was set to 10 per scene, and the object class is selected randomly (uniform distribution) each time of the insertion.

4.4. Evaluation

We compare our method (Real3D-Aug) with copy-and-paste augmentation (GT-Aug) [11] and with state-of-the-art LiDAR-Aug augmentation [8]. In the Real3D-Aug multiclass (mc), we added 4.7 pedestrians and 6.7 cyclists on average per scene. All methods were trained with global augmentations [4] if not stated otherwise.

In Table 2 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 the pedestrian class by a large margin despite all the difficulties. Both GT-Aug and Real3D-Aug achieve significant performance improvement. 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 augmentation. Our method also slightly improves the performance on the car, but Lidar-Aug and GT-Aug overcome the method.

Table 2
Object detection results with PV-RCNN. Our method achieves the best results in the categories "pedestrian" and "easy cyclists". (mc) abbreviates multiclass

	AP$_{Car}$ 70(%)			**AP$_{Ped}$ 50**(%)			**AP$_{Cyc}$ 50**(%)		
Method	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
w/o Object-Aug	87.77	78.12	76.88	65.92	59.14	54.51	76.80	59.36	56.61
GT-Aug [11]	89.17	81.92	78.78	65.69	59.33	54.78	88.30	**72.55**	**67.79**
LiDAR-Aug [8]	**90.18**	**84.23**	**78.95**	65.05	58.90	55.52	N/A	N/A	N/A
Real3D-Aug (mc)	88.70	78.63	78.09	**73.57**	**66.55**	**62.17**	**92.69**	65.06	63.43

In Table 1 we show the results for SPVNAS [3] and Cylinder3D [2] architecture. In the semantic segmentation task, we increased the mean IoU for both networks.

We are not comparing with GT-Aug [11] and

LiDAR-Aug [8] in the semantic segmentation task. The methods above were not designed for segmentation, whereas our method allows for augmenting both tasks.

In the semantic segmentation task for SPVNAS, we achieve an increase of 2.14 in mean IoU compared to the common augmentation technique [4], see Table 1. We observe an increased IoU of all classes added, except for the truck category. With the Cylinder3D network, the increment can be seen in the IoU of all added classes. Our method also increases the performance on not augmented classes since we add more negative examples to other similar classes.

4.5. Ablation Study of Object Detection

In Tables 3 and 4 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 in the other classes. We suspect that it is caused 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 the case of PV-RCNN, the addition of one class improves the performance of both.

Table 3
Real3D-Aug Object detection results with PointPillar architecture based on number of inserted classes.

Augmentation	$\mathbf{AP}_{\text{Ped}}$ **50**(%)			$\mathbf{AP}_{\text{Cyc}}$ **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 4
Real3D-Aug object detection results with PV-RCNN based on the number of inserted classes.

Augmentation	$\mathbf{AP}_{\text{Ped}}$ **50**(%)			$\mathbf{AP}_{\text{Cyc}}$ **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

5. Conclusion

We propose an object-centered point cloud augmentation technique for 3D detection and semantic segmentation tasks. Our method improves performance on important and rarely occurring classes, e.g. pedestrian, cyclist, motorcyclist, and others. Our method is self-contained and requires only 3D data. All augmentations can be preprocessed, so it does not increase the training time. One way to further improve the method is to incorporate a more informative selection of placements based on the uncertainty of the detection model.

Acknowledgments

This work was supported in part by OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics", and by Grant Agency of the CTU Prague under Project SGS22/111/OHK3/2T/13. Authors want to thank colleagues from Valeo R&D for discussions and Valeo company for a support.

References

[1] J. Tu, P. Wang, F. Liu, PP-RCNN: Point-Pillars Feature Set Abstraction for 3D Real-time Object Detection, in: IEEE International Joint Conference on Neural Networks (IJCNN), 2021.

[2] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, D. Lin, Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

[3] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, S. Han, Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution, in: European Conference on Computer Vision (ECCV), 2020.

[4] M. Hahner, D. Dai, A. Liniger, L. V. Gool, Quantifying Data Augmentation for LiDAR based 3D Object Detection, arXiv:2004.01643 (2020).

[5] X. Xu, Z. Chen, F. Yin, CutResize: Improved data augmentation method for RGB-D Object Recognition, IEEE Robotics and Automation Letters (RA-L) (2022).

[6] J. Yang, S. Shi, Z. Wang, H. Li, X. Qi, ST3D: Self-Training for Unsupervised Domain Adaptation on 3D Object Detection, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

[7] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A Simple Framework for Contrastive Learning of Visual Representations, in: International Conference on Machine Learning (ICML), 2020.

[8] J. Fang, X. Zuo, D. Zhou, S. Jin, S. Wang, L. Zhang, LiDAR-Aug: A General Rendering-based Augmentation Framework for 3D Object Detection, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

[9] N. Cauli, D. Reforgiato Recupero, Survey on Videos Data Augmentation for Deep Learning Models, Future Internet (2022).

[10] Y.-C. Liu, C.-Y. Ma, Z. He, C.-W. Kuo, K. Chen, P. Zhang, B. Wu, Z. Kira, P. Vajda, Unbiased Teacher for Semi-Supervised Object Detection, in: International Conference on Learning Representations (ICLR), 2021.

[11] Y. Yan, Y. Mao, B. Li, Second: Sparsely Embedded Convolutional Detection, Sensors (2018).

[12] S. Cheng, Z. Leng, E. D. Cubuk, B. Zoph, C. Bai, J. Ngiam, Y. Song, B. Caine, V. Vasudevan, C. Li, Q. V. Le, J. Shlens, D. Anguelov, Improving 3D Object Detection through Progressive Population Based Augmentation, in: European Conference on Computer Vision (ECCV), 2020.

[13] Y. Ren, S. Zhao, L. Bingbing, Object Insertion Based Data Augmentation for Semantic Segmentation, in: International Conference on Robotics and Automation (ICRA), 2022.

[14] P. Vacek, O. Jašek, K. Zimmermann, T. Svoboda, Learning to Predict Lidar Intensities, IEEE Transactions on Intelligent Transportation Systems (T-ITS) (2021).

[15] S. R. Richter, V. Vineet, S. Roth, V. Koltun, Playing for Data: Ground Truth from Computer Games, in: European Conference on Computer Vision (ECCV), 2016.

[16] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: An Open Urban Driving Simulator, in: Conference on Robot Learning (CoRL), 2017.

[17] A. E. Sallab, I. Sobh, M. Zahran, N. Essam, LiDAR Sensor modeling and Data augmentation with GANs for Autonomous driving, arXiv:1905.07290 (2019).

[18] A. E. Sallab, I. Sobh, M. Zahran, M. Shawky, Unsupervised Neural Sensor Models for Synthetic LiDAR Data Augmentation, Advances in Neural Information Processing Systems (NIPS) (2019).

[19] Y. Zhou, O. Tuzel, VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[20] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, M. Tomizuka, Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation, in: European Conference on Computer Vision (ECCV), 2020.

[21] A. Milioto, I. Vizzo, J. Behley, C. Stachniss, Rangenet ++: Fast and accurate lidar semantic segmentation, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2019).

[22] L. Caltagirone, S. Scheidegger, L. Svensson, M. Wahde, Fast LIDAR-based road detection using fully convolutional neural networks, in: IEEE Intelligent Vehicles Symposium (IV), 2017.

[23] P. Chu, S. Cho, S. Fong, K. Cho, Enhanced ground segmentation method for Lidar point clouds in human-centric autonomous robot systems, Human-centric Computing and Information Sciences (HCIS) (2019).

[24] I. Bogoslavskyi, C. Stachniss, Efficient Online Segmentation for Sparse 3D Laser Scans, Photogrammetrie, Fernerkundung, Geoinformation (PFG) (2016).

[25] Z. Shen, H. Liang, L. Lin, Z. Wang, W. Huang, J. Yu, Fast Ground Segmentation for 3D LiDAR Point Cloud Based on Jump-Convolution-Process, Remote Sensing (2021).

[26] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, nuScenes: A multimodal dataset for autonomous driving, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[27] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, J. Hays, Argoverse: 3D Tracking and Forecasting With Rich Maps, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[28] A. Geiger, P. Lenz, R. Urtasun, Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[29] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, J. Gall, SemanticKITTI: A Dataset for Semantic

Scene Understanding of LiDAR Sequences, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[30] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, R. Urtasun, 3D Object Proposals using Stereo Imagery for Accurate Object Class Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence (2017).

[31] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, H. Li, PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[32] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: Advances in Neural Information Processing Systems (NIPS), 2017.