Object Insertion Based Data Augmentation for Semantic Segmentation

Yuan Ren, Siyan Zhao and Liu Bingbing
Huawei Noah's Ark Lab, Toronto, Canada
{yuan.ren3, siyan.zhao, liu.bingbing}@huawei.com

Abstract— **Neural network used for the LiDAR semantic segmentation task needs the point-wise labeled point clouds for training, which is more expensive than bounding box annotations. Enhancing the diversity of training data through object insertion is an effective method to reduce labeling costs. The existing object insertion methods are mainly divided into two categories. First is "copy" the clusters from a LiDAR frame and "paste" it to other frames or positions. Second is inserting CAD models into the background then using LiDAR simulator to generate laser points of the inserted CAD models. "Copy-paste" method cannot generate realistic scanning lines and shadows, and the CAD models, especially the CAD models of flexible objects, are hard to obtain. We propose an object insertion based data augmentation method which can increase the performance of the semantic segmentation network remarkably. First, an object library is created by using the labeled LiDAR point clouds. Then, these objects are inserted into the LiDAR point clouds dynamically during the training. Finally, the realistic scanning lines and shadows are simulated according to the real LiDAR parameters. The experimental results show that the proposed augmentation method can increase the performance of different semantic segmentation frameworks remarkably.**

I. INTRODUCTION

In recent years, deep learning models have made remarkable progress in many computer vision tasks. Particularly, there has been a rising interest for deep neural networks in 3D point cloud processing, including semantic segmentation[1], [2], [3], object detection[4], [5] and tracking[6].

The excellent performance is, however, reliant on the accessibility and diversity of the data, which prevents the models from overfitting and improves the generalization. It is generally expected that performance of the the computer vision algorithms grows as training data expands [7], [8] Unfortunately, there are many tasks that lack the adequate amount of annotated data to train a well-generalized model, such as medical images analysis and driving scene segmentation. The lack of diversity in the ground truth data and imbalanced class distribution will result in poor generalization of models. It would be problematic if a model performs worse when evaluated in unseen data during training due to overfitting, especially for safety-critical tasks.

Given the diverse road conditions in real traffic, generalization is very crucial in order to build a robust perception model as part of the self-driving system, where models should learn the invariances in the driving data. However, data gathering is generally considered a time-consuming process that requires significant human effort [9]. The 3D semantic segmentation task in the LiDAR domain requires annotation in the point-level, which is more complicated than 2D data and requires professional skills, the reason being: 1) Point clouds density varies as its distance to the LiDAR changes. 2) Due to the sparse and unordered nature of LiDAR point clouds, assigning 3D bounding box is more time-consuming than 2D annotation.

Data augmentation is an effective strategy that is used to deal with limited annotated data in both image and LiDAR domain. The most commonly used data augmentation method for LiDAR is to translate, rotate and flip the whole point cloud randomly. A more advanced technology is to insert foreground objects into the point cloud, where the foreground objects are usually from other LiDAR frames [2] or CAD models [10]. Both of these two sources have their own limitations. Foreground objects from other frames cannot increase the diversity of objects. It only inserts the existing objects into the different environments. Moreover, the LiDAR could only scan one side of an object. The density of the points on the object and the direction of scanning lines are also highly related to the position of the object in LiDAR frame. In order to avoid changing those characteristics, the existing methods usually do not change the original position and orientation of the object during insertion, which limits the diversity of the distribution of objects. The realistic CAD models, especially the CAD models for deformable objects like person and cyclist, are difficult to obtain. Unfortunately, those classes happen to be exactly the ones of which we need more objects.

In this paper, we present a model-agnostic object-based data augmentation pipeline for the task of LiDAR-based semantic segmentation in driving scenario. First, an object library is built by extracting foreground objects from the labeled LiDAR point clouds and upsampling them. The weakly labeled data, such as the bounding box, can also be used to build the object library. Then, the objects are inserted into the background dynamically during the training. Thanks to our insertion pose generation algorithm, the occurrence probability of insertion and location distribution of the inserted objects can be controlled by the input parameters. For example, we can define the occurrence probability of the "pedestrian" insertion is 5%, which means "pedestrian" will be inserted into 5% of training LiDAR frames. And 90% of them will be inserted on the sidewalk, the other 10% will be inserted on the road. Finally, the realistic scanning lines and shadows of the inserted objects are generated.

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

1) We present a model-agnostic object-based data augmentation pipeline for point cloud semantic segmentation task, which can effectively increase the diversity of

objects by using point-wise labeled data or weakly labeled data.

2) We test both range-based and voxel-based semantic segmentation methods with the proposed augmentation method. The results show that the performance of both methods is significantly improved.
3) We analyze the influence of the number, class and spatial distribution of inserted objects on model training in detail.

II. Related Work

A. Data augmentation in 2D domain

Data augmentation is a commonly used strategy that enriches the datasets by leveraging the existing data to prevent overfitting and improve generalization of models.

Many data augmentation strategies have been proposed in the image domain and these can be classified into two main categories[11]: basic image manipulation and deep learning approaches. Random flipping, cropping, rotation and translation are among the frequently used basic image manipulation methods. Simple copy and paste strategy has been recently shown to be an effective approach in the image domain[12], [13]. With the development of deep learning, synthetic dataset creation[14], [15] with generative models such as GAN[16], [17] have showed promising results.

B. Data augmentation in 3D domain

The problem that existed in the current driving lidar dataset is the lack of annotated objects for rare classes, which limits the performance of segmentation models in these categories. For example, the nuScenes dataset has very small amount of annotated points that belong to categories such as ambulance car, child pedestrian and animals[18]. The models trained on this fixed set of long-tailed objects may have difficulty generalizing to new ones. However, collecting more data of the less-represented classes would be labor-intensive due to its rare appearance as compared to other classes. Attempts have been made to mitigate this problem by simulating the existing objects into different environments to artificially increase the quantity of objects.

Current widely adopted data augmentation strategy in 3D point cloud datasets can be categorised into two methods: one is random flipping along the X-axis, global scaling with a random scaling factor, global rotation around the Z-axis with a random angle[3], [19]. Another method is the cut-and-paste strategy where the ground-truth data from other scenes are randomly "paste" to the current training scenes, for simulating objects in various environments[20], [21], [2], [10]. Some works have used both of these two approaches for data augmentation[5], [22]. In this work, we propose an object-level data augmentation pipeline for 3D point clouds and evaluate it in both voxel-based and image-based 3D point cloud semantic segmentation algorithms.

C. 3D semantic segmentation

In this work, we evaluate our proposed data augmentation pipeline in the task of 3D semantic segmentation.

With the rising demand for the application of 3D semantic segmentation, many deep learning based methods are proposed, with stat-of-the-art results on several datasets. These methods employ different representation of the point clouds, which can be broadly divided into 3 categories:

1) Point-based methods. In Kernel Point Convolution (KPConv)[23], a kind of point convolution is introduced, which operates on point clouds without any intermediate representation. Its capacity to use any number of kernel points gives KPConv more flexibility than fixed grid convolutions. The main disadvantage of this method is that the preprocessing of data is very time consuming.

2) Range-based methods. These approaches project the unordered point clouds onto a range image, allowing for the use of the powerful structure developed for 2D semantic segmentation while requiring much less computation than 3D. Based on SqueezeNet[24] and conditional random field(CRF), SqueezeSeg[25] is an end-to-end method that project the point clouds onto a spherical surface and use the generated range image as model input. These methods[26], [27], [19], [25], [28] generally focused on both accuracy and real-time performance, which is essential for autonomous driving.

3) Voxel-based methods. Due to the imbalanced density distribution of LiDAR data, these methods regroup the point clouds to structured voxels and apply 3D segmentation algorithms to predict label for each voxel. Cylinder3D[29] applies 3D cylindrical partition to the raw point clouds and apply 3D convolution to the voxels with asymmetric residual blocks.

III. Proposed method

The overview of the proposed method is illustrated in Fig. 1, which has two main components. The first is object library generation, which extracts foreground objects from the labeled LiDAR point clouds, then upsamples them and stores their original pose information. The second is object insertion. Based on the given insertion probabilities, the algorithm selects the appropriate foreground objects in the object library and inserts them on different types of ground and generates the corresponding scanning lines and shadows.

A. Object library generation

For the LiDAR data set with instance label, such as SemanticKITTI and KITTI360, the foreground objects can be easily extracted from the point clouds with the instance labels. While, for these LiDAR data sets (nuScenes) which only have semantic label the foreground objects need to be extracted through two steps. First, all laser points belonging to the same class of object are extracted by using the semantic labels, then the clusters of objects are obtained with clustering algorithm. In addition, the foreground objects also can be extracted by using their bounding box.

Due to the inhomogeneity of the original LiDAR point clouds, the density of these object clusters extracted directly from the original LiDAR point cloud varies greatly. Moreover, the density of the object clusters is highly related with the type of LiDAR used in the data acquisition. In order to enhance

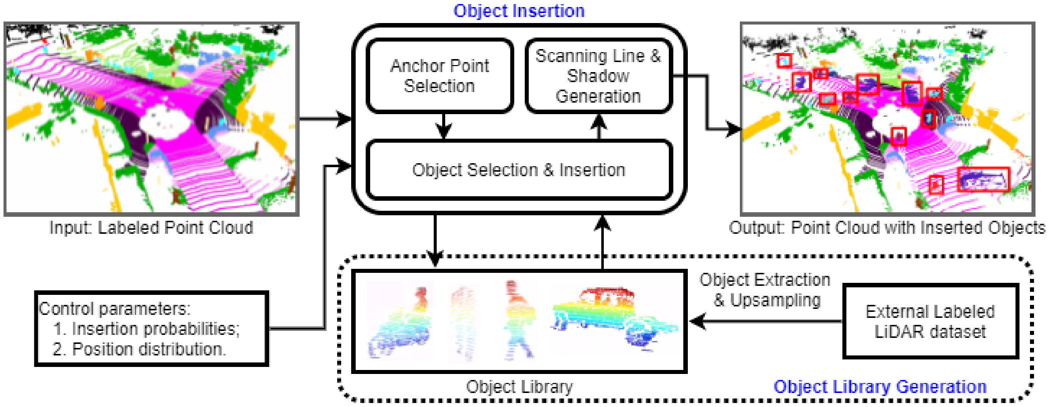

Object Insertion
Anchor Point Selection
Scanning Line & Shadow Generation
Object Selection & Insertion
Input: Labeled Point Cloud
Output: Point Cloud with Inserted Objects
Control parameters:
1. Insertion probabilities;
2. Position distribution.
Object Extraction & Upsampling
External Labeled LiDAR dataset
Object Library
Object Library Generation

Fig. 1: The overview of object insertion based data augmentation procedure

the consistency of the objects in the library, the original object clusters need to be upsampled. The upsampling algorithm is illustrated in Fig. 2. First, the density of the points on each scanning line is increased by linear interpolation. Then, a thin and tall sliding window is used to scan the cluster in horizontal direction. If the points on two adjacent scanning lines appear in the sliding window at the same time, linear interpolation is used to increase points between these two scanning lines. Note that, the reflectivity of the new points is also determined by linear interpolation.

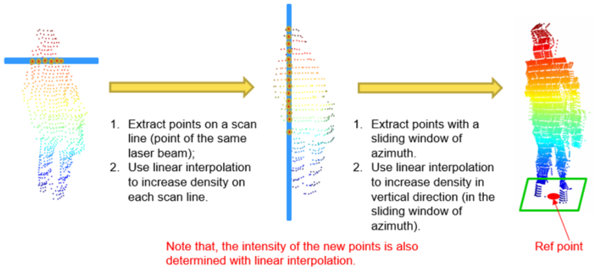

1. Extract points on a scan line (point of the same laser beam);
2. Use linear interpolation to increase density on each scan line.
1. Extract points with a sliding window of azimuth.
2. Use linear interpolation to increase density in vertical direction (in the sliding window of azimuth).
Note that, the intensity of the new points is also determined with linear interpolation.
Ref point

Fig. 2: Cluster upsampling

The upsampled cluster of each object is stored in a separate file, in which the cluster is represented in the original LiDAR coordinates frame. This format implies the orientation information of the object. During the object insertion, this orientation information is used to direct the face with laser points toward the LiDAR. The bottom center point of the cluster is defined as the reference point of the object. Its coordinates and distance to the LiDAR are saved as a properties of the object. With this method, a foreground object library is generated.

B. Object insertion

Object insertion includes five steps, which are anchor point selection, object selection, object insertion, collision test and scanning lines and shadow generation. The object insertion is controlled by two sets of parameters. The first set of parameters controls the number of insertion object and insertion probability, the second set of parameters controls the distribution of the insertion position (anchor point).

1) Anchor point selection: The anchor point will be selected from the ground points, which include road, parking, sidewalk, other-ground and terrain. First, all these ground points are extracted from the LiDAR point cloud. Then, each ground point is assigned a selection probability. The selection probability is computed by

$$p_i = \frac{p^i_{class} \cdot p^i_{pos}}{\sum_{i=1}^{N} p^i_{class} \cdot p^i_{pos}} \qquad (1)$$

where N is the total number of the ground points. Superscript i indicates the ith point. p^i_{class} specifies the occurrence probability of the anchor points on each type of ground, whose possible values are the same as the number of ground point types. The possible values are given by control parameters. For the spinning scan LiDAR, the point cloud is more dense in the area near the LiDAR. If all these ground points have the same probability of being selected as the anchor point, most of the anchor points will be generated near the LiDAR. p_{pos} is used to select anchor uniformly on the ground.

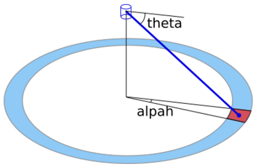


Fig. 3: Ground point density

The red region in Fig. 3 shows the area covered by a laser point, whose area can be computed by

$$S_A = \frac{r^2 \cos\theta}{\sin\theta} \delta\theta\delta\alpha \qquad (2)$$

where r is the distance from the point to the LiDAR. $\delta\alpha$ and $\delta\theta$ are the variations of azimuth and elevation. For spinning scan LiDAR, the $\delta\alpha$ is the same for all the laser points. If the laser beams are uniformly distributed in the pitch direction (like Velodyne LiDARs), $\delta\theta$ is also a constant for all the laser points. In order to select the anchor point uniformly on the ground, p_{pos} should be proportional to its area S_A, which can be written as

$$p_{pos} = \frac{r^2}{\tan\theta} = \frac{(x^2+y^2+z^2)\sqrt{x^2+y^2}}{z} \tag{3}$$

2) Object selection: After the anchor point is determined, suitable object needs to be selected from the object library. First, the distance from anchor point to LiDAR is computed and denoted by $||\mathbf{r}_A||$. Then, the objects in the object library is screened. Only these objects whose original LiDAR-object distance $||\mathbf{r}_R||$ is less than $||\mathbf{r}_A||$ will be added into the potential object list. Finally, one of these object will be selected randomly from the potential object list. This screening strategy ensures that the inserted object is always moved away from the LiDAR relative to its original position, which avoids the increase in point density of the inserted object.

3) Object insertion: When the insertion object is selected, the rotation angle ψ is the rotation angle from object reference point vector $\mathbf{r}_R$ to the anchor point vector $\mathbf{r}_A$, and the translation is $\mathbf{r}_A - \mathbf{r}_R$. By implementing this rotation and translation, the inserted object is moved to the anchor point. Before finalizing the inserted object, the random rotation and scaling of the object need to be done for data augmentation. The random rotation is relative to an axis parallel to z through the anchor point. The rotation angle lies in $[-\frac{\pi}{12}, +\frac{\pi}{12}]$ uniformly, which is small enough and can ensure that the side with laser points facing the LiDAR. The scaling is controlled by two scaling coefficients, k_{xy} and k_z, which are used to control the scale of the inserted object in $x-y$ and z directions, respectively. The value of these two coefficients are generated randomly between 0.95 and 1.05.

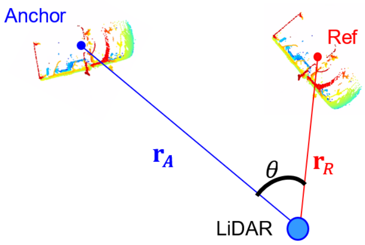


Fig. 4: Object insertion

4) Collision test: Simply inserting the objects into the background will cause the collision problem. The inserted object may overlap with the existing objects in the background or it may overlap with other inserted objects. In order to avoid this, collision test needs to be done after each object insertion. First, the 3D bounding box of the insertion object will be generated. Then, all the points in the bounding box will be extracted from the background point cloud. Finally, two tests will be done to check the occupation in the bounding box. The first test is the semantic label test. All the background points in the bounding box should be ground point. The second is the elevation test. The elevation difference of the background points in the bounding box should be less than 0.2 m. If any of the tests fail, we consider that a collision happens, and the insertion will be cancelled.

The collision test may change the probability of object insertion because some insertions are cancelled. To reduce the impact on the insertion probability, an object insertion operation will be tried 5 times. Experiments show that the probability of successful insertion by this method can reach 96.7%.

5) Scanning lines and shadow generation: The inserted objects are more dense than the surrounding environment, and they do not have scanning lines and shadows. These distortions may negatively affect the training of semantic segmentation models. The realistic scanning lines and shadows are generated by using range image. First, a range image is created based on the resolution of the LiDAR used for background data collection. For example, for a 64-beam LiDAR whose number of scans is 1800 times per revolution. The resolution of the range image should be 1800×64. Then, all the points from the inserted objects are projected into the range image. In each pixel of the range image, only keep the closest point. After this step, the remaining points may not lie on the midpoint of the elevation of the pixel. In 3D space, these points cannot form a smooth scanning line. In order to get the smooth scanning lines, the remaining points need to be adjusted to the midpoint of the corresponding pixels by using Eq. 4.

$$\tilde{x} = \frac{xr\sqrt{1-\sin^2\xi_c}}{\sqrt{x^2+y^2}};\ \tilde{y} = \frac{yr\sqrt{1-\sin^2\xi_c}}{\sqrt{x^2+y^2}};\ \tilde{z} = r\sin\xi_c \tag{4}$$

where $[x,y,z]$ and $[\tilde{x},\tilde{y},\tilde{z}]$ are the coordinates of the point before and after adjusting. ξ_c is the elevation of the laser beam. r is the depth of the point whose value is $\sqrt{x^2+y^2+z^2}$. This adjustment only changes the elevation. The azimuth and depth remain the same. Finally, the background points are projected into the range image. All the pixels occupied by points from inserted objects will be checked. If the point from inserted object is closer, all the background points in this pixel will be removed. Otherwise, inserted point will be removed. With this method, the shadows of objects are generated.

C. Training method

During the training, the objects will be inserted into the original point clouds dynamically. It is important to keep the number of inserted objects within a reasonable range. Inserting too many external objects weakens the effect of the original objects, and it also results in a loss of background points. Oppositely, if the number of inserted objects is insufficient, the performance of the semantic segmentation algorithm cannot be improved. Assuming that the objects in the training, inserting and validation datasets have the same distribution, the optimal ratio of the original/external objects should be one to one. This ratio is achieved by controlling the probability of inserting an object. For example, in training set there are $N_{bicycle}$ bicycles, and the training set includes N_f LiDAR scans. To ensure a one-to-one ratio between the original and inserted objects, the insertion probability should be $N_{bicycle}/N_f$. The insertion probabilities for SemanticKITTI data set are listed in Table I.

Position of insertion is the other factor needs to be controlled because some models may learn from the relative position between the object and background. The realistic insertion position can effectively train these models. In order to do this, we compile statistics of the foreground objects distribution in the training set. The result is shown in Table I. Our insertion will follow these probabilities.

Note that object insertion does not affect the data augmentation which is implemented on the whole point cloud. The whole point cloud will still be randomly rotated, translated and flipped after the object insertion.

IV. EXPERIMENTS

A. Dataset

1) SemanticKITTI: SemanticKITTI contains 22 sequences. Among them, sequences 00 to 10 have point-wise labels. In our experiments, sequence 08 is used as validation set, which includes 4017 LiDAR frames. Other 19130 labeled LiDAR frames are used as the training set. The training set of SemanticKITTI is used as our background point clouds. External objects are inserted into these background point clouds. The unlabeled LiDAR frames in sequences 11 to 21 are not used.

2) KITTI-360: KITTI-360 is used as the external dataset to build the object library. KITTI-360 also has instance labels, so the foreground objects can be extracted without clustering. The semantic classes of KITTI-360 are more detailed and cover the classes of SemanticKITTI. A notable difference is that the rider, bicycle and motorcycle are separated. To ensure consistency of classes, the rider will be merged with bicycle or motorcycle when they are close enough in the bird's eye view (BEV) and labeled as bicyclist or motorcyclist. In order to test the effect of upsampling, two versions of the object library are generated. One is upsampled and the other is not.

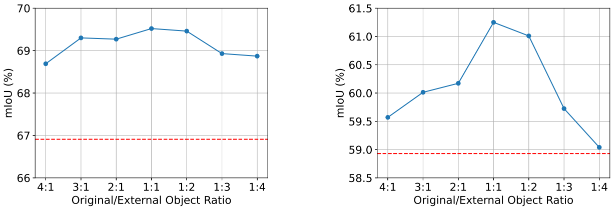
B. Experiment results

In this paper, the Cylinder3D [30], [31] and SalsaDiamond [32] are used as the baseline of voxel-based method and range-based method. For each baseline, the object libraries with and without upsampling are inserted respectively. Some classes have strong contextual relationships. For instance, traffic-signs are usually connected with poles, and trunks always have vegetation on top. Simply inserting one of these objects would destroy the context. To test this, two insertion strategies are designed. The inserted classes of these two strategies are shown in Table I. Strategy 2 includes no-context classes only. When the inserted objects are unsampled, we also try to train the segmentation model without generating the scanning line and shadow. The results are shown in Table II.

For both voxel-based and range-based backbones, the configuration E achieves the best validation mIoU. Comparing with baseline, the mIoU increases 2.61% and 2.32%, respectively. It means that upsampling, inserting objects with less context constraint and generating the realistic scanning lines and shadows are helpful in improving the performance of semantic segmentation. For voxel-based backbone, upsampling can only increase the mIoU by 0.37% (configuration C to E). The reason is that the point cloud will be downsampled during voxelization, so the density of the inserted objects will not significantly affect the voxelized input of the semantic backbone. While, for the range-based backbone, upsampling can increase the mIoU by 0.96%. That is because the background points can pass through the inserted objects that are not dense enough to create noise on the range image, and the upsampling can eliminate this noise. In addition, it can also be found that the range-based backbone is more sensitive to the fidelity of the augmented point clouds. By generating the realistic scanning lines and shadows of the inserted objects the mIoU can be increased by 0.58% (configuration A to B). While, for the voxel-based backbone, the mIoU only increases 0.21%. For both voxel-based and range-based backbones, insertion of these objects with strong contextual relationship should be avoided. Otherwise, the mIoU will loss about 1%.

C. Influence of the number of inserted objects

In the previous experiments, the ratio between the original and external objects is 1:1. Here, different ratios are tried. Other settings are the same as configuration E in Table II. Fig. 5 shows the relationship between the original/external object ratio and validation mIoU.



(a) Cylinder3D (b) SalsaDiamond

Fig. 5: Original/external object ratio

The SemanticKITTI and KITTI-360 datasets are collected in the similar environment, so we can assume that the training set, validation set and object library have very similar distributions. In this case, the mIoU should achieve its maximum value when the original/external objects ratio is close to 1:1. Fig. 5 also confirms this conclusion. In addition, the mIoU of Cylinder3D is less sensitive to the original/external object ratio than that of the SalsaDiamond, which implies that the two methods have different learning abilities.

D. Influence of the insertion position

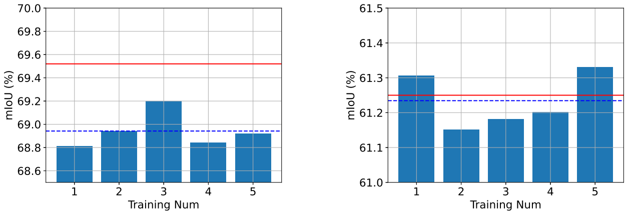
In this test, instead of following the position distribution in the training set, the insertion probabilities on road, sidewalk, parking, other-ground and terrain are generated randomly. The segmentation backbone is trained five times with random insertion probabilities. The validation mIoUs of these five training are shown by these bars in Fig. 6. The blue dashed line is the mean value of these five training. The red solid line is the result with real distribution of the training set.

	Num	Insertion prob	Road	Parking	Sidewalk	Other ground	Terrain	Stg1	Stg2
bicycle	798	4.17	0	0	91.64	0	8.36	✓	✓
car	12335	64.47	48.2	32.89	18.5	0	0		
motorcycle	637	3.33	22.64	5.66	68.55	0	3.14	✓	✓
truck	923	4.82	57.97	14.01	27.63	0	0.39	✓	
bus	44	0.23	0	0	0	1	0		
other-vehicle	3070	16.05	34.92	23.72	26.98	9.39	4.98		
trunk	286	1.49	0	0	6.58	0	93.42	✓	
person	470	2.46	7.68	4.68	84.97	1.0	1.67	✓	✓
bicyclist	–	–	0	0	1	0	0	✓	✓
motorcyclist	–	–	0	0	1	0	0	✓	✓
moving.bicyclist	372	1.94	85.89	0.3	12.9	0	0.91	✓	✓
moving.person	470	2.46	15.89	0.69	76.24	0	7.18	✓	✓
moving.motorcyclist	372	1.94	1	0	0	0	0	✓	✓
moving.bus	67	0.35	1	0	0	0	0		
moving.other-vehicle	78	0.41	91.38	0	8.62	0	0		
moving.truck	12	0.06	1	0	0	0	0	✓	
traffic_sign	594	3.1	0.9	0	74.44	2.24	22.42	✓	
pole	90	0.47	0	0	75.81	0	24.19	✓	

TABLE I: Statistic information of the SemanticKITTI training set and the insertion strategies

Method	Config	Upsamlping	No-context†	Shadow‡	Mean IoU	Car	Bicycle	Motorcycle	Truck	Other-vehicle	Person	Bicyclist	Motorcyclist	Road	Parking	Sidewalk	Other-ground	Building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic-sign
Cylinder3D	baseline				66.91	96.9	54.2	79.7	86.1	67.7	76.8	93.5	0.0	94.6	45.3	81.2	0.0	90.6	59.0	86.6	70.8	70.5	64.5	52.1
	A				68.74	97.0	56.1	83.2	81.4	69.2	81.6	94.0	32.6	94.7	38.3	81.3	0.0	90.2	58.5	88.0	67.4	73.7	65.6	53.3
	B			✓	68.95	96.9	56.6	83.4	84.7	67.1	81.5	92.4	37.0	94.6	38.3	81.4	0.0	90.2	58.4	87.7	67.9	73.0	65.8	53.0
	C		✓	✓	69.15	96.6	56.0	84.5	89.9	67.2	79.6	89.2	40.9	94.7	39.5	81.5	0.0	90.2	58.6	88.0	66.4	73.8	66.1	51.4
	D	✓		✓	68.57	97.0	55.1	82.1	77.4	67.5	80.5	91.3	43.0	94.7	39.3	81.4	0.0	90.1	58.0	87.7	67.8	72.9	65.6	51.6
	E	✓	✓	✓	69.52	96.9	54.6	83.9	89.4	69.2	80.3	90.8	48.4	94.7	38.6	81.5	0.0	90.1	57.9	88.0	66.6	73.9	65.5	50.7
Salsa-Diamond	baseline				58.93	94.0	44.9	35.2	67.9	46.1	69.8	78.2	0.0	94.5	31.7	81.9	1.6	88.5	57.5	85.0	65.5	70.7	61.2	45.4
	A				59.72	94.9	49.6	51.1	65.1	46.4	65.6	75.6	0.0	94.4	47.3	81.3	0.2	85.5	48.8	85.5	62.3	73.2	61.6	46.5
	B			✓	60.30	94.5	46.9	50.4	69.8	53.2	66.0	82.8	0.0	94.2	35.5	82.0	0.5	86.8	49.0	86.2	64.5	74.5	62.4	46.6
	C		✓	✓	60.29	95.1	42.5	45.0	71.1	43.9	63.8	84.4	0.0	94.7	40.6	81.6	0.5	89.2	60.6	86.2	64.7	74.1	62.2	45.2
	D	✓		✓	60.25	94.2	52.9	41.1	82.2	38.3	68.1	82.7	0.0	94.3	36.1	81.3	0.2	87.6	56.2	85.2	65.1	70.6	61.8	46.8
	E	✓	✓	✓	61.25	92.9	46.4	53.0	74.9	55.5	66.1	77.8	0.0	94.7	43.9	81.9	0.2	87.8	51.1	86.4	65.5	74.6	62.5	47.1

TABLE II: Segmentation IoU(%) results on the SemanticKITTI validation dataset. †Only insert objects in no-context classes (strategy 2); ‡Generate scanning lines and shadows or not.



(a) Cylinder3D

(b) SalsaDiamond

Fig. 6: Influence of the insertion position

For the voxel-based backbone, inserting objects according to the actual distribution results in better mIoU (+0.58%). While, the mIoU of range-based backbone is almost unaffected by insertion position. The reason is that in the voxel-based method the foreground object and its adjacent ground can be included in the same receptive field. So the model can learn from the object-ground context. In range-based method, the receptive field usually includes a foreground object and ground points far from that object. So it difficult for the model to learn from object-ground context.

V. CONCLUSIONS

In this work, an object insertion based data augmentation method is proposed which is used to improve the performance of LiDAR-based semantic segmentation. By extracting and upsampling foreground objects from external labeled LiDAR dataset, an object library can be founded. Then, the objects are inserted into the original LiDAR point clouds dynamically during training. Meanwhile, the realistic scan lines and shadows of these inserted objects are generated. The proposed data augmentation methods is suitable for both voxel-based and range-based semantic segmentation backbones. The effectiveness of the method is validated by inserting foreground objects from KITTI-360 to the point clouds of SemanticKITTI dataset.

References

[1] R. Cheng, R. Razani, E. Taghavi, E. Li, and B. Liu, “2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 547–12 556.

[2] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “Pointpillars: Fast encoders for object detection from point clouds,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[3] M. Gerdzhev, R. Razani, E. Taghavi, and B. Liu, “Tornado-net: multiview total variation semantic segmentation with diamond inception module,” *arXiv preprint arXiv:2008.10544*, 2020.

[4] T. Yin, X. Zhou, and P. Krahenbuhl, “Center-based 3d object detection and tracking,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 784–11 793.

[5] S. Shi, X. Wang, and H. Li, “Pointrcnn: 3d object proposal generation and detection from point cloud,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.

[6] H.-k. Chiu, J. Li, R. Ambrus, and J. Bohg, “Probabilistic 3d multi-modal, multi-object tracking for autonomous driving,” *arXiv preprint arXiv:2012.13755*, 2020.

[7] A. Halevy, P. Norvig, and F. Pereira, “The unreasonable effectiveness of data,” *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, 2009.

[8] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, “Revisiting unreasonable effectiveness of data in deep learning era,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[9] N. Paton, “Automating data preparation: Can we? should we? must we?” in *21st International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data*, 2019.

[10] J. Fang, X. Zuo, D. Zhou, S. Jin, S. Wang, and L. Zhang, “Lidar-aug: A general rendering-based augmentation framework for 3d object detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4710–4720.

[11] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[12] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, “Simple copy-paste is a strong data augmentation method for instance segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2918–2928.

[13] S. Illarionova, S. Nesteruk, D. Shadrin, V. Ignatiev, M. Pukalchik, and I. Oseledets, “Object-based augmentation improves quality of remote sensingsemantic segmentation,” *arXiv preprint arXiv:2105.05516*, 2021.

[14] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, “Synthetic data augmentation using gan for improved liver lesion classification,” in *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*. IEEE, 2018, pp. 289–293.

[15] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, “Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification,” *Neurocomputing*, vol. 321, pp. 321–331, 2018.

[16] C. Bowles, L. Chen, R. Guerrero, P. Bentley, R. Gunn, A. Hammers, D. A. Dickie, M. V. Hernández, J. Wardlaw, and D. Rueckert, “Gan augmentation: Augmenting training data using generative adversarial networks,” *arXiv preprint arXiv:1810.10863*, 2018.

[17] S.-W. Huang, C.-T. Lin, S.-P. Chen, Y.-Y. Wu, P.-H. Hsu, and S.-H. Lai, “Auggan: Cross domain adaptation with gan-based data augmentation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 718–731.

[18] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nuscenes: A multimodal dataset for autonomous driving,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[19] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, “Salsanext: fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving,” *arXiv preprint arXiv:2003.03653*, 2020.

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

[21] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, “Augmented reality meets computer vision: Efficient data generation for urban driving scenes,” *International Journal of Computer Vision*, vol. 126, no. 9, pp. 961–972, 2018.

[22] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. 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*, 2020, pp. 10 529–10 538.

[23] H. Thomas, C. R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, “Kpconv: Flexible and deformable convolution for point clouds,” *CoRR*, vol. abs/1904.08889, 2019. [Online]. Available: http://arxiv.org/abs/1904.08889

[24] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.

[25] B. Wu, A. Wan, X. Yue, and K. Keutzer, “Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1887–1893.

[26] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, “Rangenet++: Fast and accurate lidar semantic segmentation,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[27] E. E. Aksoy, S. Baci, and S. Cavdar, “Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving,” in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 926–932.

[28] L. Fan, X. Xiong, F. Wang, N. Wang, and Z. Zhang, “Rangedet: In defense of range view for lidar-based 3d object detection,” *arXiv preprint arXiv:2103.10039*, 2021.

[29] H. Zhou, X. Zhu, X. Song, Y. Ma, Z. Wang, H. Li, and D. Lin, “Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation,” *arXiv preprint arXiv:2008.01550*, 2020.

[30] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, “Cylindrical and asymmetrical 3d convolution networks for lidar segmentation,” *arXiv preprint arXiv:2011.10033*, 2020.

[31] F. Hong, H. Zhou, X. Zhu, H. Li, and Z. Liu, “Lidar-based panoptic segmentation via dynamic shifting network,” *arXiv preprint arXiv:2011.11964*, 2020.

[32] M. Gerdzhev, R. Razani, E. Taghavi, and B. Liu, “Tornado-net: multiview total variation semantic segmentation with diamond inception module,” 2020.