

LiDAR-NeRF: Novel LiDAR View Synthesis via Neural Radiance Fields

Tang Tao^{1*} Longfei Gao² Guangrun Wang³ Peng Chen² Dayang Hao²
 Xiaodan Liang^{1\\$} Mathieu Salzmann⁴ Kaicheng Yu²

¹ Shenzhen Campus, Sun Yat-sen University, China

² Autonomous Driving Lab, Alibaba Group, China

³ University of Oxford, UK ⁴ CVLab, EPFL, Switzerland

{trent.tangtao, kaicheng.yu.yt, hxdliang328}@gmail.com

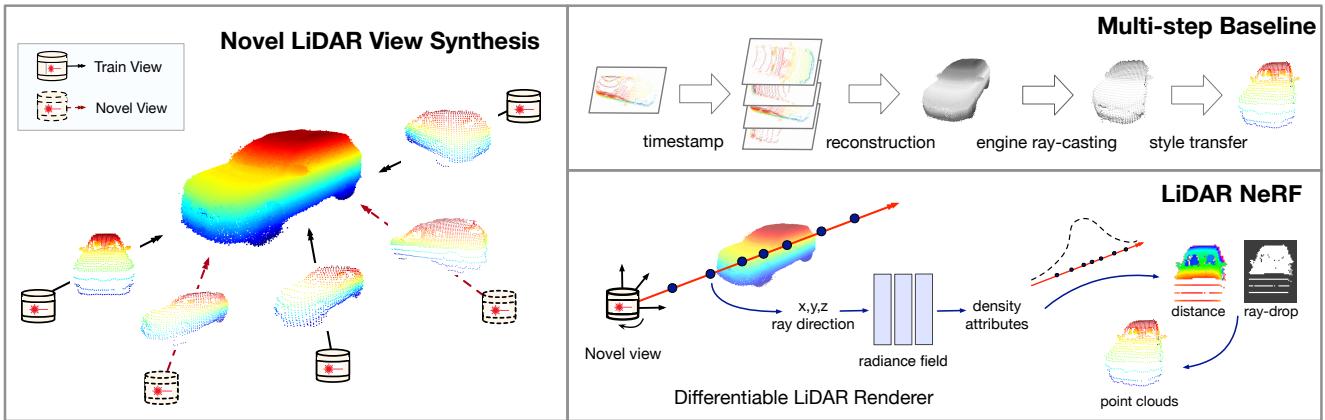


Figure 1: **(Left)** We introduce the task of novel view synthesis for LiDAR sensors. Given multiple LiDAR viewpoints of an object, novel LiDAR view synthesis aims to render a point cloud of the object from an arbitrary new viewpoint. **(Right, Top)** The mostly-closed related approaches to generating new LiDAR point clouds follow a multi-step strategy, recovering a 3D mesh from the input point clouds and leveraging game engines to simulate a new point cloud. The multiple steps involved in this strategy affect its applicability and scalability. Furthermore, it usually fails to produce realistic LiDAR patterns as it ignores some important attributes of LiDAR points. **(Right, Bottom)** By contrast, we propose a novel framework, LiDAR-NeRF, that is composed of a differentiable LiDAR renderer with an associated neural radiance field, to avoid explicit 3D reconstruction and the use of a game engine. Our method enables end-to-end optimization and encompasses the 3D point attributes into the parameters of the learnable field.

Abstract

We introduce a new task, novel view synthesis for LiDAR sensors. While traditional model-based LiDAR simulators with style-transfer neural networks can be applied to render novel views, they fall short in producing accurate and realistic LiDAR patterns, because the renderers they rely on exploit game engines, which are not differentiable. We address this by formulating, to the best of our knowledge, the first differentiable LiDAR renderer; and propose an end-to-end framework, LiDAR-NeRF, leveraging a neural radi-

ance field (NeRF) to enable jointly learning the geometry and the attributes of 3D points. To evaluate the effectiveness of our approach, we establish an object-centric multi-view LiDAR dataset, dubbed NeRF-MVL. It contains observations of objects from 9 categories seen from 360-degree viewpoints captured with multiple LiDAR sensors. Our extensive experiments on the scene-level KITTI-360 dataset, and on our object-level NeRF-MVL show that our LiDAR-NeRF surpasses the model-based algorithms significantly.

*Work done during an internship at DAMO Academy, Alibaba Group.

\\$Corresponding Author.

1. Introduction

Synthesizing novel views of a scene from a given camera is a long-standing research topic. A recent milestone in this area has been to combine differentiable rendering with neural radiance fields (NeRF) [19], resulting in a de-facto standard to render photo-realistic novel views by leveraging only a hundred or fewer input images with known camera poses. Impressively, this has already been shown to positively impact downstream tasks such as autonomous driving [23, 26, 31, 35].

In such an autonomous driving scenario, however, practical systems typically exploit not only images but also LiDAR sensors, which provide reliable 3D measurements of the environment. As such, it seems natural to seek to generate novel views not only in the image domain but also in the LiDAR one. However, the only methods that consider LiDAR point clouds for novel view synthesis [26, 35] only do so to boost training, thus still producing images as output. In other words, generating novel LiDAR views remains unexplored. Despite the 3D nature of this modality, this task remains challenging, as LiDARs only provide a partial, 2.5D, view of the scene, corrupted by sensor noise.

In this paper, we therefore introduce the first approach to novel LiDAR view synthesis. Unlike RGB view synthesis, the output of a free viewpoint LiDAR sensor is a point cloud sampled from the surrounding 3D scene according to the given LiDAR sensor specific pattern, as illustrated in Fig. 1. Therefore, one cannot directly apply the NeRF formalism, which relies on a photometric loss. To address this, we construct a differentiable LiDAR renderer that first converts point clouds with respect to a surface plane, serving as a 360-degree range pseudo image in which each pseudo pixel represents the distance between the LiDAR receiver and a world point hit by a laser beam.

We then use a neural network to encode the 3D information and predict multiple attributes for each pseudo pixel. Specifically, we regress the distance of each pseudo pixel, which represents their 3D coordinate, its intensity, which encodes the amount of reflected light that reaches the sensor at a pseudo pixel, and an attribute that we dub ray-drop, which encodes the probability of dropping a pseudo pixel. This last attribute reflects our observation that, in the real world, some laser beams of the LiDAR sensor are simply lost, due to the surface material and normal direction. As image-based NeRFs, our LiDAR-NeRF approach leverages multi-view consistency, thus enabling the network to produce accurate geometry.

Validating the effectiveness of our approach can be achieved by leveraging the existing autonomous driving datasets [16, 2, 30] that provide LiDAR data. However, as these datasets were acquired from a vehicle moving along the street, the objects they depict are observed with limited viewpoint variations, thus making them best-suited

for scene-level synthesis. This contrasts with object-level synthesis, as is more common in image novel-view synthesis [19, 17, 13]. We therefore establish an object-centric multi-view LiDAR dataset, which we dub the NeRF-MVL dataset, containing carefully calibrated sensor poses, acquired from multi-LiDAR sensor data from real autonomous vehicles. It contains more than 76k frames covering two types of collecting vehicles, three LiDAR settings, two collecting path, and nine object categories.

We evaluate our model’s scene-level and object-level synthesis ability on scenes from the challenging KITTI-360 dataset [16] and from our NeRF-MVL dataset both quantitatively and qualitatively. For comparison, we adapt existing model-based LiDAR simulators [18, 15] to create a baseline renderer. Our results demonstrate the superior performance of our approach compared to the baseline renderer in various metrics and visual quality, showcasing its effectiveness in LiDAR novel view synthesis.

Overall, we make the following contributions:

- We formulate the first differentiable framework, LiDAR-NeRF, for novel LiDAR view synthesis, which can render novel point clouds with point intensity and ray-drop probability without explicit 3D reconstruction.
- We establish the NeRF-MVL dataset from LiDAR sensors of real autonomous vehicles, to evaluate the object-centric novel LiDAR view synthesis.
- We demonstrate the effectiveness of our LiDAR-NeRF quantitatively and qualitatively in both scene-level and object-level novel LiDAR view synthesis.

2. Related Work

2.1. Novel RGB View Synthesis

Synthesizing novel RGB views of a scene from a set of captured images is a long-lasting problem. In particular, recent advances in neural radiance fields (NeRF) have demonstrated their superior performance in synthesizing images, thanks to the pioneering work of [19]. Following this, many NeRF strategies haven been proposed for acceleration [20, 36, 3] and generalization [37, 11, 22]. Noticeably, notions of depth have been used for novel RGB view synthesis. In particular, DS-NeRF [4] utilizes sparse depth to supervise the NeRF training; DONeRF [21] and Roessel et al. [27] both explore depth priors to improve ray sampling and accelerate rendering. In parallel, great progress has been made to handle complex environments, such as large-scale outdoor scenes [26, 31, 35], demonstrating the tremendous potential of NeRFs for real-world applications. Nevertheless, while these works improve quality and convergence speed, they still produce RGB images. In practical scenarios where multiple sensors, such as RGB cameras and LiDARs, are used, only synthesizing the image view is insufficient. In this work, drawing inspiration from

NeRF [19], we introduce the first framework for novel LiDAR view synthesis.

2.2. Model-based LiDAR Simulators

There are model-based LiDAR simulators that can also be regarded as LiDAR renderers. In this context, early works [5, 39, 33, 34] employ graphics engines, such as CARLA [6], Unreal Engine 4 (UE4) [29] and Grand Theft Auto V (GTA V), to simulate LiDAR sensors. However, this yields a large sim-to-real domain gap, as their virtual worlds use handcrafted 3D assets and make simplified physics assumptions, resulting in perfect scene point clouds without any modeling of material or other properties. Moreover, constructing diverse model assets and complex scenarios requires professional 3D experts and is labor-intensive. More recent works, such as LiDARsim [18] and PCGen [15], employ a multi-step, data-driven approach to simulate point clouds from real data. They first leverage real data to reconstruct both background and foreground objects, and then utilize the reconstructed 3D scene to render novel LiDAR data via ray-casting. In an attempt to close the gap between real and simulated point clouds, they further train a network to model the physics of LiDAR ray-dropping. However, the multiple steps involved in this approach affect its applicability and scalability. By contrast, our approach is simple and effective, yet produces realistic LiDAR data, and can be thought of as the first differentiable LiDAR renderer.

3. Novel LiDAR View Synthesis

In this section, we first give a formal problem definition of novel LiDAR view synthesis, and introduce our LiDAR-NeRF in detail. Finally, we describe our object-level multi-view LiDAR dataset.

3.1. Problem Definition and Preliminaries

Novel view synthesis of LiDAR sensors aims to render an object or scene from an arbitrary new viewpoint given a set of existing observations acquired from other viewpoints. Formally, given a set $\mathcal{D} = \{(P_i, G_i)\}$, where P_i is the LiDAR pose and G_i is the corresponding observed point cloud, we aim to define a rendering function f that can generate a new point cloud from an arbitrary new pose P' , i.e., $G' = f_{\mathcal{D}}(P')$. The generated point cloud should be evaluated w.r.t. the associated ground-truth one in three aspects: i) absolute geometric error; ii) point distribution similarity; and iii) realism of point attributes such as intensity value. To produce accurate and realistic novel LiDAR views, we draw inspiration from the NeRF formalism. We therefore first review image-based NeRF below.

NeRF Revisited. NeRF represents a scene as a continuous volumetric radiance field. For a given 3D point $\mathbf{x} \in \mathbb{R}^3$ and a viewing direction θ , NeRF learns an implicit function f

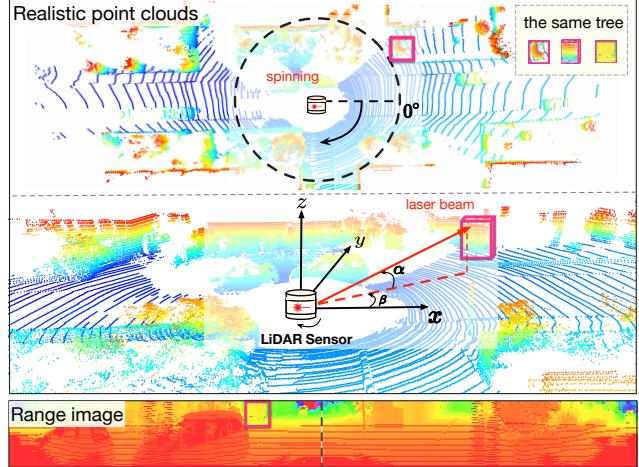


Figure 2: LiDAR model and range image representation. **(Top)** The physical model of a LiDAR can be described as follow. Each laser beam originates from the sensor origin and shoot outwards to a point in the real world or vanishes. One common pattern of laser beam is spinning in a 360-degree fashion. **(Bottom)** We convert the point clouds into a range image, where each pixel corresponds to a laser beam. Note that we highlight one object in the different views to facilitate the visualization.

that estimates the differential density σ and view-dependent RGB color \mathbf{c} as $(\sigma, \mathbf{c}) = f(\mathbf{x}, \theta)$.

NeRF uses volumetric rendering to render image pixels. Given a pose \mathbf{P} , it casts rays \mathbf{r} originating from \mathbf{P} 's center of projection \mathbf{o} in direction \mathbf{d} , i.e., $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$. The implicit radiance field is then integrated along this ray, and the color is approximated by integrating over samples lying along the ray. This is expressed as

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

where $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$ indicates the accumulated transmittance along ray \mathbf{r} to the sampled point t_i , \mathbf{c}_i and σ_i are the corresponding color and density at t_i , and $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples.

Moreover, per-view depth $\hat{D}(\mathbf{r})$ can also be approximated by calculating the expectation of the samples along the ray, i.e.,

$$\hat{D}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) t_i . \quad (2)$$

However, one cannot directly apply the NeRF formalism, which leverages a per-pixel photometric error measure, to novel LiDAR view synthesis, where the observations are 3D points. The most closely-related approaches [18, 15]

leverage game engines to simulate LiDAR data. However, they are not differentiable, and can therefore not be optimized in an end-to-end manner based on given point cloud observations. Here, we address this by introducing the first differentiable LiDAR renderer.

3.2. Differentiable LiDAR Renderer

Let us start with the LiDAR model as shown in Fig. 2 (Top), which works by emitting a laser beam and measuring the time it takes for the reflected light to return to the sensor. For a LiDAR with H laser beams in a vertical plane and W horizontal emissions, the returned attributes (e.g., distance d and intensity i) form an $H \times W$ range pseudo image. The Cartesian coordinates (x, y, z) of 3D point can then be computed from polar coordinates as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = d \begin{pmatrix} \cos(\alpha) \cos(\beta) \\ \cos(\alpha) \sin(\beta) \\ \sin(\alpha) \end{pmatrix} = d\theta, \quad (3)$$

where α is the vertical rotation, i.e., the pitch angle, β is the horizontal rotation, i.e., the yaw angle, and θ denotes the viewing direction in the local sensor coordinate system. Specifically, for the 2D coordinates (h, w) in the range pseudo image, we have

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} |f_{\text{up}}| - h f_v H^{-1} \\ -(2w - W) \pi W^{-1} \end{pmatrix}, \quad (4)$$

where $f_v = |f_{\text{down}}| + |f_{\text{up}}|$ is the vertical field-of-view of the LiDAR sensor. Conversely, each 3D point (x, y, z) in a LiDAR frame can be projected on a range pseudo image of size $H \times W$ as

$$\begin{pmatrix} h \\ w \end{pmatrix} = \begin{pmatrix} (1 - (\arcsin(z, d) + |f_{\text{down}}|) f_v^{-1}) H \\ \frac{1}{2} (1 - \arctan(y, x) \pi^{-1}) W \end{pmatrix}. \quad (5)$$

Note that if more than one point project to the same pseudo pixel, only the point with the smallest distance is kept. The pixels with no projected points are filled with zeros. In addition to the distance, the range image can also encode other point features, such as intensity.

3.3. LiDAR-NeRF Framework

Motivated by the impressive results of NeRF [19] for novel RGB view synthesis, we therefore introduce the first novel LiDAR view synthesis framework.

Implicit fields to represent LiDAR sensor. As discussed in Sec. 3.2, the LiDAR sensor employs an active imaging system, in contrast to the passive imaging principle of cameras. Furthermore, each pseudo pixel in the LiDAR range image corresponds to a real laser beam, which is more consistent with the rays in NeRF. We therefore reformulate NeRF to achieve novel LiDAR view synthesis. However,

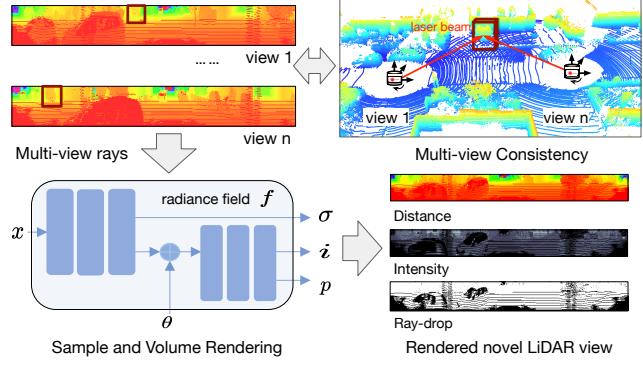


Figure 3: **Our LiDAR-NeRF framework.** Taking multi-view LiDAR range images with associated sensor poses as input, our model produces 3D representations of the distance, the intensity and the ray-drop probability at each pseudo pixel. We exploit multi-view consistency of the 3D scene to help our network produce accurate geometry.

one cannot trivially apply image-based NeRF to LiDAR range images; the sensor’s specificities must be modeled.

For a given LiDAR range image, the laser’s viewing directions θ of a pseudo pixel can be calculated according to Eq. (3) and Eq. (4). The viewing directions in our LiDAR-NeRF form a radial pattern, as shown in Fig. 2, closely matching physical reality. The expected depth obtained by integrating over samples in Eq. (2) is the distance from the LiDAR sensor. Moreover, the origin \mathbf{o} and viewing direction θ of the ray are both transformed to the global world coordinate system. Therefore, the sampled points t are actual points in the real world, which are consistent across multiple LiDAR frames/range images, as shown in the top-right portion of Fig. 3.

With these geometry aspects in mind, we design a framework able to 1) synthesize a novel LiDAR frame with realistic geometry; 2) estimate LiDAR intensities over the scene; and 3) predict a binary ray-drop mask that specifies where rays will be dropped. As mentioned in Sec. 3.2, ray-drop and intensity can be recorded as a color feature of LiDAR [24], and both of them are view-dependent. For the radiance field, we follow the original implementation of NeRF [19], which uses two successive MLPs. We utilize the first MLP to estimate the density σ and the expected distance d . The second MLP predicts a two channel feature map as in [9]: Intensities \mathbf{i} and ray-drop probabilities \mathbf{p} , respectively. Then, in the same way as for color, we can compute the per-view intensity and ray-drop probability by

integrating along a ray \mathbf{r} as

$$\begin{aligned}\hat{I}(\mathbf{r}) &= \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{i}_i, \\ \hat{P}(\mathbf{r}) &= \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{p}_i.\end{aligned}\quad (6)$$

Altogether, our LiDAR-NeRF can be formalized as a function $(\sigma, \mathbf{i}, \mathbf{p}) = f(\mathbf{x}, \theta)$, and is summarized in Fig. 3.

Loss Function. Our loss function includes three objectives

$$\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{intensity}(\mathbf{r}) + \lambda_2 \mathcal{L}_{raydrop}(\mathbf{r}) + \mathcal{L}_{distance}(\mathbf{r}), \quad (7)$$

with

$$\begin{aligned}\mathcal{L}_{intensity}(\mathbf{r}) &= \sum \|\hat{I}(\mathbf{r}) - I(\mathbf{r})\|_2^2, \\ \mathcal{L}_{raydrop}(\mathbf{r}) &= \sum \|\hat{P}(\mathbf{r}) - P(\mathbf{r})\|_2^2, \\ \mathcal{L}_{distance}(\mathbf{r}) &= \sum_{\mathbf{r} \in R} \|\hat{D}(\mathbf{r}) - D(\mathbf{r})\|_2^2,\end{aligned}\quad (8)$$

where R is the set of training rays, and λ_1 and λ_2 are two hyper-parameters to balance the influence of the terms.

Distance-Guided Sampling. Since we have ground truth distances, the distance prior contains valuable information to guide sampling along a ray, which can effectively improve the LiDAR-NeRF training. Inspired by [21, 27], we replace the fine network used for hierarchical sampling with only 4 samples around the distance prior. During optimization, the samples of the coarse network are uniformly distributed between the near and far planes, and the 4 additional distance-guide samples are drawn from a Gaussian distribution determined by the distance prior. At test time, when the distance is unknown, we use the rendered distance from our baseline LiDAR renderer.

3.4. NeRF-MVL Dataset

As will be shown in our experiments, our approach can be applied to existing autonomous driving datasets [16, 2, 30] that have LiDAR sensors data. However, these datasets focus on scene-level LiDAR observations, and thus depict views acquired from the vehicle driving along the scene, with fairly low diversity. In other words, they lack the challenging diversity of object-centric data similar to that used for novel RGB view synthesis. To facilitate future research in novel LiDAR view synthesis and verify the effectiveness of our LiDAR-NeRF, we therefore establish an object-centric multi-view LiDAR dataset, NeRF-MVL, with carefully calibrated sensor poses and gathering multi-LiDAR sensor data from real autonomous vehicles.

Data Collection. We collect the dataset in an enclosed area, employing self-driving vehicles with multiple LiDAR sensors. The vehicles drive around the object in a square path

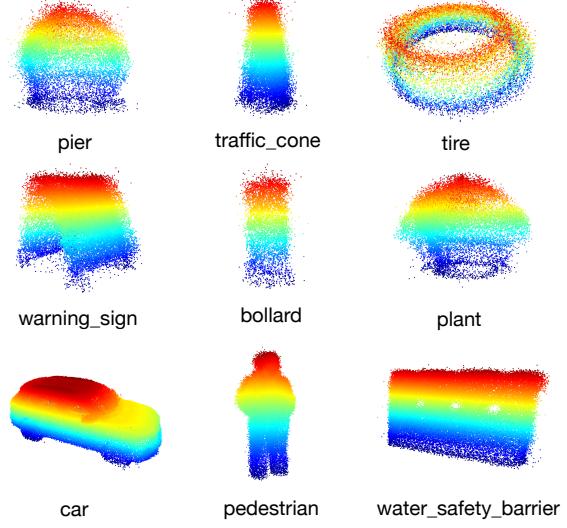


Figure 4: **Dataset visualization.** Our NeRF-MVL dataset encompasses 9 objects from common traffic categories. We align multiple frames here for better visualization.

Table 1: **LiDAR sensor configurations.**

Sensor	Details
LiDAR	Spinning, 64 beams, 10Hz capture frequency, 360° horizontal FOV, 0.6° horizontal resolution, -52.1° to +52.1° vertical FOV, ≤ 60m range, ±3cm accuracy.
LiDAR-F	Spinning, 64 beams, 20Hz capture frequency, 360° horizontal FOV, 0.4° horizontal resolution, -25° to +15° vertical FOV, ≤ 200m range, ±2cm accuracy.

Sensor location: F: front. T: top.

twice, one large square and one small square, as shown in Fig. 8 of the appendix. To provide more diverse perspectives, we use various types of vehicles with different sensor placements and specifications. See Tab. 1 for sensor details.

Data Preparation. As shown in Fig. 4, our NeRF-MVL dataset consists of nine objects from different common traffic categories. After collecting multi-path, multi-sensor data, for each object, we manually select the LiDAR frames when the vehicles are driving on the four straight edges of the square path. We then crop out the region of interest, i.e., the object¹. We carefully calibrate the LiDAR extrinsics of every sensor, i.e., the relative location of the LiDAR to the ego body. The transformation matrix from the body coordinate system to the global world coordinate system is provided from the vehicle location based on GPS and IMU. Hence, in the dataset, we finally provide the calibration of the LiDAR to the global world, i.e., the *lidar2world* matrix,

¹The raw data will also be released to the community.

to align all the frames.

Altogether, our NeRF-MVL dataset contains more than 76k frames covering two types of collecting vehicles, three LiDAR settings, two collecting paths, and nine objects. For fast validation, we extract a pocket-version of the dataset with only 7.3k frames covering the nine categories.

4. Experiments

We evaluate the scene-level and object-level synthesis ability of our LiDAR-NeRF both quantitatively and qualitatively. Additional results and details are provided in the supplementary material.

Baseline renderer. As generating novel LiDAR views remains unexplored, we moderately adapt existing model-based LiDAR simulators to form a baseline. Specifically, we use our differentiable LiDAR renderer to replace 3D reconstruction and ray-tracing. For each laser beam, we then select the closest rendered point. We validate different settings of the baseline method in Appendix A and report the best value in the following sections.

Dataset. We conduct the scene-scale experiments on the challenging KITTI-360 [16] dataset, which was collected in suburban areas. Following Panoptic NeRF[7], we evaluate LiDAR-NeRF on LiDAR frames from 4 static suburb sequences. Each sequence contains 64 frames, with 4 equidistant frames for evaluation. We conduct the object-level experiments on our NeRF-MVL dataset, which contains 9 object categories.

Metrics. For the novel LiDAR range images, we follow common practice in depth estimation, i.e., capping distance to 80m according to the LiDAR model and computing the usual metrics [8]: Root mean squared error (RMSE), and threshold accuracies (δ_1 , δ_2 , δ_3). To further evaluate the novel LiDAR view quality, we convert the rendered LiDAR range image to a point cloud based on Eq. (3) and compute the Chamfer Distance (C-D)² between the original and the novel point clouds G_1 , G_2 . It is computed as

$$\text{C-D} (G_1, G_2) = \frac{1}{|G_1|} \sum_{x \in G_1} \min_{y \in G_2} \|x - y\|_2^2 + \frac{1}{|G_2|} \sum_{y \in G_2} \min_{x \in G_1} \|y - x\|_2^2. \quad (9)$$

We also report the F-Score between the two point clouds with a threshold of 5cm. For the novel intensity image, we measure the quality of the synthesized intensities using the PSNR and SSIM [32].

Training details. We optimize our LiDAR-NeRF model per scene using a single Tesla v100 GPU. For each scene, we center the LiDAR point clouds by subtracting the origin of the global world coordinate system from the scene central

²We utilize the implementation of [ChamferDistancePytorch](#).

frame. Then, the scene frames are scaled by a factor such that the region of interest falls within a unit cube, which is required by most positional encodings used in NeRFs. For LiDAR-NeRF, the coarse and fine networks are sampled 64 and 128 times, respectively, during training. The highest frequency of the coordinates is set to 2^{15} . We use Adam [12] with a learning rate of 5e-4 to train our models. We set the loss weights to $\lambda_1 = 1$, $\lambda_2 = 1$, and optimize the total loss \mathcal{L}_{total} for 400k iterations with a batch size of 2048. For our NeRF-MVL dataset, we first get the 3D box of each object, and then project to the range view. Only a few rays within the box are trained, so the network converges quickly.

4.1. Scene-level Synthesis

We first evaluate the effectiveness of our LiDAR-NeRF on scene-level novel LiDAR view synthesis. The results are provided in Tab. 2. Although the different metrics emphasize distinct aspects of the results, our LiDAR-NeRF significantly outperforms the baseline renderer over all metrics. To be specific, LiDAR-NeRF is superior to the baseline renderer with a comfortable margin in terms of C-D (0.126 vs 0.259), δ_1 (78.26 vs 65.06) and intensity PSNR (15.83 vs 13.17). Moreover, the results also confirm the effectiveness of our view-dependent ray-drop design, as the use of ray-drop generally improves the metrics, especially the range distance ones. Furthermore, distance-guided sampling greatly reduces the training time from 70 hours to 16 hours on a single GPU and yields competitive results.

In Fig. 5, we provide qualitative results. Both methods are able to render general scene details. However, the range view distance and intensity images rendered by the baseline renderer are overly smooth and complete. This is because the baseline relies on ray-casting from the densely aligned scene, and every ray that intersects the dense surface contributes a point. By contrast, our LiDAR-NeRF can jointly render novel point clouds and their intensity, with the additional attribute ray-drop, and produces more realistic LiDAR patterns. As shown in the bottom portion of the figure, the point cloud rendered by LiDAR-NeRF from a validation viewpoint presents a similar geometry and structure to the corresponding real point cloud, whereas the baseline yields an overly dense and perfect one.

4.2. Object-level Synthesis

We conduct our object-level synthesis experiments on the nine common traffic objects in our NeRF-MVL dataset. As shown in Tab. 3, our LiDAR-NeRF still significantly outperforms the baseline renderer by a large margin over all the metrics, except for the C-D and the F-score, on all nine categories. Furthermore, the qualitative visualization in Fig. 6 evidence that our approach produces significantly higher quality point clouds than the baseline. Our ray-drop also clearly enhances realism.

Table 2: **Novel LiDAR view synthesis on KITTI-360.** LiDAR-NeRF outperforms the baseline in all metrics.

Method	Component	C-D↓	F-score↑	RMSE↓	$\delta 1\uparrow$	$\delta 2\uparrow$	$\delta 3\uparrow$	PSNR↑	SSIM↑
Base Renderer	–	0.259	83.87	4.312	65.06	67.86	69.09	13.17	0.062
LiDAR-NeRF	Guided sampling + Ray-drop channel	0.165	84.23	4.330	66.50	68.64	69.52	15.42	0.116
	Vanilla + Ray-drop channel	0.163	84.13	4.344	67.71	69.23	69.84	16.06	0.134
		0.126	87.64	3.948	78.26	79.57	80.09	15.83	0.202

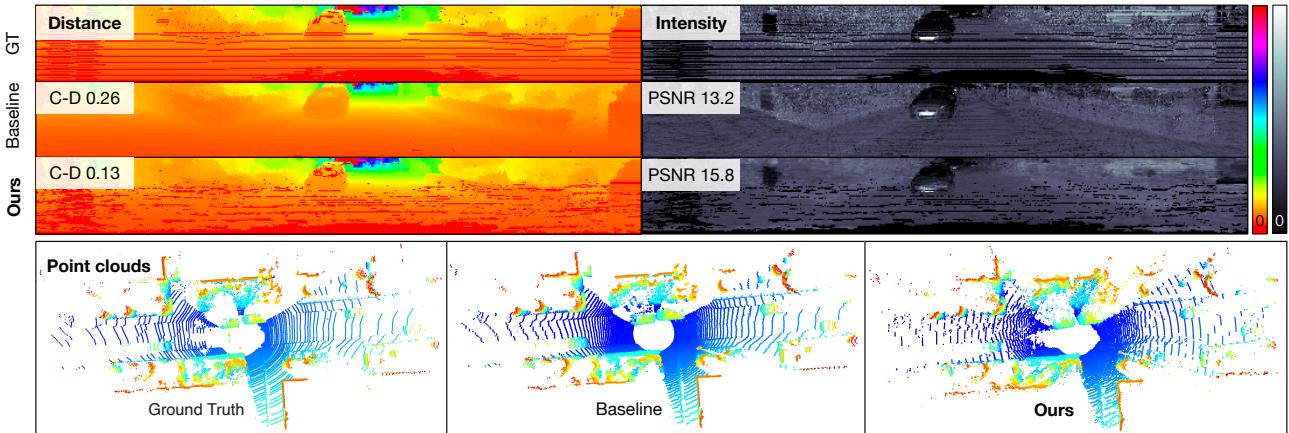


Figure 5: **Qualitative comparison on KITTI-360.** The top block demonstrates that our LiDAR-NeRF produces more realistic LiDAR patterns in range view distance and intensity images. The point clouds in the bottom portion evidence the highly detailed structure and geometry present in our rendered LiDAR views.

We conjecture that the better C-D and F-score of the baseline are due to the fact that it generates overly dense point clouds, as can be seen in Fig. 6, which is favorable to these two metrics. To verify this, we further compute the Earth Mover’s Distance (EM-D) [28]³, which considers the probability distribution of the points and is more robust to outliers. Furthermore, we also monitor the ratio of the number of generated points n_{novel} to that of the actual point cloud n_{real} . Specifically, we define the error

$$\text{N-P} (G_{\text{novel}}, G_{\text{real}}) = \left| 1 - \frac{n_{\text{novel}}}{n_{\text{real}}} \right|. \quad (10)$$

As can be observed in Tab. 3, our LiDAR-NeRF significantly outperforms the baseline in these two metrics, thus validating our conjecture and confirming the effectiveness of our model. Altogether, our model not only yields high-quality geometry but also learns realistic LiDAR patterns.

4.3. Scene Editing

As our LiDAR-NeRF can effectively synthesize novel LiDAR views at both scene-level and object-level, it can be exploited to achieve scene editing. We provide an example for novel scene arrangements, which corresponds to editing the scene from the KITTI-360 dataset by fusing novel objects from our NeRF-MVL dataset. Given the 6D pose (3D

translation and yaw, pitch, and roll rotations) of the new object, we first render the corresponding novel view of the object, and then paste it to the desired position in the scene. Screening out points that meet the occlusion as Sec. 3.2, we obtain the final novel scene arrangement. We illustrate this is in Fig. 7 and provide more visualizations in the video in the supplementary material.

4.4. Limitations and Future work

As this paper is the first attempt at novel LiDAR view synthesis, there remains room for improvement. Our LiDAR-NeRF draws inspiration from the original NeRF formalism. As such, it is better suited to static scenes, and requires training for several hours on a large-scale scene. Fortunately, much progress is being made in handling dynamic scenes [25, 23, 10] and reducing training time [20, 36, 3] in image-based NeRF, and we expect these advances to facilitate the development of counterparts for novel LiDAR view synthesis. Note that, in this work, we have considered the problem of synthesizing LiDAR data only, but jointly rendering LiDAR and images is a natural step forward. We therefore plan to extend our NeRF-MVL dataset to a multimodal one. Altogether, we hope that our work will inspire other researchers to contribute to the development of novel LiDAR view synthesis.

³We utilize the implementation of PyTorchEMD.

Table 3: **Novel LiDAR view synthesis on NeRF-MVL**. Our LiDAR-NeRF outperforms the baseline in almost all metrics. The Chamfer Distance (C-D) in which it doesn't are biased by the density of the point clouds generated by the baseline. See Fig. 6 for a visual example.

Object-Category	N-P ↓	EM-D ↓	C-D↓	F-score↑	RMSE↓	$\delta 1\uparrow$	$\delta 2\uparrow$	$\delta 3\uparrow$	PSNR↑	SSIM↑
Baseline Renderer										
Pier	7.095	0.194	0.015	97.17	9.364	59.00	59.00	59.00	31.68	0.360
Traffic-cone	7.100	0.213	0.014	98.22	10.28	64.74	64.74	64.74	11.76	0.307
Tire	4.960	0.300	0.024	95.18	8.851	62.25	62.25	62.25	45.96	0.876
Warning-sign	7.282	0.420	0.020	95.94	9.640	61.52	61.52	61.52	35.54	0.549
Bollard	12.62	0.377	0.021	96.71	9.928	57.43	57.43	57.43	29.40	0.217
Plant	4.969	0.136	0.014	97.27	8.285	67.90	67.90	67.90	36.98	0.653
Car	1.561	0.792	0.019	96.67	7.842	71.29	71.29	71.29	25.53	0.815
Pedestrian	1.568	0.251	0.008	99.67	7.133	78.64	78.64	78.64	26.25	0.485
Safety-barriers	3.528	0.324	0.011	99.30	8.791	62.51	62.51	62.51	23.53	0.385
Average	5.632	0.334	0.016	97.35	8.902	65.03	65.03	65.03	29.63	0.516
LiDAR-NeRF										
Pier	0.414	0.013	0.014	96.68	3.701	93.52	93.52	93.52	39.39	0.787
Traffic-cone	0.391	0.026	0.026	92.72	4.747	92.66	92.66	92.66	15.95	0.456
Tire	0.264	0.043	0.039	89.52	3.640	93.27	93.27	93.27	53.85	0.985
Warning-sign	0.259	0.037	0.033	90.87	4.296	92.62	92.62	92.62	47.18	0.930
Bollard	0.498	0.021	0.028	90.48	3.805	93.56	93.56	93.56	38.64	0.701
Plant	0.251	0.018	0.019	96.30	3.349	95.02	95.02	95.02	46.18	0.941
Car	0.116	0.124	0.033	92.77	4.044	93.14	93.14	93.14	32.10	0.916
Pedestrian	0.102	0.033	0.018	96.33	3.299	95.70	95.70	95.70	31.77	0.751
Safety-barriers	0.281	0.068	0.045	89.59	3.693	93.87	93.87	93.87	30.64	0.738
Average	0.286	0.043	0.028	92.81	3.864	93.65	93.65	93.65	37.23	0.801

5. Conclusion

We have introduced the new task of novel LiDAR view synthesis and proposed the first differentiable LiDAR renderer. Our proposed method, LiDAR-NeRF, jointly learns the geometry and attributes of 3D points, resulting in more accurate and realistic LiDAR patterns. We further established the NeRF-MVL dataset, containing 9 objects over 360-degree LiDAR viewpoints acquired with multiple sensors. Our experiments on both scene-level and our object-level data have evidenced the superiority of our approach over model-based simulators. More importantly, our approach is simple and does not rely on explicit 3D reconstruction and rendering engine. We hope that our work can shed light on realistic LiDAR simulation and motivate future research in this field.

References

- [1] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. [12](#)
- [2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liang, Qiang Xu, Anush Krishnan, Yu Pan, Giacomo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. [2, 5](#)
- [3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorrf: Tensorial radiance fields. In *Computer Vision-ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350. Springer, 2022. [2, 7](#)
- [4] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022. [2](#)
- [5] Jean-Emmanuel Deschaud. Kitti-carla: a kitti-like dataset generated by carla simulator. *arXiv preprint arXiv:2109.00892*, 2021. [3](#)
- [6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. [3](#)
- [7] 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. [6](#)
- [8] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular

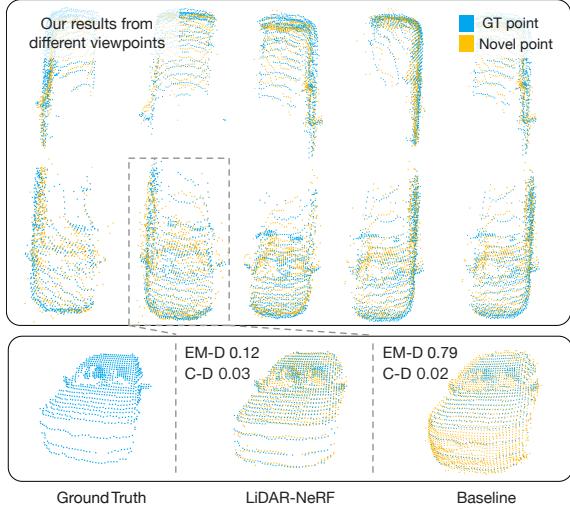


Figure 6: Qualitative results on our NeRF-MVL dataset. **(Top)** The high quality of the results from different viewpoints shown in the top block demonstrates the effectiveness of our method. **(Bottom)** Our LiDAR-NeRF produces realistic patterns with much smaller EM-D, a similarity metric of two point clouds, while the baseline yields overly dense and smooth point clouds. This shows that C-D can be biased towards dense predictions.

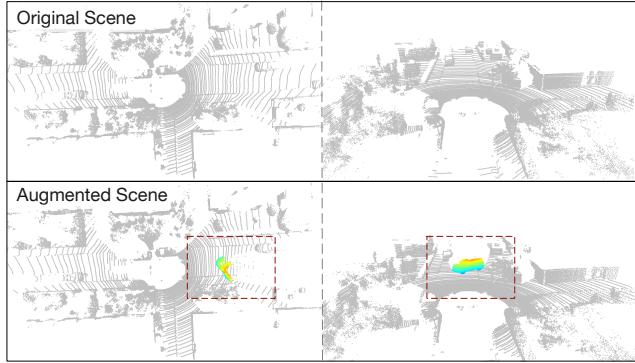


Figure 7: Scene editing. We augment an original scene (top) by placing an object at an arbitrary point and in an arbitrary pose. Note that the augmented scene (bottom) has realistic occlusion effects thanks to our pseudo range-image formalism. Best viewed in color and zoomed in. See the supplementary material for more visual examples.

- depth estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3828–3838, 2019. 6
- [9] Benoît Guillard, Sai Vemprala, Jayesh K Gupta, Ondrej Miksik, Vibhav Vineet, Pascal Fua, and Ashish Kapoor. Learning to simulate realistic lidars. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8173–8180. IEEE, 2022. 4
- [10] Hankyu Jang and Daeyoung Kim. D-tensorf: Tenso-

- rial radiance fields for dynamic scenes. *arXiv preprint arXiv:2212.02375*, 2022. 7
- [11] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12912–12921, 2022. 2
- [12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6, 11
- [13] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 2
- [14] Ferdinand Langer, Andres Milioto, Alexandre Haag, Jens Behley, and Cyrill Stachniss. Domain transfer for semantic segmentation of lidar data using deep neural networks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8263–8270. IEEE, 2020. 11
- [15] Chenqi Li, Yuan Ren, and Bingbing Liu. Pcgen: Point cloud generator for lidar simulation. *arXiv preprint arXiv:2210.08738*, 2022. 2, 3, 11
- [16] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2, 5, 6
- [17] Lingjie Liu, Jitao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 2
- [18] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11167–11176, 2020. 2, 3, 11
- [19] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2, 3, 4
- [20] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2, 7
- [21] 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*, pages 45–59. Wiley Online Library, 2021. 2, 5
- [22] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5480–5490, 2022. 2
- [23] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knott, and

- Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2856–2865, 2021. 2, 7
- [24] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 143–152, 2017. 4
- [25] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 7
- [26] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12932–12942, 2022. 2
- [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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12892–12901, 2022. 2, 5
- [28] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover’s distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99, 2000. 7
- [29] Andrew Sanders. *An introduction to Unreal engine 4*. AK Peters/CRC Press, 2016. 3
- [30] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 2, 5
- [31] Matthew Tancik, Vincent Casser, Xinchen 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. 2
- [32] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6
- [33] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1887–1893. IEEE, 2018. 3
- [34] Aoran Xiao, Jiaxing Huang, Dayan Guan, Fangneng Zhan, and Shijian Lu. Transfer learning from synthetic to real lidar point cloud for semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2795–2803, 2022. 3
- [35] Ziyang Xie, Junge Zhang, Wenye Li, Feihu Zhang, and Li Zhang. S-nerf: Neural radiance fields for street views. *arXiv preprint arXiv:2303.00749*, 2023. 2, 12
- [36] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131*, 2021. 2, 7
- [37] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 2
- [38] Kaicheng Yu, Tang Tao, Hongwei Xie, Zhiwei Lin, Zhongwei Wu, Zhongyu Xia, Tingting Liang, Haiyang Sun, Jiong Deng, Dayang Hao, et al. Benchmarking the robustness of lidar-camera fusion for 3d object detection. *arXiv preprint arXiv:2205.14951*, 2022. 11
- [39] Xiangyu Yue, Bichen Wu, Sanjit A Seshia, Kurt Keutzer, and Alberto L Sangiovanni-Vincentelli. A lidar point cloud generator: from a virtual world to autonomous driving. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 458–464, 2018. 3

A. Additional Details

A.1. NeRF-MVL details.

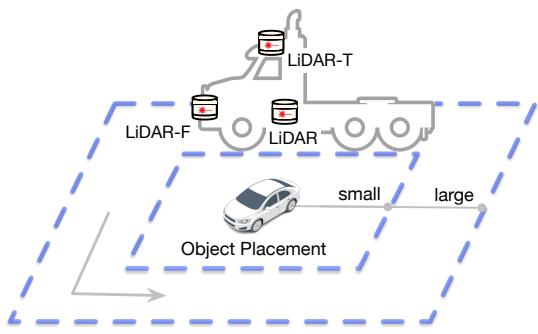


Figure 8: **Drive planning and sensor placements.**

We provide an illustration of the collection drive planning and sensor placements of our NeRF-MVL dataset in Fig. 8. Concretely, we design two square paths of collection, small and large with 7 and 15 meters in length respectively. For each side path of a square, we ensure the vehicle moves along the path from the start to the end without turning to ensure the path is collected along the desired path.

A.2. Baseline LiDAR Renderer

As generating novel LiDAR views remains unexplored, we moderately adapt existing model-based LiDAR simulators to form a baseline. Given a sequence of LiDAR frames, the objective is to generate novel LiDAR scenes with realistic geometry. In essence, our baseline, summarized in Fig. 9, follows the physics-based, multi-step approach of existing simulation pipelines [18, 15, 14]. In short, we first gather a set of LiDAR frames, which are transformed to the global world coordinate system. The resulting aligned dense 3D scene is projected to the novel view via ray-casting. Finally, the problem of LiDAR ray-dropping is simulated to improve realism. Below, we discuss ray-casting and ray-dropping in more detail.

Ray-casting. Ray-casting defines the intersections between the laser beams and the dense point cloud. The standard approach is closest-point ray-casting [14], which projects the dense point cloud to a range image using Eq. (5), and, for each ray, selects the point with the smallest measured distance, as is commonly done in rendering pipelines with the so-called z-buffer. In real-world scenarios, however, the calibration, sensor synchronization, and other properties [38] are affected by different noise sources, and the points in the dense point cloud do not strictly lie on the scene surface. Thus, considering only the closest point tends to render noisy ray-casted point clouds. We therefore take more points in the z-buffer into account, which is similar to the

FPA ray-casting in PCGen [15], averaging the points within a certain threshold near the closest point, and weighting them by their inverse distance.

Ray-drop network. Ray-casting yields a nearly perfect point cloud in the novel view. However, the laser returns of a real LiDAR sensor are affected by many factors, such as the distance to the scene, the incidence angle, and the material texture and reflectivity. To improve realism, we follow from PCGen [15] to employ a small surrogate MLP to learn the ray-drop, we dub ray-drop MLP. Specifically, we model a ray’s return probability as

$$p = \text{MLP}(\theta', d, i), \quad (11)$$

where θ' , d , and i are the viewing direction in the local coordinate system, the distance and the intensity of the ray.

Training details. For raydrop MLP, it has 4 layers with width 128. And it also utilize the sample positional encoding as NeRFs. The highest frequency for the viewing direction (in local coordinate system), distance and intensity are set to 2^{10} , 2^4 and 2^4 , respectively. It is trained with Adam [12] with a learning rate of 5e-3, and supervised with MSE loss for 80k iterations with a batch size of 2048.

Differences between the baseline and LiDAR-NeRF. The baseline renderer is in fact a LiDAR simulator, which mimics the physical LiDAR model through explicit 3D reconstruction and ray-tracing via traditional renderer. Consequently, the baseline renderer needs an additional style-transfer network to enhance realism to reduce the sim2real gap. The multi-step simulation pipeline restricts its scale and controllability. In contrast, our LiDAR-NeRF benefits from the implicit representation of the real 3D world. We first replace the traditional render engine with a novel differentiable LiDAR renderer. We then leverage the implicit fields to represent a 3D scene and achieve novel LiDAR view synthesis. In addition, our LiDAR-NeRF can encode other features, such as the intensity, ray-drop and other attributes in the same radiance fields, resulting in not only superior performance but also a much simpler pipeline in an end-to-end manner.

B. Additional Experimental Results

B.1. Ablating ray-drop MLP.

In Tab. 4, we investigate the impact of the ray-drop network. For the baseline render, the baseline renderer achieves comparable performance with the average z-buffer ray-casting and ray-drop MLP. Moreover, employing the ray-drop MLP into LiDAR-NeRF can boost the performance, e.g., the PSNR from 15.83 to 18.45. These results demonstrate the effectiveness of ray-drop MLP. For the quality of final rendered point clouds in settings with or without the ray-drop network, our method constantly out-

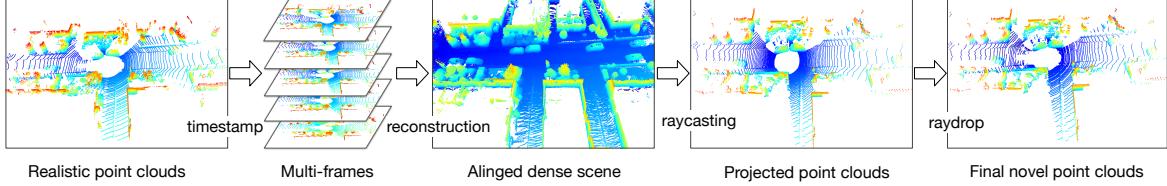


Figure 9: **Baseline multi-step LiDAR renderer pipeline.**

performs the baseline renderer by a clear margin. This further evidences the effectiveness of our approach.

Training the ray-drop network necessitates considerable effort, entailing the initial rendering of training sets, followed by the construction of paired training sets for the ray-drop MLP, comprising the rendered outcomes and their corresponding ground truth. Subsequently, it is imperative to meticulously calibrate the model’s architecture and training parameters to achieve superior results for each scene or object. Owing to resource limitations, the ray-drop network is not incorporated in other experiments presented within this paper.

B.2. More results on KITTI-360 dataset.

We report results on additional sequences on KITTI-360 dataset in Tab. 5. Our LiDAR-NeRF consistently outperforms the baseline over all sequences in all metrics. To further compare the point distribution between ground-truth and rendered point clouds, we provide results of the N-P and EM-D metrics here. However, current implementation of the EM-D metric relies on operations in CPU, which is computationally expensive and can not scale well to the large point clouds. We divide the point clouds into two groups given a condition of $x > 0$ and $x \leq 0$, compute the EM-D of each group and report the mean value. We observe that, on average, our LiDAR-NeRF reduces the EM-D from 18.62 to 6.32, which is around 66% smaller, evidencing the superior quality of our generated point clouds.

B.3. Ablations.

We ablate the training strategies in Tab. 6. In training process, we scale (Scale) the scene frames with a factor such that the region of interest falls within a unit cube. There are also parameterization function (Contract) used in [1, 35] as:

$$\text{contract}(x) = \begin{cases} x/r, & \text{if } \|x\| \leq r, \\ (1 + b - \frac{br}{\|x\|}) \frac{x}{\|x\|}, & \text{otherwise.} \end{cases} \quad (12)$$

Where r and b are the radius parameters to decide the mapping boundary. We set $r = 10$ and $b = 1$ for the scene point clouds.

As expected, without parameterization function, the model hardly learned anything. While the ‘Scale’ and ‘Contract’ functions achieve comparable results. Moreover, we

employ a cosine schedule with one thousand iterations to warm-up for the learning rate (Cos-lr) to stable training, which slightly improves the performance.

C. Additional qualitative visualization.

Scene editing. As shown in Fig. 10, our LiDAR-NeRF can render the corresponding novel view and the yield augmented scene has realistic occlusion effects, compared with the common cope-paste strategy.

Qualitative results on KITTI-360. We provide more qualitative results on KITTI-360 dataset in Fig. 11, which shows that our LiDAR-NeRF produces high quality point clouds similar to the ground truth.

Video demo. In addition to the figures, we have attached a video demo in the supplementary materials, which consists of hundreds of frames that provide a more comprehensive evaluation of our proposed approach.

Table 4: **Novel LiDAR view synthesis on KITTI-360 dataset.** LiDAR-NeRF consistently outperforms the baseline w or w/o ray-drop MLP.

Method	Component	C-D↓	F-score↑	RMSE↓	$\delta 1\uparrow$	$\delta 2\uparrow$	$\delta 3\uparrow$	PSNR↑	SSIM↑
Base Renderer	Closest point	0.259	83.87	4.312	65.06	67.86	69.09	13.17	0.062
	+ Average z-buffer	0.248	85.43	4.332	65.21	67.93	69.13	14.08	0.087
LiDAR-NeRF	–	0.126	87.64	3.948	78.26	79.57	80.09	15.83	0.202
Base Renderer	+ Raydrop MLP	0.213	87.96	3.538	87.06	89.45	90.47	16.82	0.472
LiDAR-NeRF	+ Raydrop MLP	0.129	87.16	3.292	89.32	90.66	91.19	18.45	0.524

Table 5: **More results of novel LiDAR view synthesis on KITTI-360.** LiDAR-NeRF outperforms the baseline over all sequences in all metrics. * donates that we calculate the EM-D of two split groups and report the mean value.

Sequence	N-P ↓	EM-D* ↓	C-D↓	F-score↑	RMSE↓	$\delta 1\uparrow$	$\delta 2\uparrow$	$\delta 3\uparrow$	PSNR↑	SSIM↑
Baseline Renderer										
Seq-1538-1601	0.372	16.31	0.193	86.22	4.404	64.78	68.81	71.29	13.69	0.083
Seq-1728-1791	0.411	22.06	0.227	81.94	4.843	63.68	67.05	69.17	14.05	0.097
Seq-1908-1971	0.412	19.08	0.248	85.43	4.332	65.21	67.93	69.13	14.08	0.087
Seq-3353-3416	0.403	17.02	0.164	86.70	4.744	66.76	68.40	69.66	13.98	0.103
Average	0.400	18.62	0.208	85.07	4.579	65.11	68.05	69.81	13.95	0.093
LiDAR-NeRF										
Seq-1538-1601	0.036	5.126	0.148	84.88	4.007	76.66	79.09	79.85	15.59	0.187
Seq-1728-1791	0.041	7.186	0.148	83.88	4.207	78.44	80.35	81.09	16.06	0.224
Seq-1908-1971	0.023	6.117	0.126	87.64	3.948	78.26	79.57	80.09	15.83	0.202
Seq-3353-3416	0.021	6.865	0.150	87.31	4.039	79.15	80.14	80.63	15.96	0.249
Average	0.030	6.324	0.143	85.93	4.050	78.13	79.79	80.42	15.86	0.216

Table 6: **Ablation of LiDAR-NeRF training strategies.**

Scale	Contract	Cos-Ir	C-D↓	F-score↑	RMSE↓	$\delta 1\uparrow$	$\delta 2\uparrow$	$\delta 3\uparrow$	PSNR↑	SSIM↑
			132.4	00.19	9.910	00.07	00.62	2.087	9.472	0.037
			0.129	87.08	3.849	79.63	81.09	81.64	16.13	0.226
			0.146	84.07	3.978	77.81	79.33	79.81	15.78	0.193
			0.126	87.64	3.948	78.26	79.57	80.09	15.83	0.202

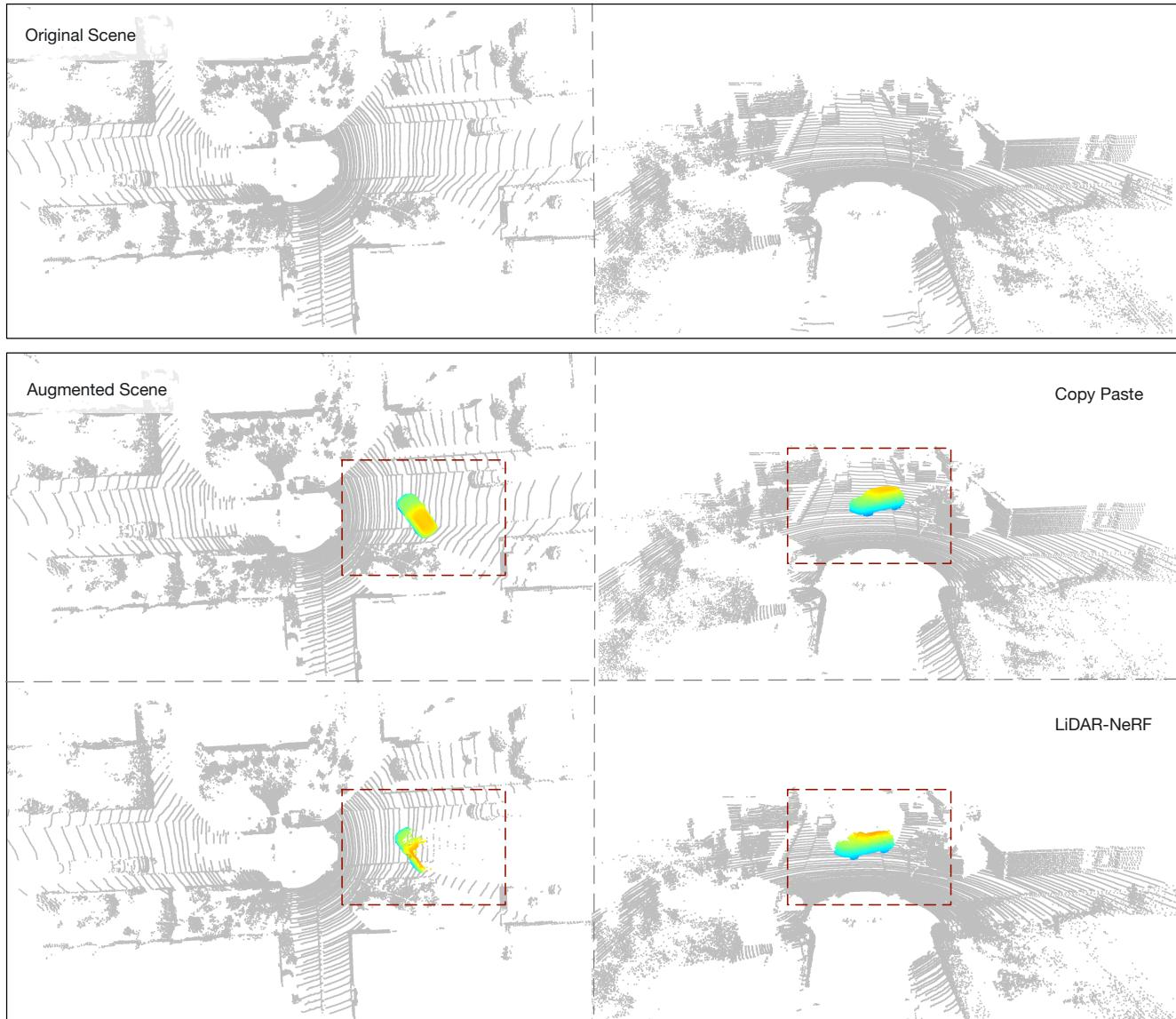


Figure 10: **Scene editing.** The augmented scene from our LiDAR-NeRF has realistic occlusion effects thanks to our pseudo range-image formalism, compared with the common copy-paste strategy.

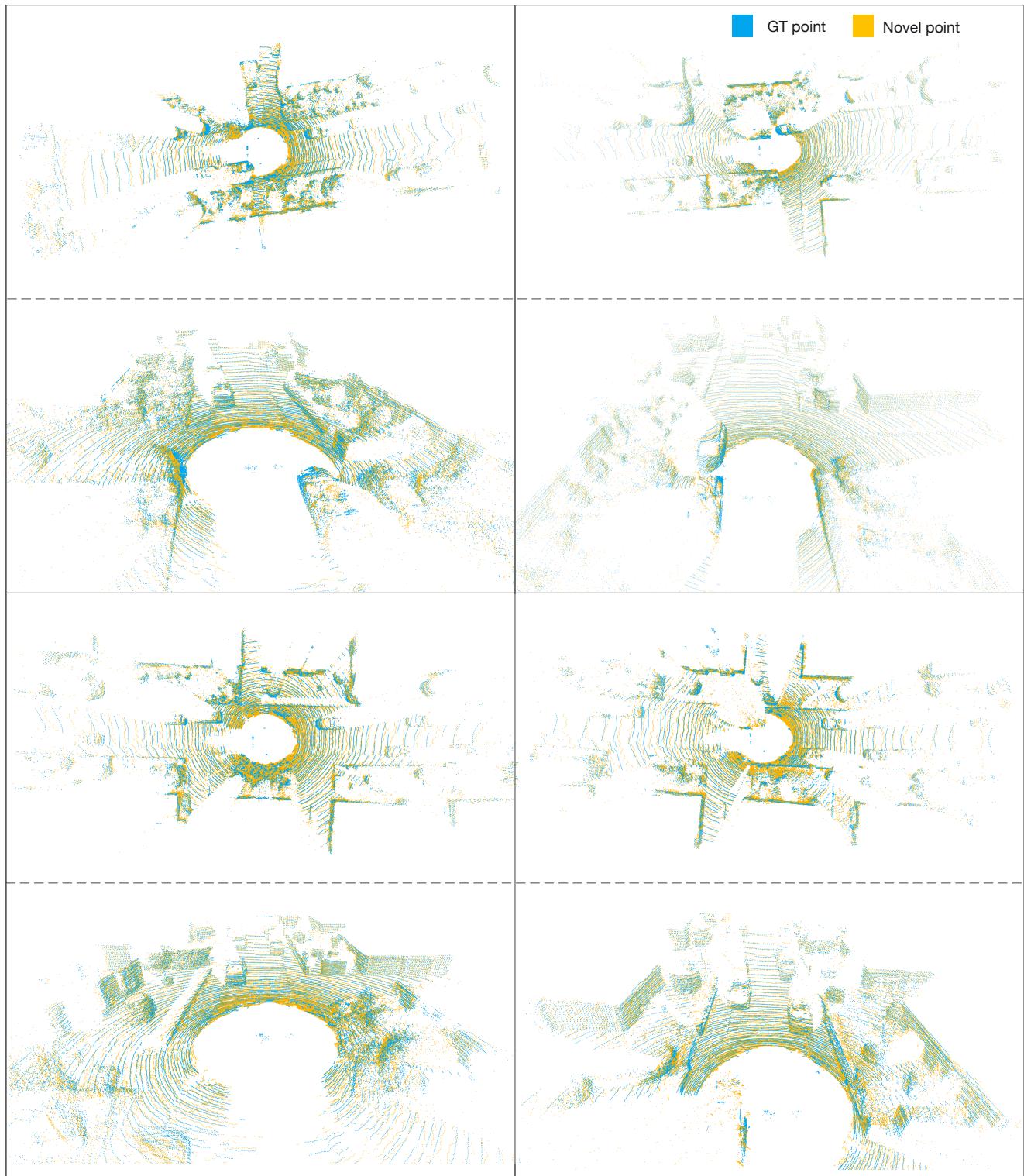


Figure 11: **Qualitative results on KITTI-360.** The high quality of the results from different view-points demonstrates the effectiveness of our method.