

# NeRF-LiDAR: Generating Realistic LiDAR Point Clouds with Neural Radiance Fields

Junge Zhang<sup>1</sup>Feihu Zhang<sup>2</sup>Shaochen Kuang<sup>1</sup>Li Zhang<sup>1\*</sup><sup>1</sup>Fudan University<sup>2</sup>University of Oxford

## Abstract

*Labelling LiDAR point clouds for training autonomous driving is extremely expensive and difficult. LiDAR simulation aims at generating realistic LiDAR data with labels for training and verifying self-driving algorithms more efficiently. Recently, Neural Radiance Fields (NeRF) have been proposed for novel view synthesis using implicit reconstruction of 3D scenes. Inspired by this, we present NeRF-LiDAR, a novel LiDAR simulation method that leverages real-world information to generate realistic LiDAR point clouds. Different from existing LiDAR simulators, we use real images and point cloud data collected by self-driving cars to learn the 3D scene representation, point cloud generation and label rendering. We verify the effectiveness of our NeRF-LiDAR by training different 3D segmentation models on the generated LiDAR point clouds. It reveals that the trained models are able to achieve similar accuracy when compared with the same model trained on the real LiDAR data. Besides, the generated data is capable of boosting the accuracy through pre-training which helps reduce the requirements of the real labeled data.*

## 1. Introduction

LiDAR sensor plays a crucial role in autonomous driving cars for 3D perception and planning. However, labelling the 3D point clouds for training 3D perception models is extremely expensive and difficult. In view of this, LiDAR simulation that aims at generating realistic LiDAR point clouds for different types of LiDAR sensors becomes increasingly important for autonomous driving cars. It can generate useful LiDAR data with labels for developing and verifying the self-driving system.

Many previous works have studied the LiDAR simulation, which can be mainly categorized into two types: the virtual environment creation method and the reconstruction-based

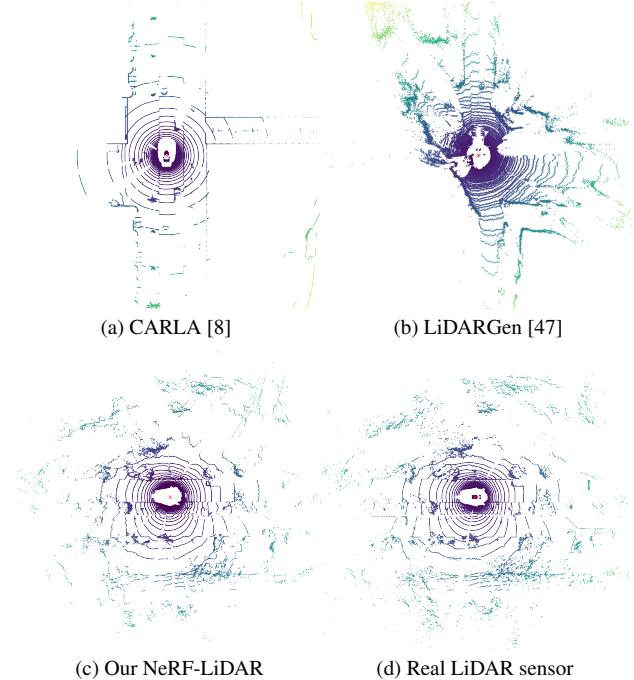


Figure 1. Comparisons of results between our NeRF-LiDAR and other existing LiDAR simulation methods. (a) Method [8] that creates virtual world for LiDAR simulation. (b) Diffusion model used for LiDAR generation [47]. (c) Our NeRF-LiDAR can generate realistic point clouds that is nearly the same as the real LiDAR point clouds (d).

method. The former creates the 3D virtual world by graphics-based 3D modeling and then generates the 3D LiDAR point clouds by physics-based simulation (ray tracing). These kinds of works [8, 16] have natural limitations as it's impossible for 3D modelers to create a virtual world that is the same as the complex real world. The simulated LiDAR points have significant domain differences from the real LiDAR points and cannot be used to train robust deep neural network models. The latter [9, 21] relies on multiple LiDAR scans to densely reconstruct the street background and then place the foreground objects into the background. However, it's expensive to collect dense LiDAR scans which may need special devices [9]. Moreover, it's expensive to generate point-wise semantic labels for simulated LiDAR

\*Li Zhang (lizhangfd@fudan.edu.cn) is the corresponding author with School of Data Science, Fudan University.

data. It still requires human annotations on the 3D scans.

Recently, Neural Radiance Fields (NeRF) [2, 22] have been proposed for implicit reconstruction of the 3D object/scenes with multiple images as inputs. NeRF can render photo-realistic novel views along with dense depth maps.

Inspired by this, we proposed to learn a NeRF representation for real-world scenes and render LiDAR point clouds along with accurate semantic labels. Different from existing reconstruction-based LiDAR-simulation methods [9, 21] or the virtual world creation [8] (Fig. 1a), our method takes full use of the multi-view images to implicitly reconstruct the labels and 3D real-world spaces. The multi-view images can assist the simulation system to learn more accurate 3D geometry and real-world details and generate more accurate point labels. The proposed NeRF-LiDAR model consists of two important modules: 1) the reconstruction module that uses NeRF to reconstruct the real world along with labels; 2) the generation module that learns to generate realistic point clouds through a point-wise alignment and a feature-level alignment.

Since our NeRF-LiDAR can generate realistic LiDAR point clouds along with accurate semantic labels, in the experiments, we verify the effectiveness of our NeRF-LiDAR by training different 3D segmentation models on the generated LiDAR point clouds. The trained 3D segmentation models are shown able to achieve competitive performance when compared with the same model trained on the real LiDAR data which implies that the generated data can be directly used to replace the real labeled LiDAR data. Besides, by using the generated LiDAR data for pre-training and a small number of real data (*e.g.* 1/10) for fine-tuning, the accuracy can be significantly improved by a large margin which is even better than the model trained on a 10 times larger real LiDAR dataset.

## 2. Related Work

LiDAR point-cloud simulation has been studied for many years from the initial game engine based rendering to the state-of-the-art real-world reconstruction-based LiDAR rendering.

### 2.1. Lidar Simulation

The first type of the LiDAR simulation method [8, 13, 16, 43] relies on creating 3D virtual world and rendering the point clouds with physics-based simulation. However, the generated virtual data have large domain gaps with the real data when used for training deep neural networks. This is because the 3d virtual world cannot simulate the complexity and details of the real world. Another point-cloud simulation method [1, 20, 38, 47] generates 3D point clouds based on generative models. However, these generated data cannot be used to train deep neural network models as they also

have significant domain differences with real point clouds. Moreover, it's difficult to generate labels for the point clouds.

State-of-the-art LiDAR simulation methods [9, 21] first reconstruct the real-world driving scenes into 3D meshes and then run the physics-based simulation. In order to achieve dense accurate reconstruction results, these methods need to scan the street many times using expensive LiDAR devices [9]. More importantly, it's still expensive to generate point-wise semantic labels for simulated LiDAR data as it requires human annotations on the reconstructed 3D scenes.

Instead of simulating the whole LiDAR scenes, others, *e.g.* Jin *et al.* [10] use the real-world 3D scenes and propose a rendering-based LiDAR augmentation framework to enrich the training data and boost the performance of LiDAR-based 3D object detection.

Our method also leverages real-world information for learning LiDAR simulation. Our NeRF-LiDAR creates an implicit neural-radiance-field representation of the real world for both point clouds and label rendering.

### 2.2. Neural Radiance Fields

Recently, Neural Radiance Fields (NeRF) [22] have been proposed as an implicit neural representation of the 3D real world for novel view synthesis. NeRFs can take multiple 2D images and their camera-view directions to represent the whole 3D space. However, early NeRFs are only applicable to small object-centric scenes.

Many recent NeRFs have been proposed to address the challenges of large-scale outdoor scenes [2, 3, 32, 37, 44]. There are also some methods [7, 24, 26, 27, 35] leveraging depth supervision to help creat more accurate 3D geometry of scenes. Panoptic or semantic label synthesis for novel views is also explored in [11, 17, 45]. They utilize the density in the NeRF volume to render image labels along with the novel view synthesis.

Inspired by these works, our method reconstructs the accurate 3D geometry using the NeRF methodology in the driving scene and generates 3D point clouds along with accurate semantic labels for the LiDAR simulation.

## 3. NeRF-LiDAR

In this section, we present our NeRF-based LiDAR simulation framework (as shown in Fig. 2). The method consists of three key components: 1) NeRF reconstruction of the driving scenes, 2) realistic LiDAR point clouds generation and 3) point-wise semantic label generation. We formulate the three components into end-to-end deep neural network models for learning LiDAR simulation.

### 3.1. Preliminary

**Neural Radiance Fields** Neural Radiance Fields learn implicit representation for the scenes and render novel view

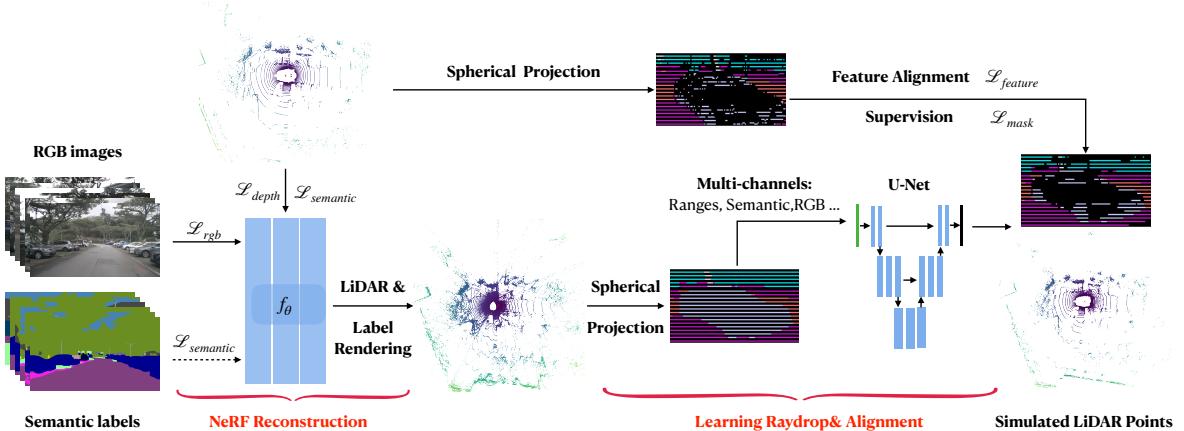


Figure 2. Overview of the NeRF-LiDAR. Image sequences along with the predicted weak semantic labels are used as inputs to reconstruct the implicit NeRF model. LiDAR signals are also used to help create more accurate 3D geometry. Initial coarse point clouds are generated by the NeRF reconstruction through Eq. (5)~(7). The initial point clouds are projected into 2D equirectangular images. We then utilize a U-Net to learn raydrop and the alignment (detailed in Fig. 3) to make the generated point clouds more realistic.

synthesis through volume rendering. It learns a function  $f : (\mathbf{x}, \mathbf{v}) \leftarrow (\mathbf{c}, \sigma)$  for mapping coordinates  $\mathbf{x}$  and viewing directions  $\mathbf{v}$  to color  $\mathbf{c}$  and density  $\sigma$ . The volume rendering is based on discrete rays  $\mathbf{r} = \mathbf{o} + t\mathbf{d}$  in the space and applying numerical integration along the rays to query color:

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - e^{-\sigma_i \delta_i}) \mathbf{c}_i, \quad T_i = \exp \left( - \sum_{j=1}^{i-1} \sigma_j \delta_j \right) \quad (1)$$

where  $\mathbf{o}$  is the origin of the ray,  $\mathbf{d}$  is the direction,  $T_i$  is the accumulated transmittance along the ray, and  $\mathbf{c}_i$  and  $\sigma_i$  are the corresponding color and density at the sampled point  $t_i$ .  $\delta_j = t_{j+1} - t_j$  refers to the distance between the adjacent sampled points.

### 3.2. NeRF Reconstruction

State-of-the-art LiDAR simulation methods [21] rely on dense LiDAR scans for scene reconstruction. To achieve the dense reconstruction of the street, Jin *et al.* use a special (expensive) LiDAR device to collect dense depth maps. [21] scans the street many times to accumulate much denser point clouds. These dense depth maps or point clouds are then used to extract the meshes of the street. Finally, the meshes are used to generate point clouds of different types of LiDAR sensors.

Different from them, we present a new method that can take multi-view images and sparse LiDAR signals to reconstruct the street scenes and represent the 3D scenes as an implicit NeRF model. Recently, more and more driving-scene data are collected by self-driving cars. We propose to use these data to learn the NeRF reconstruction of the driving scenes.

NeRF reconstruction of the unbounded large-scale driving

scenes is challenging. This is because most of NeRFs [22] are designed for small scene reconstruction with object-centric camera views. However, the driving data are often collected in the unbounded outdoor scenes without object-centric camera views settings (*e.g.* nuScenes [5] dataset).

Moreover, since the ego car moves fast during the data collection, the overlaps between adjacent camera views are too small to be effective for building multi-view geometry. We reconstruct the NeRF representation based on the multi-view images collected from the ego vehicle and leverage the LiDAR points to provide extra depth supervision to create more accurate 3D geometries. Besides, the real LiDAR point clouds are used as supervision to learn more realistic simulated LiDAR data.

To reconstruct the NeRF for driving scenes, we use the unbounded NeRF [3] with a modified supervision of:

$$\mathcal{L}_{rgb} = \|\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r})\|_2, \quad (2)$$

$$\mathcal{L}_{depth} = \|\hat{\mathcal{D}}(\mathbf{r}) - \mathcal{D}(\mathbf{r})\|_1. \quad (3)$$

Here,  $\hat{\mathcal{D}}$  is the rendered depth by the volume rendering in Eq. (1):

$$\hat{\mathcal{D}}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - e^{-\sigma_i \delta_i}) z_i, \quad (4)$$

where  $z_i$  is the depth value at the sampled point  $t_i$  on the ray  $\mathbf{r}$ . Since the original unbounded NeRF [3] is extremely slow which takes about one day to train each NeRF block. We adopt voxel-based NeRF [31] to speed up the training and rendering process.

### 3.3. Point-cloud Generation

After learning the implicit NeRF representation of the driving scenes, we set a virtual LIDAR to simulate the

real LiDAR sensor. The virtual LiDAR shares the same parameter settings with the real LiDAR sensors. For example, nuScenes [5] uses Velodyne HDL32E LiDAR sensor, of which the spinning speed is 20Hz and the field of view ranges from -30.67 degree to 10.67 degree (consists of 32 channels). We can therefore simulate NeRF-LiDAR rays with the LiDAR center ( $\mathbf{o}$ ) and direction  $\mathbf{d}$  accordingly:

$$\mathbf{d} = (\cos \theta \cos \phi, \sin \theta \sin \phi, \cos \phi)^T \quad (5)$$

where  $\theta, \phi$  represent the azimuth and vertical angle of the ray, determined by the time interval of the lasers and the settings of the LiDAR sensor.

The origin of the rays  $\mathbf{o}$  changes according to the defined motion of ego cars.

$$\mathbf{o} = \mathbf{o}_0 + \Delta t \cdot \mathbf{v} \quad (6)$$

Here,  $\mathbf{v}$  is the velocity of the ego vehicle and  $\Delta t$  means the time interval from the previous state.  $\Delta t$  is decided by the frame rate of the LiDAR sensors (e.g. 20Hz for nuScenes LiDAR).

Each simulated point  $\mathbf{p} = \{x, y, z\}$  can be then calculated by the pre-defined directions  $\mathbf{d}$  of rays and the distances  $\hat{D}$  from the LiDAR sensor to the real world objects:

$$\mathbf{p} = \mathbf{o} + \hat{D}\mathbf{d} \quad (7)$$

There are about 20~40k points in one frame of a standard 32-channel LiDAR point clouds. To simulate the whole point clouds, for each point, we render a ray to compute the exact 3D location.

### 3.4. Label Generation

To achieve the point-wise semantic labels of the simulated LiDAR point clouds, we use the 2D semantic labels of the images to learn the 3D label generation.

Semantic NeRF [45] proposes to use the semantic logits that could be rendered through volume rendering (Eq. (1)) like RGB color:

$$\hat{\mathbf{S}}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - e^{-\sigma_i \delta_i}) \mathbf{s}_i \quad (8)$$

where  $T_i, \sigma_i, \delta_i$  follows the definition of Eq. 1,  $s_i$  is the semantic logit of the sampled point.

Here, we first consider the most difficult cases (e.g. nuScenes scenes) where there is no any annotated label from the collected driving data (images and LiDAR points). Given unlabeled real images collected from multiple cameras of the self-driving cars, we train a SegFormer [36] model, on the mixture of other datasets including Cityscapes [6], Mapillary [25], BDD [41], IDD [33] to compute weak labels that serve as inputs to the NeRF reconstruction model. To

achieve better cross-dataset generalization of the SegFormer and avoid conflicts in the label definition, we utilize the learning settings and label merging strategy in the [18].

Considering that the generated weak labels may have many outliers, to avoid the influence of the outliers and generate more accurate 3D point labels, we take full use of multi-view geometric and video spacial temporal consistency in our NeRF reconstruction.

In the NeRF training, we combine the image-label supervision into the reconstruction learning by constructing the semantic radiance fields:

$$\hat{\mathcal{L}}_l = CE(\hat{\mathbf{S}}(\mathbf{r}), \mathbf{S}(\mathbf{r})) \quad (9)$$

where  $CE$  is the cross-entropy loss,  $\mathbf{r}$  represents pixel-wise camera rays corresponding to each image pixel,  $\hat{\mathbf{S}}(\mathbf{r})$  is the rendered labels by the NeRF model (Eq.(8)) and  $\mathbf{S}(\mathbf{r})$  is the label predicted by the image segmentation model.

In some other cases, when there is a small number of labeled images or LiDAR frames, we can also leverage the existing ground-truth labels for more robust label generation. For example, in the nuScenes dataset, a small part of the LiDAR frames (about 1/10) was labeled with semantic annotations. We take the sparse 3D point labels along with the weak 2D image labels to learn more accurate semantic radiance fields.

$$\mathcal{L}_l = CE(\hat{\mathbf{S}}(\mathbf{r}_{LiDAR}), \mathbf{S}(\mathbf{r}_{LiDAR})). \quad (10)$$

Here  $\mathbf{r}_{LiDAR}$  represents the point-wise rays emitted by the LiDAR sensor. The total loss for learning our NeRF reconstruction can be represented as:

$$\mathcal{L}_{rec} = \mathcal{L}_{depth} + w_{rgb} \mathcal{L}_{rgb} + \hat{w}_l \hat{\mathcal{L}}_l + \mathcal{L}_l \quad (11)$$

where  $w_{rgb}$  and  $w_l$  balance the RGB geometry reconstruction, the LiDAR rendering and the semantic label rendering.

### 3.5. Learning Raydrop & Alignment

In the real world, the LiDAR sensor cannot receive all beams emitted by itself, influenced by the reflectance ratio of different materials (e.g. glasses), the incidence angle and many other factors [9, 21]. Points are usually dropped when the reflected intensity is below the perception threshold. To make generated LiDAR points closer to the real LiDAR points, we learn a raydrop processing on the generated dense points.

NeRF-LiDAR allows us to render depths (3D points) at arbitrary positions and directions. We use the ground-truth LiDAR frames as supervision to learn the raydrop. Given one ground-truth LiDAR frame  $P$ , we render the simulated LiDAR frame  $\hat{P}$  at the same location accordingly.

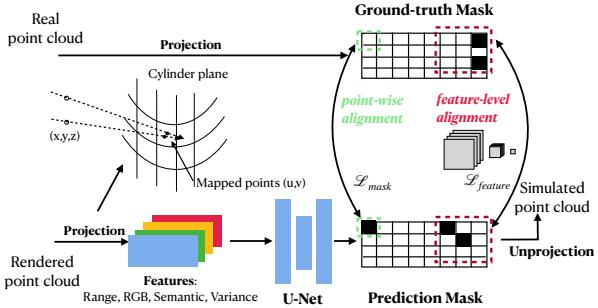


Figure 3. Illustration of learning raydrop and alignment. The initial coarse point clouds are projected into 2D equirectangular images. We use the projected depth, RGB texture, and depth variances as input to a standard U-Net. The U-Net learns the raydrop mask to improve the initial coarse point clouds through the point-wise alignment (Eq. (13)) and the feature-level alignment (Eq. (14)). Finally, the refined equirectangular images are back-projected to 3D space to achieve the expected LiDAR point clouds.

The ground-truth  $P$  and the simulated  $\hat{P}$  should have strong point-wise correspondence.

$$P \simeq \hat{P} \quad (12)$$

We adopt such point-to-point correspondence as the learning target.

**Equirectangular image projection** It's difficult to create a point-to-point correspondence between the two irregular 3D point clouds. To better leverage the point-wise correspondence, similar to [47], we first render all generated points into a 2D equirectangular image (a panorama sparse depth image, as illustrated in Fig. 2). For example, in nuScenes dataset, the resolution of the 32-channel LiDAR equirectangular image is set as  $32 \times 1024$ .

To project the irregular LiDAR points, we adopt the spherical projection [12] to project our points into the equirectangular image grids (as illustrated in Figure 3).

Similarly, the real LiDAR frame is also transferred into a 2D  $32 \times 1024$  equirectangular image. In this way, we can easily create the correspondence in 2D grids.

**Point-to-point Alignment** To learn the drop probabilities for each 2D grid location in the equirectangular image, we employ a standard U-Net which encodes the depth ranges, semantic labels, RGB textures, and depth variances between neighborhoods into a feature representation. The U-Net outputs a 2D probability map (a binary mask) to represent the raydrop results. We take the corresponding real LiDAR equirectangular image as the learning target.

$$\mathcal{L}_{mask} = CE(\hat{M}, M_{gt}) \quad (13)$$

where  $\hat{M}, M_{gt}$  represent predicted and true mask respectively. Extra points/grids are dropped through the

learned drop mask. The expected 3D point clouds can be achieved through back-projection of the equirectangular image.

**Feature-level alignment** The above point-wise alignment aims at making the generated points more realistic or spatially closer to the real point locations. However, the generated LiDAR data will finally be used to train deep neural network models for 3D perceptions. To make the generated data more effective and able to achieve better accuracy when training deep neural networks (e.g. 3D perception models), we propose the feature-level alignment to further regulate the distribution of the generated point clouds.

$$\mathcal{L}_{feat} = \sum_{k=1}^n w_k \|F(\hat{I})_k - F(I_{gt})_k\|_1 \quad (14)$$

Where  $F$  is a pre-trained and fixed feature extractor (e.g. VGG [30], Point-cloud segmentation Net [23]). We use  $n$ -level pyramidal features to compute the feature distance.  $n = 4$  is the number of feature levels,  $w_k = 2^{k-n}$  is the weights for  $k$ th-level features. The loss  $\mathcal{L}_{feat}$  measures the feature-level similarity between the real and the simulated point clouds. To enable the back-propagation from the feature loss to the previous raydrop module, we apply the gumble-softmax proposed in [15] on the ray drop processing.

The whole generation target can be represented as:

$$\mathcal{L}_{gen} = \mathcal{L}_{mask} + w_{feat}\mathcal{L}_{feat} \quad (15)$$

The feature-level distribution alignment can make the generated data more effective and achieve better accuracy in training deep segmentation networks. Besides, we find that it also helps to remove extra outliers and floats in the generated point clouds (examples are shown in Fig. 4 and Tab. 2).

## 4. Optimization

We optimize the NeRF-LiDAR model through the reconstruction loss  $\mathcal{L}_{rec}$  (Eq. (11)) and the generation loss  $\mathcal{L}_{gen}$  (Eq. (15)). We set balance loss weights as  $w_{feat} = 0.2, w_{rgb} = 0.5$  and  $\hat{w}_l = 0.2$ . For a stable training and faster convergence, we first train the reconstruction for 40K iterations without using the generation loss. Then, the reconstruction part is frozen and we further learn the generation module for another 20k iterations. As for the reconstruction part, the batch size for sampling RGB rays is 2048 and the number of the rays for LiDAR supervision is 2048. We adopt Adam optimizer to optimize our network. The learning rate is annealed log-linearly from  $5 \times 10^{-4}$  to  $5 \times 10^{-6}$  with a warm-up phase of 2500 iterations. In the

generation part, we adopts Adam with a learning rate of  $1 \times 10^{-4}$ . The batch size of images is 4(More implementation details are available in appendix Sec. A.4.)

## 5. Experiments Results

### 5.1. Experimental Settings

**Dataset** We use the standard nuScenes self-driving dataset for training and evaluation. NuScenes contains about 1000 scenes collected from different cities. Each scene consists of about 1000 images in six views that cover the 360° field of view (captured by the front, right-front, right-back, back, left-back and left-front cameras). 32-channel LiDAR data are also collected at 20 Hz. Human annotations are given in key LiDAR frames (one frame is labeled in every 10 frames).

We use both the unlabeled images and the LiDAR data for training our NeRF-LiDAR model. The labeled key LiDAR frames are used for evaluations. Limited by computing resources, we take 30 nuScenes sequences from the whole dataset. Each sequence covers a street scene with a length of 100~200m, and the total length is about 4km. Namely, we reconstruct about 4km of driving scenes for training and evaluation.

**Evaluation Settings** To avoid conflicts in the label definitions, we remap the image segmentation labels into five different categories (road, vehicles, terrain, vegetation and man-made) in accordance with the nuScenes LiDAR segmentation labels.

In the training set, we use a total of 7000 unlabeled LiDAR frames and 30000 images for training our NeRF-LiDAR model.

There are extra 1000 labeled LiDAR frames provided in these nuScenes scenes. We mainly use these labeled data for testing and fine-tuning in the experiments.

To evaluate the quality of the generated point clouds and point-wise labels, we train different LiDAR segmentation models (Cylinder3D [46] and RangeNet++ [23]) on the generated data and compare the segmentation model with those models trained on the real nuScenes LiDAR data(25k iterations).

We use two evaluation sets to evaluate the 3D segmentation results. The first *Test Set 1* consists of 400 labeled real point clouds that is extracted from the 30 reconstructed scenes which are not used for training. This validation set is from the same scenes as the simulation data.

The second *Test Set 2* is the whole nuScenes validation set which consists of ~5700 LiDAR point clouds from other nuScenes scenes (not including the 30 selected scenes). This is used to test the quality and generalization/transfer abilities of the simulation data. We test the trained model in other unseen scenes (*results are available in Appendix Sec. A.1*).

Training Set	Test Set 1					
	road	vege.	terrain	vehicle	manmade	mIoU
CARLA	76.4	47.3	X	33.7	54.4	52.9*
Real 1k	96.2	83.6	83.1	83.0	86.4	86.5
Real 10k	97.0	83.6	84.5	89.3	87.8	88.4
Sim20k	93.5	70.4	77.6	79.1	80.7	80.3
Sim20k + real1k	<b>97.1</b>	<b>84.1</b>	<b>85.3</b>	<b>92.2</b>	<b>86.9</b>	<b>89.1</b>

Table 1. Evaluation an comparisons with the real LiDAR data and CARLA. \* mean IoU of four classes.

### 5.2. Ablation study

To demonstrate the effectiveness of our method components, in Table 2, we conduct experiments on different components of our NeRF-LiDAR(The ablation on user defined parameters (*e.g.* balance weights) are presented in Appendix Sec. A.4). We conduct ablation studies on 25 sequences of all data.

**Effects of RayDrop** We compare three different raydrop settings in Table 2 and Fig. 4. Without any raydrop, the generated LiDAR data report a mean IoU of 65.7 after training the 3D segmentation model [46]. By adding a random drop strategy, the mean IoU is dropped to 63.5. And when we use our learning-based raydrop, the mean IoU is improved from 65.7 to 66.3.

**Effects of Feature Loss** We also evaluate the effects of the feature loss in Table 2. We use two different feature extractors (VGG [30] and RangeNet++ [23]) to implement the feature-level alignment. Without feature loss for feature-level alignment, our method reports a mean IoU of 66.3. By using the pre-trained VGG Net [30] for feature alignment, the result is improved to 66.5. By implementing a pre-trained 3D segmentation network as the feature extractor, the results are significantly improved to 69.9.

### 5.3. Label Quality

In Fig. 5, we visualize the generated LiDAR labels and compare them with the ground-truth labels in the real data. We can observe that the generated point clouds by our NeRF-LiDAR have strong point-to-point correspondence with the real data and labels. The label is accurate and close to the manual annotations. In Appendix Sec. A.4, we also evaluate the quality of the labels by computing the mIoU by comparing it with the manual labels. Our NeRF-LiDAR can generate accurate labels with a high mean IoU of 80~95 under different settings.

**Pseudo segmentation of real LiDAR scenes.** Compared with the pseudo segmentation labels on the real data, our NeRF-LiDAR can generate more rarely seen hard cases (in different view angles and routes) which have not been be collected by the dataset. These data significantly improve

	Ray drop			Feature loss		mIoU
	no ray drop	random drop	learning drop	vgg [30] loss	rangenet [23] loss	
Simulation	✓					65.7
		✓				63.5
			✓			66.3
				✓	✓	66.5
					✓	<b>69.9</b>

Table 2. Ablations on different settings of ray drop and feature loss. Models are evaluated on the validation set *Test Set 2*.

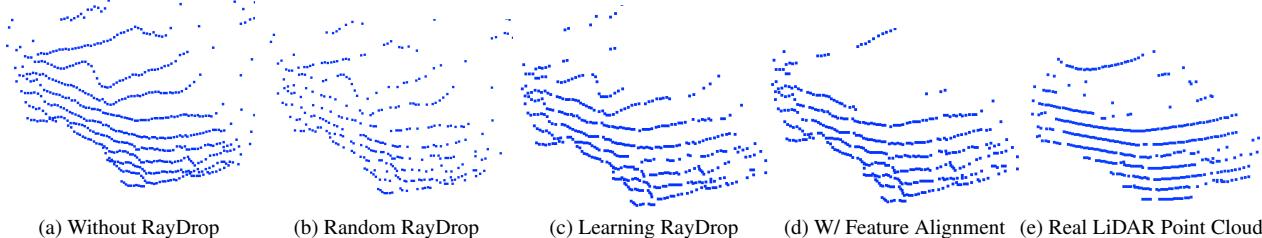


Figure 4. Comparisons of different settings for LiDAR rendering. (a) Point clouds without raydrop, (b) The generated point clouds after random raydrop. (c) Point clouds after our learning based raydrop but without using the feature-level alignment. (d) The final generated point clouds with both learning based raydrop and the feature-level alignment. (e) the real LiDAR point clouds.

	road	vege.	terrain	vehicle	manmade	mIoU
LiDAR + pseudo Seg	70.6	39.4	30.7	20.9	52.0	42.7
NeRF-LiDAR 10k	<b>91.6</b>	<b>61.1</b>	<b>59.2</b>	<b>69.7</b>	<b>68.0</b>	<b>69.9</b>

Table 3. Comparisons between the NeRF-LiDAR data and the unlabeled real LiDAR data that uses pseudo segmentation labels.

the diversity of the training dataset and boost the accuracy in training 3D segmentation models (Table 3).

#### 5.4. Comparisons with CARLA and Real LiDAR

In Table 1, we evaluate the quality of the generated data by using it to train 3D segmentation model [46], the mean IoU is used as the evaluation metric.

In Table 1, we use the predicted weak image segmentation labels and 1000 labeled LiDAR frames to train our NeRF-LiDAR. We use the *Test Set 1* which is extracted from the same scenes as the simulation data to evaluate the accuracy of the 3D segmentation models. CARLA simulator [8] and different real LiDAR sets are taken as baselines. The real 1k and real 10k data contain the labeled frames in the scenes. When we train the point cloud segmentation network on the 20k simulation data, it achieves a mean IoU of 80.3, which is close to real 1k data and far better than the model trained on the 20k CARLA data. If we use 1k real data for fine-tuning, the mIoU can be further improved to 89.1, which exceeds real 10k data.

#### 5.5. Comparisons with Other Reconstruction-based Simulators

Our NeRF-LiDAR possesses apparent advantages over other reconstruction-based LiDAR simulators [9, 21]. First, RGB images are used to assist the reconstruction of the real

world in our NeRF-LiDAR. Images provide useful multi-view geometry information for more accurate reconstruction and label generation. Secondly, no manual annotations on the point clouds are required. We use NeRF representation to learn label generation. Multi-view images provide useful label information and geometry consistency to reduce the outlier labels. Moreover, no mesh reconstruction of the real world is required. NeRF-LiDAR is an implicit representation of the real 3D world. It can preserve more details of the real world by leveraging the RGB texture contents. Finally, NeRF-LiDAR does not require dense point clouds for the reconstruction of the real world. LiDARSim [21] uses a 64-channel LiDAR and scans the street many times to achieve the dense point clouds for LiDAR simulation. Augmented LiDAR simulator [9] utilizes a special and expensive LiDAR device to achieve the dense point clouds. As a comparison, our NeRF-LiDAR relies more on the 2D images to achieve dense 3D geometry information.

We compare our method with LiDARSim [21] in Table 4 and Figure 6. About 1km driving scenes are reconstructed for evaluations. Considering that the official code of LiDARSim is not published, we tried our best to reproduce the procedures, *i.e.* accumulating the LiDAR points, calculating normals, building meshes and doing ray-casting and ray-dropping to generate simulation data. However, the aggregated points and labels on nuScenes dataset are not dense enough to build high-quality meshes and thus produce poor simulated results.

#### 5.6. Combining Real and Simulation Data

We combine real data with NeRF-LiDAR data generated from ground-truth scenes to see if simulated data can further boost performance when used for training. As shown in



Figure 5. Comparisons between the data and label generated by the NeRF-LiDAR and the real LiDAR data with human annotations. For better visualization, we project the 3D point cloud as 2D equirectangular image with colorized labels. Our NeRF-LiDAR (a) is shown able to generate accurate labels and realistic point clouds that is almost the same as the real data (b).

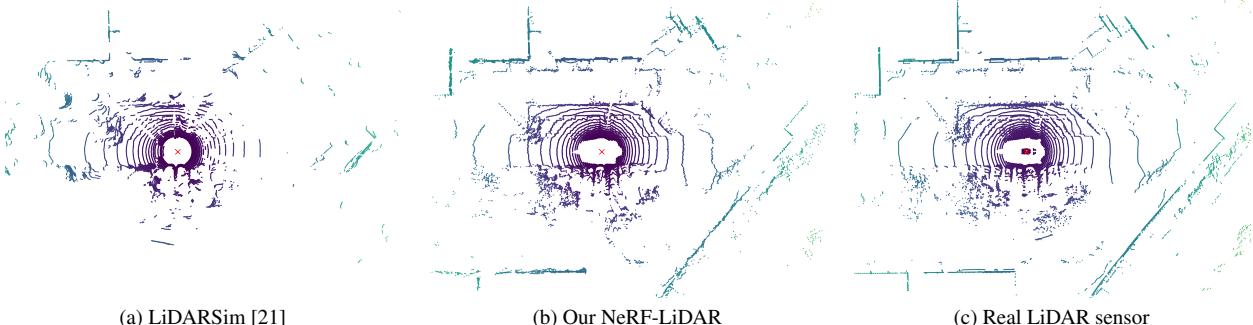


Figure 6. Comparison between our NeRF-LiDAR and LiDARSim [21]. Sparse aggregated point cloud leads to poor-quality mesh when reconstructing the scene. Thus, the simulated LiDAR points lose reality compared to NeRF-LiDAR.

	road	vege.	terrain	vehicle	manmade	mIoU
LiDARSim [21]	83.1	55.1	39.1	36.7	75.2	57.8
NeRF-LiDAR	<b>92.5</b>	<b>69.9</b>	<b>70.1</b>	<b>74.7</b>	<b>84.8</b>	<b>78.4</b>

Table 4. Comparison with LiDARSim. We reconstruct 25 sequences for both LiDARSim and our NeRF-LiDAR and generate 10k frames of LiDAR point clouds to train 3D segmentation models. The mean IoU is evaluated on the correspond part of these scenes in *Test set 1*

Table 1, simulated data along with 10% real data (real 1k) are able to achieve better performance (Sim10k + real 1k: 89.1) than 100% real data (real 10k: 88.4).

## 5.7. Moving objects and object placement

We apply masks on both images and LiDAR points for moving objects during the NeRF reconstruction stage to eliminate the blur caused by the moving objects. We reconstructed the cars according to the [42] and then place them into the simulated scenes. The processing of raydrop is accomplished with the rendered background scenes (More experiments and details are available in Appendix Sec. A.3).

## 5.8. Comparisons with Different NeRF Baselines

There are also many other NeRFs that can be used for scene reconstruction. Most of them (*e.g.* Neus [34], VolSDF [39]) can only be used for object or small scene reconstruction. These methods fail in large-scale outdoor driving scenes. So we mainly make comparisons with the large-scale NeRFs (Urban-NeRF [26] and Panoptic Neural Fields (PNF) [17]) (More detailed comparisons are available in Appendix Sec. B).

	Training	Rendering (per frame)
Urban-NeRF [26]	24 hours	6s
PNF [17]	~ 24 hours	~ 6s
NeRF-LiDAR	1 hour	0.3s

Table 5. Comparison of efficiency with other NeRFs

As shown in Table 5, our NeRF model is 20× faster than Urban-NeRF and PNF. Using Urban-NeRF or PNF to reconstruct 4km driving scenes would take about 60 days. That is a big limitation in real-world applications. Our NeRF-LiDAR also produces more accurate depth rendering than PNF since PNF does not use point clouds in supervision.

## 5.9. Efficiency

We train a total of 100 NeRF-LiDAR blocks to represent the 4km 3D driving scenes. Each NeRF-LiDAR model is trained on 1 NVIDIA A40 GPU which takes about 1 hour. During rendering point clouds and labels, each LiDAR frame costs less than 1s.

## 6. Conclusion

NeRF-LiDAR is proposed to generate realistic LIDAR point clouds via learning neural implicit representation. Images are used in NeRF-LiDAR to compute accurate 3D geometry and labels rendering. The real LiDAR data is also utilized as supervision to learn more realistic point clouds. The effectiveness of our NeRF-LiDAR is verified by training 3D segmentation neural networks. It's shown that the 3D segmentation models trained on the generated LiDAR data

can achieve similar mIoU as those models trained on the real LiDAR data.

## References

- [1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018.
- [2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.
- [3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.
- [4] Lucas Caccia, Herke Van Hoof, Aaron Courville, and Joelle Pineau. Deep generative modeling of lidar data. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5034–5040. IEEE, 2019.
- [5] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint*, 2019.
- [6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.
- [7] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *CVPR*, 2022.
- [8] 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.
- [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.
- [10] Jin Fang, Xinxin Zuo, Dingfu Zhou, Shengze Jin, Sen Wang, and Liangjun Zhang. Lidar-aug: A general rendering-based augmentation framework for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4710–4720, 2021.
- [11] Xiao Fu, Shangzhan Zhang, Tianrun Chen, Yichong Lu, Lanyun Zhu, Xiaowei Zhou, Andreas Geiger, and Yiyi Liao. Panoptic nerf: 3d-to-2d label transfer for panoptic urban scene segmentation. *arXiv preprint arXiv:2203.15224*, 2022.
- [12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.
- [13] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Blensor: Blender sensor simulation toolbox. In *International Symposium on Visual Computing*, pages 199–208. Springer, 2011.
- [14] Benoît Guillard, Sai Vemprala, Jayesh K Gupta, Ondrej Miksik, Vibhav Vineet, Pascal Fua, and Ashish Kapoor. Learning to simulate realistic lidars. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)*, number CONF, 2022.
- [15] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.
- [16] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.
- [17] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12871–12881, 2022.
- [18] John Lambert, Zhuang Liu, Ozan Sener, James Hays, and Vladlen Koltun. Mseg: A composite dataset for multi-domain semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2879–2888, 2020.
- [19] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018.
- [20] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2021.
- [21] 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.
- [22] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
- [23] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4213–4220. IEEE, 2019.
- [24] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In *Computer Graphics Forum*, 2021.
- [25] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kortschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999, 2017.

- [26] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *CVPR*, 2022.
- [27] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, 2022.
- [28] Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492, 2021.
- [29] 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. Springer, 2018.
- [30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.
- [31] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Improved direct voxel grid optimization for radiance fields reconstruction. *arXiv preprint arXiv:2206.05085*, 2022.
- [32] Matthew Tancik, Vincent Casser, Xincheng Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022.
- [33] Girish Varma, Anbumani Subramanian, Anoop Namboodiri, Mammoohan Chandraker, and CV Jawahar. Idd: A dataset for exploring problems of autonomous navigation in unconstrained environments. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1743–1751. IEEE, 2019.
- [34] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.
- [35] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. *arXiv preprint*, 2021.
- [36] Enze Xie, Wenhui Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34:12077–12090, 2021.
- [37] Ziyang Xie, Junge Zhang, Wenye Li, Feihu Zhang, and Li Zhang. S-nerf: Neural radiance fields for street views. In *The Eleventh International Conference on Learning Representations*, 2023.
- [38] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019.
- [39] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021.
- [40] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *CVPR*, 2021.
- [41] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [42] Guo Yuanchen. Neural surface reconstruction based on instant-ngp, 2022.
- [43] 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, ICMR ’18*, page 458–464, New York, NY, USA, 2018. Association for Computing Machinery.
- [44] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.
- [45] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15838–15847, 2021.
- [46] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. *arXiv preprint arXiv:2008.01550*, 2020.
- [47] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. *arXiv preprint arXiv:2209.03954*, 2022.

# APPENDIX

## A. More Experimental Results

### A.1. LiDAR segmentation

In our paper, we evaluate the quality of the generated LiDAR point clouds by training two popular 3D segmentation algorithms Cylinder3D [46] and RangeNet++ [23]. The evaluation results of Cylinder3D are reported in the paper (Table 1) and Table 7. In Table 6, we report the evaluation results of the RangeNet++ on different training sets.

We can observe that the LiDAR point clouds generated by our NeRF-LiDAR are effective in training both the Cylinder3D and the RangeNet++. Especially, by using 20k simulation data for pre-training and another 1k real data for fine-tuning, the RangeNet++ achieves an improvement of 10% in mean IoU when compared with the model trained on 1k real data and the model can achieve a better accuracy than 10k real data training. In Table 7, fine-tuning the pretrained model with 1k real data will boost the accuracy of foreground objects from 80.1 to 86.2. This demonstrates the generalization of our NeRF-LiDAR and implies that our simulation LiDAR data can be used in many existing 3D segmentation networks and boost their training to achieve better accuracy.

Test Set 1						
Training Set	road	vege.	terrain	vehicle	manmade	mIoU
CARLA	70.5	31.3	X	9.7	29.3	35.2 *
Real 1k	89.0	35.9	53.9	45.4	52.5	55.3
Real 10k	89.9	40.6	59.0	<b>55.3</b>	56.2	60.2
Sim20k	87.9	42.4	53.9	37.9	44.7	53.4
Sim20k + real1k	<b>91.5</b>	<b>46.1</b>	<b>65.3</b>	53.5	<b>60.0</b>	<b>63.3</b>

Table 6. Evaluation on the *Test Set 1* (the nuScenes validation set), where the simulation data and the test set is from the same driving scenes. Rangenet++ [23] is trained on the different training sets for evaluations. \* mean IoU of four classes.

Test Set 2						
Training Set	road	vege.	terrain	vehicle	manmade	mIoU
Real 1k	97.0	78.7	73.9	76.5	85.7	82.4
Real 10k	<b>97.8</b>	<b>81.3</b>	<b>82.1</b>	80.1	<b>87.6</b>	<b>85.8</b>
Sim20k + real1k	97.0	79.5	77.6	<b>86.2</b>	85.3	85.1

Table 7. Evaluation on the *Test Set 2* (the nuScenes validation set), where the simulation data and the test set is from the different driving scenes. \* mean IoU of four classes.

Training Data	Average Precision (AP)
Real 1k	41.0
Simu 20k + real1k	44.0

Table 8. Evaluation on the 3D detection task. We train a 3D vehicle detector CenterPoint [40] on the real data as the baseline. Then, we pre-train the same detector model on the synthetic LiDAR data and further finetune the model on the real data. By introducing synthetic data in training the 3D detector. The average precision is significantly improved from 41.0 to 44.0.

### A.2. 3D Detection Task

We also conduct experiments on the 3D detection task. As shown in Table 8, the synthetic data achieved from our NeRF-LiDAR can significantly improve the 3D detection results. The average precision is improved by 7.3% (from 41.0 to 44.0). This further demonstrates the effectiveness of our NeRF-LiDAR.

### A.3. Performance Boost by Foreground-object Placement

To boost the perception performance for the foreground objects, more 3D vehicles were placed into the background LiDAR scenes. Firstly, 3D meshes of the cars were reconstructed with the real-world images. It was followed by arrangement of cars on the road according to the rendered semantic information. Finally point dropout are conducted to simulate the realistic cars in the LiDAR scenes. As shown in Table 9, by placing more vehicles into the NeRF scenes, the vehicle detection results are significantly improved by 10.5%.

Training Setting	Average Precision (AP)
W/o object placement	25.7
W/ object placement	28.4

Table 9. Effectiveness of object placement. We train the 3D vehicle detector CenterPoint [40] on the original NeRF-LiDAR data (9561 frames) and the augmented NeRF-LiDAR data with 15654 new cars inserted. The average precision is significantly improved by 10.5%.

### A.4. More Ablation Studies

**Label quality of the generated point clouds** In Table 10, we test the label quality of our generated LiDAR data. In training our NeRF-LiDAR, both the weak image labels (computed by the pre-trained SegFormer [36]) and the manually annotated semantic LiDAR point labels (if available) can be used in learning label generation. We design several experiments to test the label quality given different available labels for training.

In the first setting, only the weak image segmentation labels are used in training our NeRF-LiDAR for label rendering. We find that the weak image labels are effective in training our NeRF-LiDAR. Many label outliers can be resolved by constructing the multi-view geometry and consistency in the NeRF representation. It reports a mean IoU of 84.45 when compared with the ground-truth labels. This experiment implies that The generated labels are highly accurate and are effective in training 3D segmentation models (as demonstrated in Table 1 of the paper and Table 1& 2 of this supplementary).

In the second and the third settings, different amount of manual labels are provided. When using 20% of the manually labeled LiDAR point clouds (about 40 LiDAR frames) to boost the training of our NeRF-LiDAR, the label quality can be improved by 2.62% in mean IoU. When more manual labels are provided for training (50%, about 100 labeled LiDAR frames), the label accuracy can be further improved to 89.66% (a 5.21% improvement in mean IoU).

Label generation	road	vege.	terrain	vehicle	m.m.	mIoU
Image label	91.17	81.47	75.78	86.28	87.52	84.45
Image label + 20% LiDAR label	93.96	83.21	81.94	87.63	88.58	87.07
Image label + 50% LiDAR label	95.81	85.63	87.50	89.09	90.25	89.66

Table 10. Ablations on label generation. There are 200 labeled LiDAR point clouds provided in the nuScenes dataset (from the 10 nuScenes scenes used in the experiments). Besides the weak image labels, we also use 0%, 20% and 50% of these labeled LiDAR point clouds in training our NeRF-LiDAR for label generation. Other labeled LiDAR point clouds (the rest 50%) is used to compute the mean IoU for evaluating the quality of the generated labels. For each point in the real point clouds, we assign it a label using our NeRF-LiDAR. This point label is compared with the ground-truth label to compute the mean IoU.

**Ablation on balance weights** In the experiments, we set  $w_{rgb} = 1.0$ ,  $\hat{w}_l = 0.2$  and  $w_{feat} = 0.2$  for the balance weights in Eq. (11) and Eq. (15). We further perform ablation studies on these balance weights in Table 11. For the RGB balance weights  $w_{rgb}$ , we find that  $w_{rgb} = 1.0$  is the best setting. Lower or higher values will lead to worse performance with a drop of  $1.6 \sim 3.6\%$  in mIoU. For the semantic label weight  $\hat{w}_l$  that is used for training label generation,  $[0.2, 1.0][0.2, 1.0]$  is a reasonable range for the label balance weight  $\hat{w}_l$ . To also preserve a high-quality RGB image rendering, we use default  $\hat{w}_l = 0.2$  (a smaller weight), instead of 1.0 (that can further improve the mean IoU by another 0.5%). And for the feature loss used in point cloud raydrop & alignment, we use  $w_{feat} = 0.2$ , which is the best setting. Using lower or higher values cannot achieve better mean IoU in training 3D segmentation networks.

Balanced weights						Segmentation results
NeRF reconstruction			Generation			
$w_{rgb}$	$\hat{w}_l$	$w_{feat}$				mIoU
0.5	1	2	0.05	0.2	1	0.05 0.2 0.5
✓			✓			79.7
	✓		✓			<u>81.5</u>
	✓	✓	✓			77.7
✓	✓		✓			81.3
✓		✓	✓			<u>81.5</u>
✓		✓	✓			82.0
✓	✓		✓			80.1
✓		✓	✓			<u>81.5</u>
✓	✓			✓		80.4

Table 11. Ablations on balanced weights. We train the cylinder3d model [46] on the generated LiDAR point clouds. 2k simulation point clouds is generated. *Test set 1* is used for evaluation. The default setting is highlighted with underlines.

**LiDAR depth supervision** In training our NeRF-LiDAR, unlabeled real LiDAR point clouds play an important role. LiDAR points provided extra 3D information to improve the accuracy of the NeRF reconstruction of the real world. Furthermore, it can supervise the NeRF-LiDAR model to make the generated point clouds more realistic.

In Table 12, we demonstrate the effectiveness of using unlabeled LiDAR point clouds to train our NeRF-LiDAR. We train our NeRF reconstruction with and without LiDAR depth supervision respectively. Then we generate simulation data and train 3D segmentation network [46] on these generated data respectively. When training without LiDAR depths (only images are used), the 3D segmentation network only achieves a mean IoU of 63.7%. When the LiDAR depths are provided for training, the accuracy is significantly improved to 84.4% (with an improvement of 20.7% in mean IoU).

Depth supervision	road	vege.	terrain	vehicle	manmade	mIoU
No depth supervision	75.0	54.4	48.1	71.4	69.5	63.7
Ours	89.3	80.0	74.5	91.1	87.2	84.4

Table 12. Ablations on the LiDAR depth supervision. LiDAR depth is used to reconstruct more accurate 3D geometry in learning the NeRF representation and also make the generated point clouds more realistic. We studied the effects of using unlabeled real LiDAR point clouds in training our NeRF-LiDAR.

**Ablation on the number of reconstructed sequences** Ablation studies was performed on the number of sequences used to verify the effectiveness of enriching the reconstructed

scenes. Referring to Fig. 7 and Table 7, it revealed that reconstructing more scenes can improve the effect of simulation data. Only 30 scenes were reconstructed in the whole dataset due to the limited computing resource. Under the circumstances that nuScenes dataset contains over 1000 scenes, it was believed that the effect of simulation data would definitely be further improved with more scenes reconstructed.

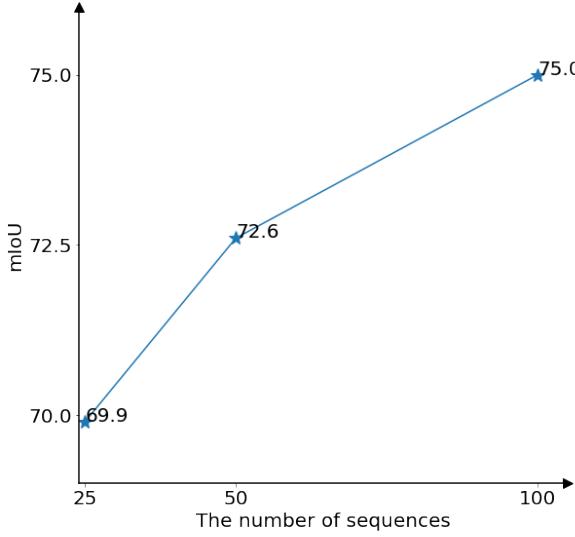


Figure 7. Ablation on the number of used sequences

## B. More Comparisons and Analysis

### B.1. RGB Image and LiDAR simulation

Our NeRF-LiDAR can simultaneously simulate the RGB novel views and the LiDAR point clouds. This makes it possible to develop and verify multi-sensor autonomous driving systems in our NeRF-LiDAR system. Such multi-sensor simulation is impossible for any other existing LiDAR simulation methods.

In Fig. 8, we present the simulation example of a novel sensor configuration which is different from the training data (nuScenes [5]). This enables us to design and verify new sensor configurations and autonomous driving systems through the NeRF-LiDAR simulator. We also provide the RGB metrics with other NeRF such as Urban-NeRF [26].

	PSNR↑	SSIM ↑	LPIPS↓
NeRF-LiDAR	<b>25.19</b>	<b>0.802</b>	<b>0.328</b>
Urban-NeRF [26]	21.12	0.673	0.503

Table 13. Evaluation of NeRF-LiDAR and Urban-NeRF using RGB metrics in the reconstructed scenes.

### B.2. Novel LiDAR view rendering

New sensor settings can be synthesized for self-driving cars (*e.g.* fewer or more video sensors), which is impossible for LiDARSim [21]. The sensors can be rotated and translated to generate rarely seen hard cases along with RGB rendering for video sensors.

### B.3. Visual Comparisons

In Fig. 9, we visualize the 3D segmentation results. Our simulated LiDAR point clouds can be used to boost the training of the 3D segmentation models. Compared with the model trained only with real data, the model that is pre-trained on our generated data and fine-tuned on the real data achieves far better segmentation accuracy. The segmentation accuracy of the five classes are all significantly improved.

### B.4. NeRF-LiDAR vs Virtual-world Creation

Compared with the LiDAR simulators (*e.g.* CARLA [8], AIRSIM [29]) that is based on virtual-world creation, our NeRF-LiDAR leverages the real-world information for point-cloud simulation. As shown in Fig. 10, Our LiDAR-NeRF can generate point clouds that are almost the same as the real LiDAR data (Fig. 10c). CARLA (Fig. 10a) tends to generate evenly distributed LiDAR points. This is because the manually created 3D virtual world cannot simulate the complexity of the real world. Guillard *et al.* [14] learns a deep neural network for raydrop to make the CARLA LiDAR simulator more realistic. However it still cannot simulate the complex details of the real world, such as the scene organization, object distributions and various shapes, surfaces and sizes of the real world contents.

### B.5. NeRF-LiDAR vs Generative Models

The generative models utilize generative adversarial networks (GAN) [4, 19, 28] or diffusion model [47] for point cloud generation. It utilizes a large number of real LiDAR point clouds to train the LiDAR generators. Even though these generated point clouds are visually similar to the real point clouds, they cannot be used to train deep neural networks for segmentation or detection. This is because they have significant domain gaps with the real LiDAR point clouds in details (as compared in Fig. 10b) and the existing LiDAR generative models cannot generate accurate point labels for training 3D segmentation models.

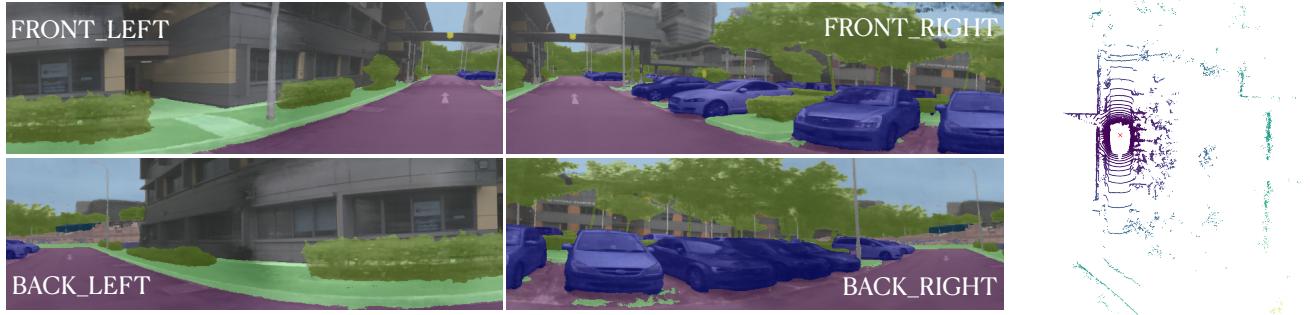


Figure 8. NeRF-LiDAR can also render high-quality RGB novel views along the the LiDAR. We simulate a 4 cameras and one LiDAR sensor configuration that is different from the neScenes configuration in the training dataset.

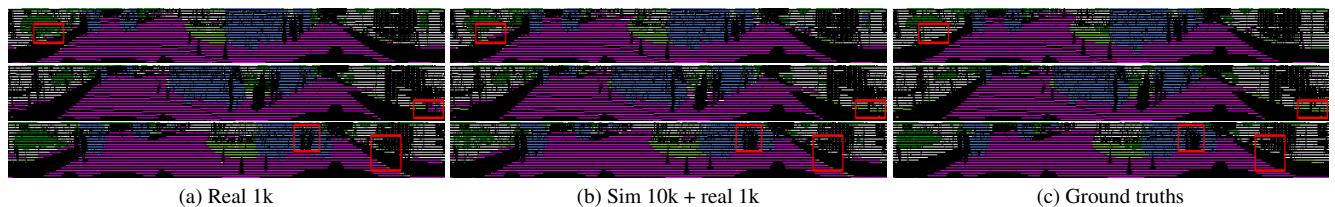


Figure 9. Visualization of the 3D segmentation results. Cylinder3D [46] is trained on different datasets for comparisons. Model (a) is trained on real 1k LiDAR data. Model (b) is pre-trained on the 10k simulation data generated by our NeRF-LiDAR and then fine-tuned on the 1k real data. (c) Ground truth lalels. As highlighted by the red windows, the LiDAR data generated by our NeRF-LiDAR can significantly improve the segmentation accuracy. *Zoom in to see details*

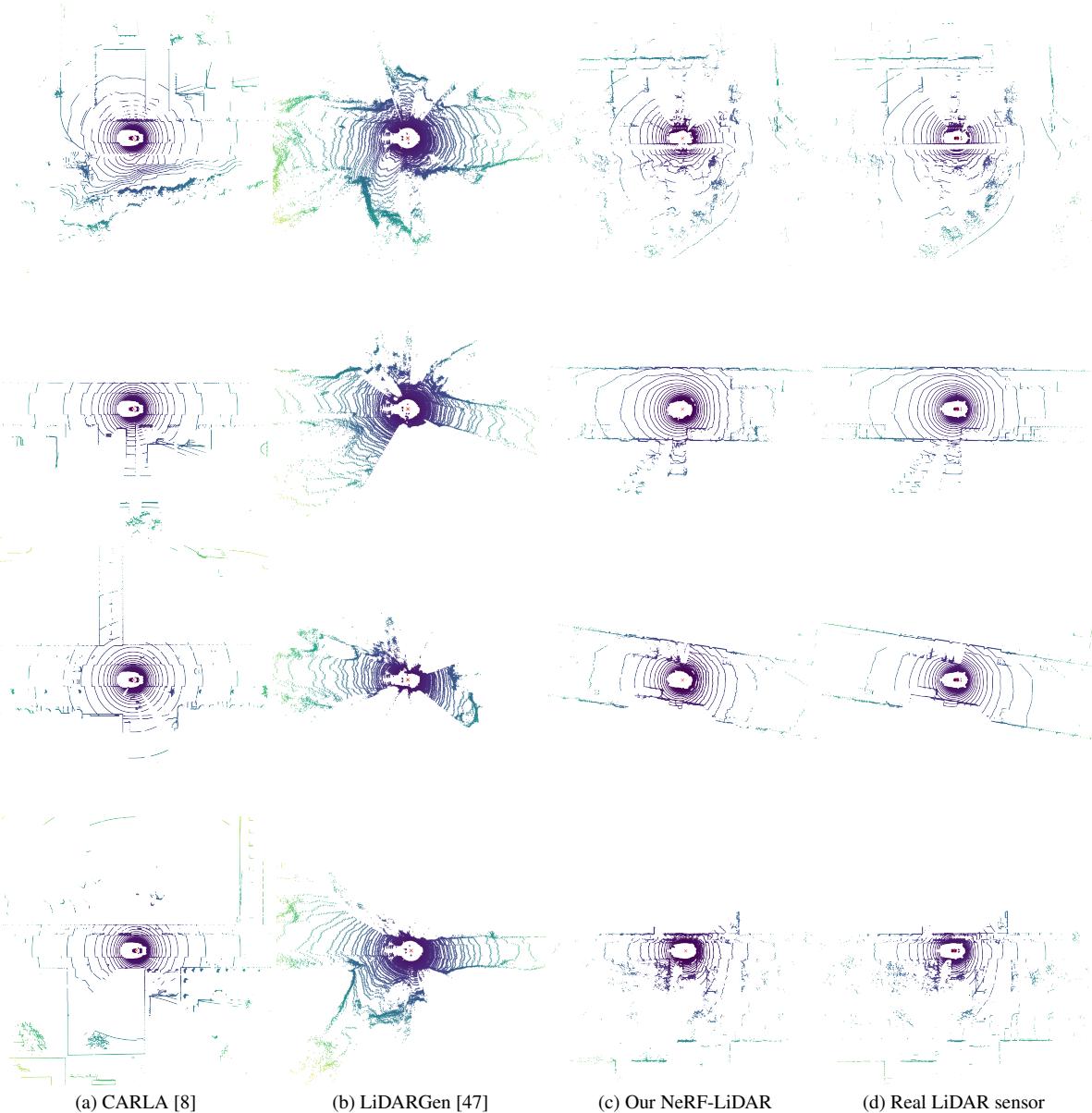


Figure 10. Comparisons of results between our NeRF-LiDAR and other existing LiDAR simulation methods. (a) Method [8] that creates virtual world for LiDAR simulation. (b) Diffusion model used for LiDAR generation [47]. (c) Our NeRF-LiDAR can generate realistic point clouds that is nearly the same as the real LiDAR point clouds (d). Real LiDAR sensor *Zoom in to see details*