LiDAR-Aug: A General Rendering-based Augmentation Framework for 3D Object Detection

Jin Fang[1,2], Xinxin Zuo[3,4]*, Dingfu Zhou[1,2]*, Shengze Jin[5], Sen Wang[3,4] and Liangjun Zhang[1,2]
[1] Baidu Research [2] National Engineering Laboratory of Deep Learning Technology and Application, China
[3]University of Alberta [4]University of Guelph [5]ETH Zürich, Switzerland
fangjin@baidu.com xzuo@ualberta.ca zhoudingfu@baidu.com

Abstract

Annotating the LiDAR point cloud is crucial for deep learning-based 3D object detection tasks. Due to expensive labeling costs, data augmentation has been taken as a necessary module and plays an important role in training the neural network. "Copy" and "paste" (i.e., GT-Aug) is the most commonly used data augmentation strategy, however, the occlusion between objects has not been taken into consideration. To handle the above limitation, we propose a rendering-based LiDAR augmentation framework (i.e., LiDAR-Aug) to enrich the training data and boost the performance of LiDAR-based 3D object detectors. The proposed LiDAR-Aug is a plug-and-play module that can be easily integrated into different types of 3D object detection frameworks. Compared to the traditional object augmentation methods, LiDAR-Aug is more realistic and effective. Finally, we verify the proposed framework on the public KITTI dataset with different 3D object detectors. The experimental results show the superiority of our method compared to other data augmentation strategies. We plan to make our data and code public to help other researchers reproduce our results.

1. Introduction

Due to its precise range sensing ability to capture the 3D geometric information of environment, LiDAR sensor has been widely used in many applications, especially in Autonomous Driving (AD).

Recently, deep learning-based point cloud analysis such as model classification [27], scene segmentation [28, 42], object detection [35] has achieved significant progress by employing high-capacity neural network. Generally, for training a high-performance neural network, a large amount of annotated training data [14] is a prerequisite, especially in the AD applications where Safety is the first priority.

*Corresponding author

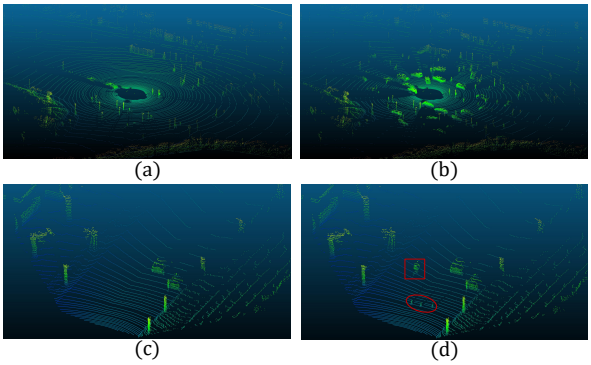

Figure 1: Examples of augmented LiDAR data, where (a), (c) are the original LiDAR point cloud and (b), (d) are the augmented LiDAR point cloud by "LiDAR-Aug". Specifically, the "LiDAR-Aug" can be used to expand the rare objects such as traffic cone and cyclist which has been highlighted in the sub-fig (d).

However, creating large datasets with pixel-level labels is extremely difficult which requires extensive labor work. This problem is even more serious when dealing with the sparse LiDAR data. Due to its sparsity in remote distance, how to identify and annotate the objects with rotated 3D bounding boxes in 3D space becomes particularly difficult.

To reduce the burden of data annotation, generating synthetic data using simulators [7, 34, 25] becomes one of the mainstream solutions. However, the simulator itself is still very expensive, time-consuming, and difficult to generalize to different scenarios. Besides, the distinct domain gap between simulated data and real data is another problem. Although a variety of approaches [16, 41] have been proposed, domain adaption is still an open problem to be investigated.

Another common strategy to deal with limited annotated data is data augmentation. The effectiveness of data augmentation has been verified in many tasks [11, 38], which has become a typical pre-processing process before training the neural network. For 2D tasks, various data augmentation approaches have been proposed such

as [37, 48]. Besides the simple image operations such as flipping, cropping, resizing, etc., domain adaptation and generative models [47, 39] are also introduced to synthesize realistic training images. However, for 3D point cloud, especially LiDAR point cloud, the data augmentation is rarely discussed. Basically, two simple data augmentation strategies have been employed for 3D point cloud related tasks [43, 19, 36, 35, 12]. One is called global augmentation which manipulates the whole LiDAR data globally, such as randomly scaling, flipping and rotating around z-axis. The other is called object augmentation such as the "GT-Aug" (ground truth augmentation) [43, 36] which copies the obstacles or dynamic objects from other frames and pastes them into the current frame. Although the "GT-Aug" strategy is simple, it can indeed improve the detection performance. However, the occlusion between different augmented objects and the occlusion between augmented objects and background points have not been considered. Therefore, it can not be applied to algorithms using spherical projection representation like MV3D [3] and SqueezeSeg [40, 41, 42].

To well address the limitations mentioned above, we propose the first rendering-based LiDAR data augmentation framework "LiDAR-Aug", and apply it to the 3D object detection task. In the proposed framework, we solve the most important question: where and how to insert obstacles into real background frames. Different from the previous method (e.g., "GT-Aug"), we utilize a light-weighted method "ValidMap" to generate the poses for augmented objects while avoiding collision to achieve more reasonable obstacle placements. Finally, we leverage the rendering technique to compose the augmented objects into the real background frames, from which the occlusion constraints are automatically enforced. Therefore, the augmented data generated by our proposed method will be more realistic and diverse. We show sampled data generated from our framework in Fig. 1, from which we can clearly see the data diversity is enormously enriched (Fig. 1 (b)), and the occlusion state is naturally handled. Moreover, "LiDAR-Aug" is more than a data augmentation strategy. We can also utilize it to include and expand the rare objects and corner case scenarios as shown in Fig. 1 (d), which is important to improve the robustness of perception module in AD.

The main contributions of our work can be summarized as below:

1. We present "LiDAR-Aug", a rendering-based data augmentation framework for LiDAR point cloud data, which is more general and effective as compared with the commonly used approach "GT-Aug".
2. We evaluate the proposed framework on the public KITTI dataset, and demonstrate that the "LiDAR-Aug" can significantly improve 3D object detection performance on different baselines.
3. We provide in-depth analyses of the factors in "LiDAR-Aug" processing, including the obstacle placement, augmentation combination, and domain gap.

2. Related Work

Data Simulation: data collection and labeling is a laborious, costly and time-consuming task. To address this problem, synthetic data simulation has emerged as a promising solution from which all ground-truth labels can be easily obtained. For example, [31] generated a large collection of synthetic images of urban scenes called SYNTHIA for semantic segmentation, where the generated images have diverse illumination, textures, pose of dynamic objects, and camera view-points with pixel-wise annotation. Recently, video games have also been used to create large-scale ground truth data for training purposes. For example, in [30, 29, 46], a popular video game called Grand Theft Auto V (GTA-V) is used to generate semantic segmentation ground truth for the synthesized in-game images.

In addition, built upon Unreal engine [1], many driving simulators such as Carla [7] and Airsim [34], gradually spring up from which various kinds of labeled data could be obtained for autonomous driving purposes. Although we have seen improved performance by taking the synthetic data during training [15], the domain gap between synthetic and real data is still a big issue. [32, 33] deal with this domain gap through image translation to map the synthetic images into real ones via GANs.

In contrast to generating purely synthetic datasets, there are image synthesis works [2, 5, 21] that propose to combine real-world imagery and virtual objects of the target category by rendering the virtual objects into the real background. In this way, with only moderate human effort, we can get annotated data with great diversity. [9] shares the same idea in LiDAR simulation by inserting the synthetic CADs into the scanned real background, which is obtained by using the expensive scan device RIEGL. Recently, LiDARsim [24] further replaced the scan background by LiDAR data registration and fusion of multiple frames, and used the reconstructed dynamic objects from point cloud instead of synthetic CADs.

Data Augmentation Data augmentation is a commonly used strategy to avoid overfitting and improve generalization ability of neural networks by artificially enlarging the quantity and diversity of the training samples. There are some general image augmentation strategies such as flipping, rotation and translation. Depending on the specific learning task, various other augmentation approaches have also been proposed. For example, Instaboost [8] presented a copy and paste augmentation method with location probability map for instance segmentation. In addition to aug-

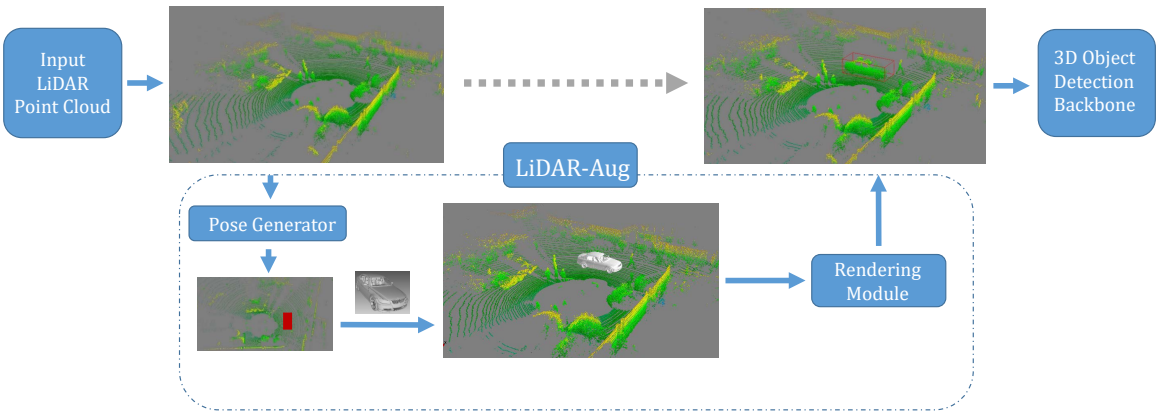

Figure 2: The overview of our proposed "LiDAR-Aug" framework. The input original LiDAR point cloud is first sent to the "Pose Generator" module. Then the CAD models are inserted into the scene under the sampled poses , after which the augmented LiDAR point cloud is generated by rendering the inserted models onto the original background via the "Rendering Module". We then exploit the augmented LiDAR data and validate the effectiveness in the 3D object detection task.

mentation in the image space, [6] brought up the potential to implement augmentation in feature space. Readers can refer to [37] for a comprehensive summary of image augmentation.

While there are extensive works on image augmentation, only a few focus on data augmentation for point cloud. Random scale, flipping, rotation around the gravity axis, and point jittering are the most commonly used augmentation operations for point cloud. [11] investigated those general augmentation operations and evaluated their performance in 3D object detection. Inspired by Mixup [48] in image augmentation, PointMixup [4] proposed an interpolation method that generated new examples through an optimal assignment of the path function between two point clouds. [20] presented an interesting differentiable auto-augmentation framework for point cloud classification, but it could not generalize to detection tasks. For 3D object detection specifically, [43] proposed an augmentation method ("GT-Aug") by sampling ground-truth objects from the database and copying-pasting them into the training frames. But it ignored the placement rationality and occlusion relationship. By simply copying and pasting objects, the foreground and background diversity actually does not increase.

3D Object Detection 3D object detection has been studied for decades with various approaches being proposed. Readers can refer to papers for a thoroughly summary on 3D object detection where the object detection from LiDAR data can be divided into three categories depending on the ways of representing point clouds: projection/view-based [17, 44, 23, 44, 22, 49, 13], voxel-based [51, 43, 18, 45] and raw point cloud-based [50, 19, 26]. Instead of bringing forward new detection methods, in this paper, we evaluate our data augmentation on several representative detection approaches. For example, as a typical voxel-based method, Voxelnet [51] split the LiDAR data into voxels and sent the points in each voxel into a voxel feature encoding layer to get point-wise features. We evaluate on SECOND [43] which has made a series of improvements over VoxelNet [51] to reduce the training and testing time. We also validate our data augmentation with several point cloud based approaches. For instance, PointRCNN [36] directly generated 3D proposals from raw point cloud in a bottom-up scheme followed by a refinement stage combining semantic features. PointPillars [19] is also utilized in this paper which used a pillar representation combining 3D point cloud feature representation and 2d object detection backbone. In addition, we exploit a very recent work called PV-RCNN [35] which combined both point cloud and multi-scale voxel representation and achieves high performance.

By evaluating our data augmentation using various kinds of detection methods, we want to demonstrate the effectiveness and generalization ability of our proposed augmentation approach.

3. Proposed Method

The overview of the proposed framework is illustrated in Fig. 2 which includes two modules namely the *Pose Generator* and the *Rendering Module*. The former is a data distribution-based pose generation engine which provides the location of the object to be inserted and the latter is to render the inserted object into the scene. The detailed introduction of each module is given in the following subsections.

3.1. Pose Generator with ValidMap

As mentioned in previous works [2, 9], how to place the augmented obstacles also affects the perception perfor-

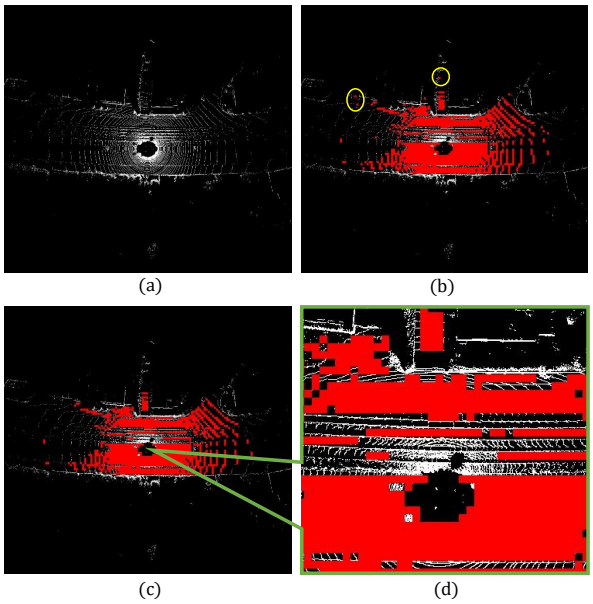

Figure 3: Pose generator with ValidMap. (a) Original input LiDAR point cloud with bird's-eye-view; (b) ValidMap shown in red color pixels are overlaid with (a); (c) Post-processing results for the ValidMap with the isolated pixels marked by yellow circles in (b) are removed; (d) better illustration for (c) by zooming in.

mance. Previous methods (e.g., "GT-Aug") "Copy" the objects from other frames and "Paste" them into the current frame, however, the diversity (e.g., location and orientation) of these objects is not increased with the naive "Copy" from the training data. To ensure the consistency between augmented objects and the background scene, the collision detection between the foreground object and background scene also needs to be considered. It means the occupied LiDAR points in the current frame will be removed. However, since the augmented objects are directly copied from other scenes, it will result in augmented scenes which are not appropriate in real life such as a person inside a bush, a car across a wall or a cyclist inside a building. In addition, the "GT-Aug" uses a plane equation to represent the road to make sure the augmented objects lays on the ground which is only a rough approximation. To well address these problems, we propose a simple yet effective obstacle placement strategy to decide the final pose of obstacles based on a predefined "ValidMap". The generation of the "ValidMap" will be discussed in the following text and the final pose discussed here consists of the position, orientation, and the type of obstacles to be inserted.

ValidMap: the "ValidMap" is defined as a bird's-eye-view map that represents the validness information of obstacle placement, as shown in Fig. 3. To generate a "ValidMap", the LiDAR point cloud is first divided into different pillars [19]. Depending on the points' height distribution in each pillar, they can be classified into three states: *valid*, *invalid*, or *empty*. For each pillar, we set its state according to the following strategy,

$$S^p = \begin{cases} empty, & if \ \ length(Z) = 0; \\ valid, & if \ \ max(Z) - min(Z) < \delta \ \ and \\ & abs(mean(Z) - mean(L(X,Y))) < \gamma; \\ invalid, & others. \end{cases}$$

where L is the road plane equation estimated by plane fitting, Z is an array representing the height of the points, X and Y are the coordinate values for x- and y-axis, δ and γ are hyper-parameters for threshold.

An example of "ValidMap" is shown in Fig. 3, where the isolated valid pixel (having no valid neighbors as shown in sub-figure 3 (c)) will be removed. Based on the "ValidMap", a valid pillar is uniformly sampled, and then the exact position will be randomly selected within this pillar. Here, the height can be directly computed as the mean height of points in the local valid pillar, which is more precise than the global plane estimation strategy used in the "GT-Aug" method.

For the orientation and object's size, we randomly sample them from a prepared ground truth database. The database stored the information of all the annotated obstacles, including position, orientation, and size, from the public dataset.

Collision Avoidance: simply inserting the augmented objects into the background with the sampled position will cause the collision problem. The collision comes from two aspects: 1) overlap with the existing objects in the surroundings of the sampled position, and 2) collision between other inserted objects. To handle this problem, a novel algorithm has been proposed to remove objects that violate collisions. Details of the proposed algorithm are shown in Alg. 1. Specifically, the inserted poses are represented as an array of 3D bounding box and the original LiDAR points inside the non-empty 3D bounding box are removed first. Then the IoUs (Intersection of Union) between every remaining 3D bounding box is computed. The IoUs are represented as a symmetric matrix, in which each value is the IoU between the row item and the column item respectively, and the diagonal values are set to be zero. Instead of simply removing all the non-zero rows and columns, as shown in Alg. 1, we propose to keep the non-overlap 3D bounding boxes as much as possible and remove the boxes with high IoU preferentially.

3.2. Rendering Module

LiDAR captures the 3D information of the surrounding environment by emitting the laser beams and counting the travel time between the target surface to estimate the distance. As inserting the obstacles (CADs) into the current frame, the point cloud of the original background scene may

Algorithm 1 Collision Avoidance Algorithm

Inputs: - LiDAR point cloud $\mathbf{P}$
- Inserted 3D bbox array $\mathbf{O}$
- Threshold τ for points number;

Outputs: - Valid 3D bbox array $\mathbf{O}$;

► Count the points number in every 3D bbox, and remove the non-empty bbox (the bbox whose points number is more than τ is defined as non-empty).
► Count the IoUs between each bbox, the IoUs can be represented as a symmetric matrix $\mathbf{I}$, the values in diagonal are set to zero.
while $\text{sum}(\mathbf{I}) > 0$ **do**
    $i = \text{argmax}(\text{sum}(\mathbf{I}, axis = 0))$
    $delete(\mathbf{O}[i])$
    $\mathbf{I}[i] = 0$
    $\mathbf{I}[:, i] = 0$
end while

be affected. Basically, in order to simulate the LiDAR sensor, for each laser beam, the intersection point to the surface should be updated whenever an obstacle is to be inserted into the scene. Different from LiDAR simulators which need to consider the vertical and azimuth angles distribution for different LiDAR devices, we directly update the range for every laser beam in the current frame.

A straightforward method is to iterate over each laser beam, recompute the intersection points for all the faces of all the CADs, and finally update the intersection point, however, this is very time-consuming. To solve this problem efficiently, we propose a rendering-based method as shown in Fig. 4.

First, the laser beam marked as a red dotted line is supposed to intersect with the object surface where the red dotted line ends, shown in Fig. 4 (a). Affected by the newly inserted CAD, the intersection point should be updated. The inserted CAD is projected onto the current frame to generate an objectness map and depth map. The objectness map decides whether the objects exist and the depth map stores the depth values for the objects. All the laser rays can find the corresponding pixel in the maps with the projection matrix. The rays will not update if the corresponding pixel value in the objectness map is invalid. Otherwise, it will update directly by looking up the depth value from the depth map.

We also add Gaussian noise as the measurement noise of the sensor to simulate the real situation where the laser actually has some perturbation in distance. The noise parameters are estimated from the ground truth database.

3.3. Efficiency Analysis

For data augmentation, the data generation speed should be taken into account. In this paper, the "ValidMap" can be computed with parallel technique efficiently. In addition, the "ValidMap" can also be generated and stored offline. For the rendering and distance updating operation, the updating of all the laser beams can be totally executed on GPUs in parallel.

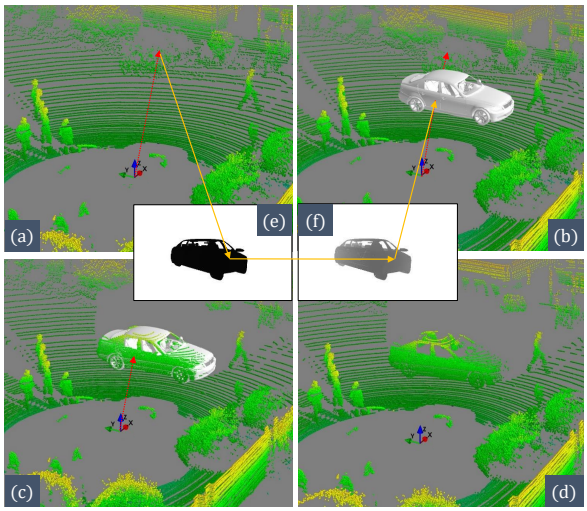

Figure 4: Overview of the LiDAR data rendering procedure. (a) The original LiDAR point cloud, where the red dotted line stands for a laser ray. (b) The CAD is inserted with the pose sampled from the Pose Generator module. The laser ray is occluded by the CAD. (e) shows the objectness map, and (f) is the depth map. The laser ray can find the corresponding pixel from the maps. From the objectness map, we decide for each ray whether to update the range. And we update the distance value for each ray according to the depth map. (c) The laser ray is updated. (d) Final augmented LiDAR point cloud.

4. Experimental Results

In order to verify the effectiveness of the proposed "LiDAR-Aug", we compare it to the "GT-Aug" technique on the public KITTI [10] benchmark for the 3D object detection task. In the following subsections, we first introduce the state-of-the-art 3D object detection baselines that we have used, then the implementation details and the datasets, as well as the evaluation metric, will be described. Finally, the evaluation results on the KITTI benchmark will be given.

4.1. 3D Object Detection Baselines

Four state-of-the-art 3D object detection baselines have been employed here: 1) **SECOND** [43] which greatly improved the detection speed of 3D convolutions based methods by introducing sparse 3D convolution; 2) **PointPillars** [19] which divided the LiDAR point cloud into pillars and transformed all computations on pillars into dense 2D convolutions and significantly improved the detection

speed; 3) **PointRCNN** [36] which proposed a generic point-based proposal method and generated 3D proposals directly from the whole point clouds instead of 2D images; 4) **PV-RCNN** [35] which integrated both the 3D voxel convolution network and point-based set abstraction.

4.2. Implementation Details

We follow the same settings in their original papers to train the baseline methods mentioned above. For a fair comparison, the baseline models that are trained without data augmentation, with "GT-Aug" and with "LiDAR-Aug" share the same hyper-parameters. "Car" and "Pedestrian" classes are used for evaluation here. For each frame, 10 augmented objects will be generated for both "Car" and "Pedestrian'. The pillar resolution for generating the "ValidMap" is $1m \times 1m$. Both δ and γ are set to 0.1m and the range noise variance we used here is 0.5cm. All the models are trained with global augmentations by default, including random frame rotation around gravity axis by an angle from $[-\frac{\pi}{4}, \frac{\pi}{4}]$, random flipping along the heading direction of LiDAR data, and random world scaling with factor from 0.95 to 1.05 uniformly.

4.3. Dataset

KITTI: as one of the most popular benchmarks for 3D object detection in AD, it contains 7481 samples for training and 7518 samples for testing. The objects in each class are divided into three difficulty levels as "easy", "moderate" and "hard", according to the object size, the occlusion ratio, and the truncation level. Since the ground truth annotations of the test samples are not available and the access to the test server is limited, we follow the idea in [3] and split the source training data into training and validation where each set contains 3712 and 3769 samples respectively. Here we only use LiDAR point cloud data in the experiments.

Metrics: we follow the metrics of average precision (AP) on the KITTI benchmark. For "Car" class, we use the metric $\mathbf{AP}_{\text{Car}}\mathbf{70}$ which means that only the samples in which the overlay of bounding box with ground truth is more than 70% are considered as positive for all the detection results. While the threshold for "Pedestrian" class is 0.5 (e.g., $\mathbf{AP}_{\text{Ped}}\mathbf{50}$) due to its smaller size.

4.4. LiDAR-Aug Evaluation

We evaluate our proposed "LiDAR-Aug" with the baseline detectors for 3D object detection on the validation set of KITTI and the AP is calculated by 11 recall positions. To clarify the definition, we use the name "Object-Aug" to represent all the augmentation methods which aim at inserting more objects into the current scene, including "GT-Aug" and "LiDAR-Aug". For each baseline, three types of comparisons are executed here:

1. *No "Object-Aug"*: the baselines are trained without any "Object-Aug";
2. *GT-Aug*: the baselines are trained with "GT-Aug";
3. *LiDAR-Aug*: the baselines are trained with the proposed "LiDAR-Aug".

Methods	$\mathbf{AP}_{\text{Car}}\mathbf{70}$ (%)			$\mathbf{AP}_{\text{Ped}}\mathbf{50}$ (%)		
	Easy	Mod	Hard	Easy	Mod	Hard
SECOND [43] w/o Object-Aug	87.07	76.12	68.44	51.08	46.53	43.89
SECOND [43] w GT-Aug	87.43	76.48	69.10	56.55	52.98	47.73
SECOND [43] w LiDAR-Aug	**88.65**	**76.97**	**70.44**	**58.53**	**54.56**	**51.78**
PointPillars [19] w/o Object-Aug	85.41	73.59	68.76	47.51	43.82	42.20
PointPillars [19] w GT-Aug	87.29	76.99	70.84	57.75	52.29	47.91
PointPillars [19] w LiDAR-Aug	**87.75**	**77.83**	**74.90**	**59.99**	**55.15**	**52.66**
PointRCNN [36] w/o Object-Aug	88.45	77.67	76.30	63.19	54.99	49.18
PointRCNN [36] w GT-Aug	88.88	78.63	77.38	62.16	58.00	50.53
PointRCNN [36] w LiDAR-Aug	**89.56**	**79.51**	**77.89**	**67.46**	**59.06**	**56.23**
PV-RCNN [35] w/o Object-Aug	88.86	78.83	78.30	60.56	53.75	51.90
PV-RCNN [35] w GT-Aug	89.57	83.90	78.91	63.12	54.84	51.78
PV-RCNN [35] w LiDAR-Aug	**90.18**	**84.23**	**78.95**	**65.05**	**58.90**	**55.52**

Table 1: Evaluation results of "LiDAR-Aug" for the "Car" and "Pedestrian" classes on KITTI validation dataset, where "w" and "w/o" stand for "with" and "without". The best values are highlighted in bold and all the values are the higher the better.

The evaluation results are shown in Tab. 1, where we can see significant improvements after using "Object-Aug". It indicates that sampling foreground objects from other frames to the current frame indeed have a positive influence on the detection performance. Compared with the vanilla version without any "Object-Aug", the "LiDAR-Aug" contributes about 1.59%, 2.05%, and 1.65% improvements in average for "easy", "moderate" and "hard" modes. For the class of "Car", the proposed "LiDAR-Aug" performs much better than the "GT-Aug" strategy by 0.74%, 0.64%, and 1.49% for "easy", "moderate" and "hard" modes respectively. For the class of "Pedestrian", "LiDAR-Aug" outperforms the "GT-Aug" averagely by 3.82%, 3.19%, and 6.08% for all the three modes respectively. Therefore, we can conclude that the proposed "LiDAR-Aug" can improve the performance for 3D object detection, and outperforms the previous state-of-the-art method "GT-Aug" in all three modes.

5. Ablation Studies

To further explore the effectiveness as well as the limitation of "LiDAR-Aug", more experiments are conducted with visualizations demonstrated in this section. All the results are evaluated among "Car" class on validation split of KITTI benchmark and the evaluation metric is $AP_{\text{Car}}70$ which is the same as defined in Sec. 4.3.

5.1. Influence of Obstacles Placement

In this section, we investigate the augmentation effectiveness over different obstacle placement strategies. For

"GT-Aug", three types of placements are evaluated here, (i) *random pose*, where the pose is uniformly sampled in the valid range; (ii) *GT Pose*, in which the pose of the inserted object is identical to the one where it comes from; (iii) *ValidMap Pose*, in which the pose is generated with the proposed "ValidMap" algorithm.

For "LiDAR-Aug", four kinds of placement strategies are evaluated here, (i) *random pose*, (ii) *GT Pose*, which is inherited from the experiment in "GT-Aug"; (iii) *ValidMap without collision avoidance* to the background; (iv) *ValidMap* with proposed collision avoidance strategy.

Methods	$AP_{Car}70$ (%)		
	Easy	Mod	Hard
GT-Aug + Random Pose	88.76	78.32	77.35
GT-Aug + GT Pose	88.88	78.63	77.38
GT-Aug + ValidMap Pose	88.48	78.06	77.32
LiDAR-Aug + Random Pose	79.11	65.92	62.41
LiDAR-Aug + GT Pose	87.69	74.23	69.74
LiDAR-Aug + ValidMap Pose w/o collision avoidance	87.14	74.09	69.34
LiDAR-Aug + ValidMap Pose w collision avoidance	**89.56**	**79.51**	**77.89**

Table 2: Evaluation over the influence of obstacles placement for "Car" class on KITTI validation dataset. All the models are trained with PointRCNN baseline.

From the results in Tab. 2, we can find that the "GT-Aug" is not very sensitive to different placement strategies, and the gap among them is small. The reason is that "GT-Aug" directly copies objects from other frames with relatively complete context and the collision can be solved by simply removing the points overlaid with the inserted bounding box.

As for our proposed LiDAR-Aug, the objects are inserted into the background point cloud in an interactive manner. We can see significant improvement in the performance using the ValidMap based placement strategy as compared with the other two strategies. Moreover, as shown in the Tab. 2, collision avoidance is also an indispensable part of our rendering-based LiDAR data augmentation. The AP decreased by 5.46% without enforcing collision avoidance, on average.

5.2. Influence of Augmentation

In this section, we explore the impact of global augmentation (Global-Aug) on detection. Global augmentation refers to the operations applied onto the overall point cloud, including random rotation and flipping. The detailed settings for global augmentation are described in Sec. 4.2. From Tab. 3, we can see that the global augmentation is also important in model training. Without global augmentation, The AP drops about 2.46%, 1.91% and 4.21% in average for "easy", "moderate" and "hard" modes, respectively. And "LiDAR-Aug" alone contributes about 2.42%, 2.17% and 7.82% increase in AP for those three modes respectively. From the evaluation results, we can see that combining object augmentation together with global augmentation can eventually achieve the best performance.

Methods	$AP_{Car}70$ (%)		
	Easy	Mod	Hard
w/o Global-Aug, w/o LiDAR-Aug	85.34	75.59	68.98
w Global-Aug, w/o LiDAR-Aug	88.45	77.67	76.30
w/o Global-Aug, w LiDAR-Aug	87.76	77.76	76.80
w Global-Aug, w LiDAR-Aug	**89.56**	**79.51**	**77.89**

Table 3: Evaluation on the two kinds of augmentation for "Car" class on KITTI validation dataset. All the models are trained with PointRCNN baseline.

5.3. Domain Gap

Referring to [9], CAD-based rendering simulation data still has domain gap with real data. Even when the background comes from scanned point cloud with high precision devices, the domain gap still exists. Different from LiDAR simulation, the background of the augmented LiDAR data generated by "LiDAR-Aug" has no domain difference with original LiDAR data since it is based on the real LiDAR. Instead the domain gap may come from two ways, the quality of augmented objects as well as the simulated poses.

The real annotated objects and the corresponding poses are named as "real object" and "real pose" respectively. The synthetic objects generated with the rendering module are named as "sim object", and the poses sampling from ValidMap are called "sim pose". To explore the domain gap between the augmented data and original data, four extra experiments are conducted here. (i) real object + real pose, which is a baseline without object augmentation. (ii) real object + sim pose, in which the real objects are moved to the positions with synthetic pose sampled from ValidMap. (iii) sim object + real pose, in which the real annotated objects are removed, and the synthetic objects are inserted to the same position using the corresponding pose. Instead of simply cropping the object, we move the points in the real object to the inserted object to make sure that there is no discarding of laser rays. (iv) sim object + sim pose, in which the real annotated objects are removed, and the synthetic objects are inserted with the pose generated via ValidMap. LiDAR simulator is used as a baseline here.

The evaluation results are shown in Tab. 4 where we can find that the domain gap exists between the real object and synthetic object. But the performance outperforms the LiDAR simulator by a remarkable margin, benefiting from the real LiDAR point cloud foreground and laser beams distribution. When trained by combining with original objects, the performance achieves the best performance. Leveraging with domain adaptation technique, the results can be

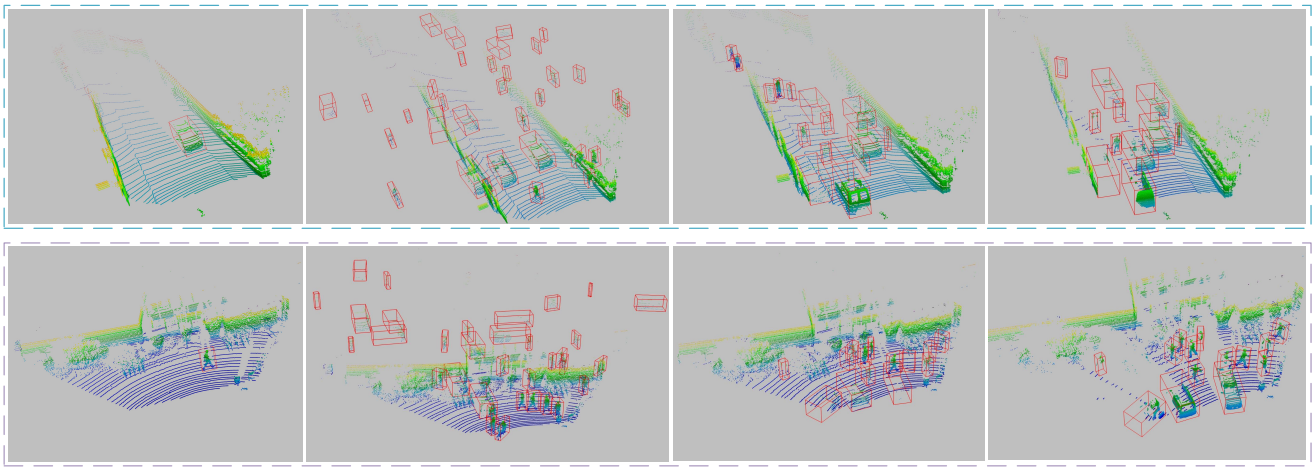

Figure 5: Visualization comparison between "GT-Aug" and "LiDAR-Aug". (a) shows the original data with annotation from KITTI; (b) shows the augmented LiDAR data with "GT-Aug"; (c) shows the augmented LiDAR data with "GT-Aug" but the poses are sampled with ValidMap; (d) shows the augmented LiDAR data with "LiDAR-Aug" while the poses keep the same as in (c).

further improved to some extent, but this is out of our scope. What's more, the performance of using simulated pose drops only 0.27% averagely in comparison with real pose, which verified the effectiveness of our ValidMap.

Methods	$AP_{Car}70$ (%)		
	Easy	Mod	Hard
Augmented LiDAR Simulator [9]	48.88	44.71	40.61
real object + real pose	88.45	77.67	76.30
real object + sim pose	88.05	77.24	76.31
sim object + real pose	76.73	69.38	64.97
sim object + sim pose	75.98	68.57	64.87
LiDAR-Aug	**89.56**	**79.51**	**77.89**

Table 4: Evaluation of the domain gap between original LiDAR point cloud and augmented LiDAR point cloud for "Car" class on KITTI validation dataset. All the models are trained with PointRCNN baseline, except Augmented LiDAR Simulator, which is trained on SECOND and the results refer to [9].

5.4. Results Visualization

In this section, we show some visualizations of "GT-Aug" and "LiDAR-Aug". In Fig. 5, the two rows are from two different frames. (a) is the original data without any augmentation. (b) is augmented LiDAR point cloud with "GT-Aug", where we can find that the placement is quite messy. Some objects are out of the road, or even stand alone in an empty place with no neighboring points. Moreover, some augmented objects have overlap with the background such as the wall and barriers, which is unrealistic. (c) is generated by "GT-Aug" with validMap pose generator. Obviously, the placement is more reasonable and most of the objects are on the ground. (d) is the augmented results with proposed "LiDAR-Aug", with the same ValidMap poses as in (c).

The augmented LiDAR point cloud from "LiDAR-Aug" is visually natural and realistic, since the occlusion state is automatically enforced during the rendering procedure. The effectiveness of "LiDAR-Aug" benefits from the diversity of augmented data. The diversity comes from two ways: firstly, different from simple copy-paste operations, our augmented foreground from CAD models has more diversity in the distribution of object types; Secondly, since the occlusion inference affects both the original background and foreground, some laser beams will be cut off by the inserted objects, which also increases the diversity. It is worth mentioning that some objects are filtered due to the occlusion, but this can be compensated by increasing the number of sampled objects.

6. Conclusion and Future Works

In this paper, we propose a simple but effective framework for LiDAR data augmentation, LiDAR-Aug. The proposed framework includes two modules, a pose generator with ValidMap and a rendering module for combining the real LiDAR point cloud background with synthetic foreground. The whole framework is self-consistent without any redundancy. Moreover, LiDAR-Aug is a light-weighted framework without any learning procedure. The ValidMap can be computed offline in advance and the update of ray distance can be processed in parallel. We evaluate the effectiveness of the framework and conduct deep analysis using the public KITTI dataset. Currently, the domain gap still exists between the augmented frame and the real frame. Combining with domain adaptation technology to further reduce the gap will be one of the future directions.

References

[1] Unreal engine. https://www.unrealengine.com/. 2

[2] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision*, 126(9):961–972, 2018. 2, 3

[3] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017. 2, 6

[4] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees GM Snoek. Pointmixup: Augmentation for point clouds. In *ECCV*, 2020. 3

[5] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *ACM SIGGRAPH 2008 classes*, pages 1–10. 2008. 2

[6] Terrance DeVries and Graham W. Taylor. Dataset augmentation in feature space. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. 3

[7] 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. 1, 2

[8] Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 682–691, 2019. 2

[9] Jin Fang, Dingfu Zhou, Feilong Yan, Tongtong Zhao, Feihu Zhang, Yu Ma, Liang Wang, and Ruigang Yang. Augmented lidar simulator for autonomous driving. *IEEE Robotics and Automation Letters*, 5(2):1931–1938, 2020. 2, 3, 7, 8

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. 5

[11] Martin Hahner, Dengxin Dai, Alexander Liniger, and Luc Van Gool. Quantifying data augmentation for lidar based 3d object detection. *arXiv preprint arXiv:2004.01643*, 2020. 1, 3

[12] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11873–11882, 2020. 2

[13] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11001–11009, 2020. 3

[14] Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2702–2719, 2019. 1

[15] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 746–753. IEEE, 2017. 2

[16] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4893–4902, 2019. 1

[17] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 3

[18] Hongwu Kuang, Bei Wang, Jianping An, Ming Zhang, and Zehan Zhang. Voxel-fpn: Multi-scale voxel feature aggregation for 3d object detection from lidar point clouds. *Sensors*, 20(3):704, 2020. 3

[19] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 2, 3, 4, 5, 6

[20] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Pointaugment: an auto-augmentation framework for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6378–6387, 2020. 3

[21] Wei Li, CW Pan, Rong Zhang, JP Ren, YX Ma, Jin Fang, FL Yan, QC Geng, XY Huang, HJ Gong, et al. Aads: Augmented autonomous driving simulation using data-driven algorithms. *Science Robotics*, 4(28), 2019. 2

[22] Zhidong Liang, Ming Zhang, Zehan Zhang, Xian Zhao, and Shiliang Pu. Rangercnn: Towards fast and accurate 3d object detection with range image representation. *arXiv preprint arXiv:2009.00206*, 2020. 3

[23] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 3

[24] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11167–11176, 2020. 2

[25] Matthias Mueller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. Ue4sim: A photo-realistic simulator for computer vision applications. 2017. 1

[26] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018. 3

[27] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 1

[28] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 1

[29] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2213–2222, 2017. 2

[30] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016. 2

[31] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016. 2

[32] Ahmad El Sallab, Ibrahim Sobh, Mohamed Zahran, and Nader Essam. Lidar sensor modeling and data augmentation with gans for autonomous driving. In *ICML*, 2019. 2

[33] Ahmad El Sallab, Ibrahim Sobh, Mohamed Zahran, and Mohamed Shawky. Unsupervised neural sensor models for synthetic lidar data augmentation. In *NIPS*, 2019. 2

[34] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, pages 621–635, 2018. 1, 2

[35] 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*, pages 10529–10538, 2020. 1, 2, 3, 6

[36] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 2, 3, 6

[37] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019. 2, 3

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 1

[39] Zhiqiang Tang, Xi Peng, Tingfeng Li, Yizhe Zhu, and Dimitris N Metaxas. Adatransform: Adaptive data transformation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2998–3006, 2019. 2

[40] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018. 2

[41] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4376–4382. IEEE, 2019. 1, 2

[42] 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. 1, 2

[43] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2, 3, 5, 6

[44] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 3

[45] Junbo Yin, Jianbing Shen, Chenye Guan, Dingfu Zhou, and Ruigang Yang. Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11495–11504, 2020. 3

[46] Xiangyu Yue, Bichen Wu, Sanjit A Seshia, Kurt Keutzer, and Alberto L Sangiovanni-Vincentelli. A lidar point cloud generator: from a virtual world to autonomous driving. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 458–464, 2018. 2

[47] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2100–2110, 2019. 2

[48] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 2, 3

[49] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. Iou loss for 2d/3d object detection. In *2019 International Conference on 3D Vision (3DV)*, pages 85–94. IEEE, 2019. 3

[50] Dingfu Zhou, Jin Fang, Xibin Song, Liu Liu, Junbo Yin, Yuchao Dai, Hongdong Li, and Ruigang Yang. Joint 3d instance segmentation and object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1839–1849, 2020. 3

[51] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 3