

# Supplementary Materials: Novel LiDAR View Synthesis via Neural Radiance Fields

Anonymous Authors

## A 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 per-scene optimization. Fortunately, much progress is being made in handling dynamic scenes [2, 8, 9] and generalization [10, 12] 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. Moreover, we are integrating various LiDAR renderers, including LiDAR simulators and our proposed LiDAR-NeRF framework, with the goal of developing a unified Codebase for novel LiDAR view synthesis that can benefit the community. Altogether, we hope that our work will inspire other researchers to contribute to the development of novel LiDAR view synthesis.

## B ADDITIONAL DETAILS

### B.1 Ablation details

All ablation experiments were conducted on the seq-1908-1971 of the KITTI-360 dataset, which is a large, clear scene with numerous objects, making it an ideal sequence for comparison.

### B.2 Baseline details

As generating novel LiDAR views remains unexplored, we moderately adapt existing model-based LiDAR simulators as our baseline. Given a sequence of LiDAR frames, the objective is to generate novel LiDAR scenes with realistic geometry. In essence, the baseline, summarized in Fig. 1, follows the physics-based, multi-step approach of existing simulation pipelines [6, 7]. In short, they first gather a set of LiDAR frames, which are transformed into the global world coordinate system. The resulting aligned dense 3D scene or mesh is projected to the novel view via ray-casting. Finally, the problem of LiDAR ray-dropping is simulated to improve realism. Because the official codes of LiDARsim [7] and PCGen [6] are not publicly available, we re-implemented them following the papers as closely as possible. Below, we discuss the implementation in more detail.

**B.2.1 LiDARsim. Point cloud meshing and mesh-ray intersection.** Following LiDARsim [7], our pipeline also consists of two steps: point cloud meshing and mesh-ray intersection. The point cloud meshing step is implemented based on the Poisson surface reconstruction algorithm [3], which is a modification from the original author’s Surfel-based meshing. The mesh-ray intersection step is implemented based on Intel Embree using the Open3D library [14]. The performance of Poisson surface reconstruction is sensitive to hyperparameters, namely the maximum depth of the

tree and the weight threshold for selecting mesh vertices. For each dataset, we perform grid search of these two parameters on the training set and select the best-performing one on the test set.

**Ray-drop network.** To train the ray-drop network, following LiDARsim [7], we concatenate range, intensity, incidence angle (in cosine), and the normal of surface hit as inputs. We then train a U-Net with 4 down-sampling (with 64, 128, 256, and 512 channels respectively) and 4 up-sampling layers (with 1024, 512, 256, and 128 channels respectively) to predict the ray-drop mask. We train on the training frames for 10 epochs. The final range image prediction is the ray-casted results multiplied by the ray-drop mask. We use this final range image for evaluation.

**B.2.2 PCGen. Ray-casting.** Ray-casting defines the intersections between the laser beams and the dense point cloud. The standard approach is closest-point (CP) ray-casting [5], which projects the dense point cloud to a range image, 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 [13] 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 and more points in the z-buffer need to be taken into account. PCGen [6] propose the first peak averaging (FPA) raycasting, 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 [6] to employ a small surrogate MLP to learn the ray-drop, which we dub ray-drop MLP. Specifically, we model a ray’s return probability as  $p = MLP(\theta', d, i)$ , where  $\theta'$ ,  $d$ , and  $i$  are the viewing direction in the local coordinate system, the distance, and the intensity of the ray. For the implementation of ray-drop MLP, it has 4 layers with a width of 128. It is trained with Adam [4] with a learning rate of  $5e-3$ , and supervised with MSE loss for 10k iterations with a batch size of 2048.

## C ADDITIONAL EXPERIMENTAL RESULTS

### C.1 More ablations of LiDAR-NeRF.

**LiDAR-NeRF (w NeRF).** We ablate the training strategies of our LiDAR-NeRF (w NeRF) in Table 1. In the training process, we scale (Scale) the scene frames with a factor such that the region of interest falls within a unit cube. There is also a parameterization function (contract) used in [1, 11] as:

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

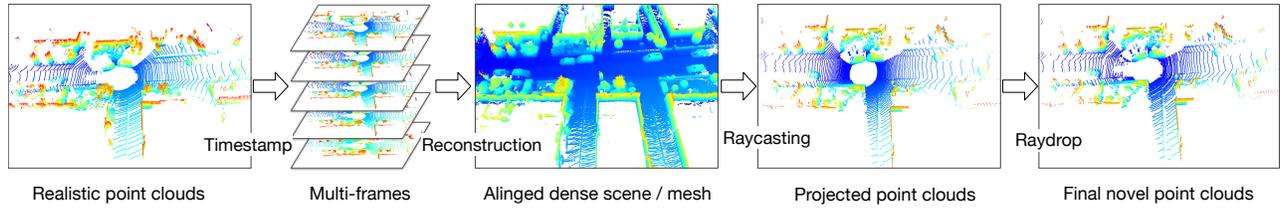


Figure 1: Baseline multi-step LiDAR simulator pipeline.

Table 1: Ablation of LiDAR-NeRF (w NeRF) training strategies.

Scale	Contract	Cos-lr	C-D↓	F-score↑	RMSE↓	$\delta_1$ ↑	$\delta_2$ ↑	$\delta_3$ ↑	SSIM↑	MAE↓
✗	✗	✓	132.4	00.19	9.910	00.07	00.62	2.087	0.296	0.491
✗	✓	✓	0.129	87.08	3.849	79.63	81.09	81.64	0.575	0.231
✓	✗	✗	0.146	84.07	3.978	77.81	79.33	79.81	0.549	0.225
✓	✗	✓	0.126	87.64	3.948	78.26	79.57	80.09	0.555	0.226

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 the 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 the learning rate (Cos-lr) to stable training, which slightly improves the performance.

## C.2 More ablations of baseline.

For exhaustive evaluation and fair comparisons, we also validate different settings of the baseline methods as follows. In Table 2, we investigate the different components of baseline simulators.

**LiDARsim.** For LiDARsim [7], the ray-drop U-Net can both boost the performance and demonstrate its effectiveness.

**PCGen.** For PCGen [6], the FPA z-buffer ray-casting and ray-drop MLP can both boost the performance and demonstrate their effectiveness.

It is worth noting that 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 network, comprising the rendered outcomes and their corresponding ground truth. Subsequently, it is imperative to meticulously adjust the model's architecture and training parameters to achieve superior results for each scene or object.

## C.3 More results.

**KITTI-360 dataset.** We report detailed results on the four sequences of the KITTI-360 dataset in Table 3. Our LiDAR-NeRF consistently outperforms the baseline over all sequences in all metrics.

**NeRF-MVL dataset.** We report detailed results on nine object categories of NeRF-MVL dataset in Table 4. Our LiDAR-NeRF consistently outperforms the baseline over all categories in all metrics.

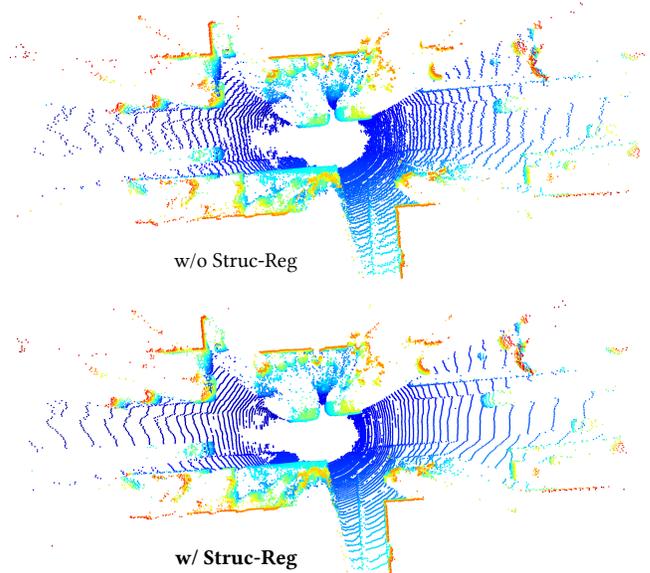


Figure 2: Our structural regularization significantly improves the geometry estimation and produces more realistic LiDAR patterns.

## D ADDITIONAL QUALITATIVE VISUALIZATION.

**Structural regularization.** Our structural regularization significantly improves the geometry estimation and produces more realistic LiDAR patterns as shown in Fig. 2

**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

**Table 2: Ablation of baseline simulators.**

Method	Component	C-D↓	F-score↑	RMSE↓	$\delta 1$ ↑	$\delta 2$ ↑	$\delta 3$ ↑	SSIM↑	MAE↓
LiDARsim [7]	Mesh ray-casting	1.026	64.62	5.674	52.09	57.90	60.15	0.592	0.156
	+ Ray-drop U-Net	0.981	66.77	5.404	67.33	72.70	74.76	0.711	0.122
PCGen [6]	CP ray-casting	0.259	83.87	4.312	65.06	67.86	69.09	0.153	0.274
	+ FPA ray-casting	0.248	85.43	4.332	65.21	67.93	69.13	0.155	0.280
	+ Ray-drop MLP	0.220	87.75	4.029	77.35	79.95	81.05	0.220	0.238

**Table 3: More results of novel LiDAR view synthesis on KITTI-360. LiDAR-NeRF outperforms the baseline over all sequences in all metrics.**

Sequence	C-D↓	F-score↑	RMSE↓	$\delta 1$ ↑	$\delta 2$ ↑	$\delta 3$ ↑	SSIM↑	MAE↓
<b>LiDARsim [7]</b>								
Seq 1538–1601	0.794	72.13	5.455	67.49	73.68	78.17	0.719	0.125
Seq 1728–1791	0.819	65.38	6.083	67.20	70.86	74.11	0.694	0.122
Seq 1908–1971	0.981	66.77	5.404	67.33	72.70	74.76	0.711	0.122
Seq 3353–3416	1.211	63.29	6.038	63.35	67.20	70.64	0.660	0.133
Average	0.951	66.89	5.745	66.34	71.11	74.42	0.696	0.126
<b>PCGen [6]</b>								
Seq 1538–1601	0.159	88.64	4.091	75.64	79.62	82.08	0.558	0.254
Seq 1728–1791	0.194	83.87	4.560	76.19	79.37	81.35	0.545	0.243
Seq 1908–1971	0.220	87.75	4.029	77.35	79.95	81.05	0.559	0.238
Seq 3353–3416	0.173	88.37	4.633	78.40	79.92	81.05	0.536	0.244
Average	0.187	87.16	4.328	76.90	79.72	81.38	0.550	0.245
<b>LiDAR-NeRF (w/ NeRF)</b>								
Seq 1538–1601	0.148	84.88	4.007	76.66	79.09	79.85	0.527	0.242
Seq 1728–1791	0.148	83.88	4.207	78.44	80.35	81.09	0.549	0.239
Seq 1908–1971	0.126	87.64	3.948	78.26	79.57	80.09	0.555	0.226
Seq 3353–3416	0.150	87.31	4.039	79.15	80.14	80.63	0.550	0.233
Average	0.143	85.93	4.050	78.13	79.79	80.42	0.545	0.235
<b>LiDAR-NeRF (w/ iNGP and StructReg)</b>								
Seq 1538–1601	0.073	92.55	3.530	80.76	82.71	83.52	0.597	0.102
Seq 1728–1791	0.088	90.95	3.766	82.91	84.26	84.91	0.646	0.091
Seq 1908–1971	0.077	92.98	3.511	82.25	83.28	83.73	0.635	0.096
Seq 3353–3416	0.086	93.46	3.654	82.78	83.36	83.71	0.625	0.094
Average	0.081	92.49	3.615	82.18	83.40	83.97	0.626	0.096

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. Furthermore, it is worth mentioning that our method has the capability to adjust the intrinsics of LiDAR, thereby addressing the issue of inconsistent LiDAR patterns resulting from the use of different LiDAR devices in the NeRF-MVL and KITTI-360 datasets. As illustrated in Fig. 3, our LiDAR-NeRF can render the corresponding novel view, and the yield augmented scene has realistic occlusion effects and a consistent LiDAR pattern, compared with the common

cope-paste strategy. We provide more visualizations in the video in the supplementary material.

**Qualitative results on KITTI-360.** We provide more qualitative results on KITTI-360 dataset in Fig. 4, which shows that our LiDAR-NeRF produces high-quality point clouds fidelity with 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 NeRF-MVL. Our LiDAR-NeRF outperforms the baseline in all metrics.

Object-Category	C-D↓	F-score↑	RMSE↓	$\delta_1$ ↑	$\delta_2$ ↑	$\delta_3$ ↑	SSIM↑	MAE↓
<b><i>LiDARsim [7]</i></b>								
Bollard	0.011	98.92	5.751	84.64	84.64	84.64	0.518	2.172
Car	0.067	94.06	6.331	82.35	82.37	82.37	0.645	1.426
Pedestrian	0.011	97.91	4.788	90.20	90.20	90.20	0.748	4.353
Pier	0.016	95.33	6.606	78.67	78.67	78.67	0.510	2.541
Plant	0.018	93.81	5.315	86.45	86.45	86.45	0.667	0.852
Safety barrier	0.015	97.95	6.697	78.29	78.29	78.29	0.565	5.801
Tire	0.021	95.45	6.936	75.53	75.53	75.53	0.566	0.457
Traffic cone	0.013	99.59	5.861	88.14	88.14	88.14	0.628	19.14
Warning sign	0.024	91.05	5.569	86.57	86.57	86.57	0.663	0.546
Average	0.022	96.01	5.984	83.43	83.43	83.43	0.612	4.143
<b><i>PCGen [6]</i></b>								
Bollard	0.021	96.71	9.928	57.43	57.43	57.43	0.082	4.046
Car	0.446	72.23	5.829	86.37	86.37	86.37	0.331	0.123
Pedestrian	0.063	92.43	4.364	92.61	92.61	92.61	0.472	6.073
Pier	0.017	96.92	9.534	58.11	58.11	58.11	0.088	4.557
Plant	0.021	91.33	6.729	76.26	76.26	76.26	0.280	1.475
Safety barrier	0.070	74.88	5.382	87.33	87.33	87.33	0.207	5.689
Tire	0.026	94.84	8.748	62.58	62.58	62.58	0.095	0.714
Traffic cone	0.014	98.52	8.698	71.80	71.80	71.80	0.223	32.26
Warning sign	0.020	95.77	8.810	65.72	65.72	65.72	0.171	1.472
Average	0.078	90.40	7.558	73.13	73.13	73.13	0.217	6.268
<b><i>LiDAR-NeRF (w/ NeRF)</i></b>								
Bollard	0.028	90.48	3.805	93.56	93.56	93.56	0.245	0.426
Car	0.033	92.77	4.044	93.14	93.14	93.14	0.544	0.457
Pedestrian	0.018	96.33	3.299	95.70	95.70	95.70	0.627	6.454
Pier	0.014	96.68	3.701	93.52	93.52	93.52	0.353	0.837
Plant	0.019	96.30	3.349	95.02	95.02	95.02	0.571	0.472
Safety barrier	0.045	89.59	3.693	93.87	93.87	93.87	0.517	4.070
Tire	0.039	89.52	3.640	93.27	93.27	93.27	0.448	0.100
Traffic cone	0.026	92.72	4.747	92.66	92.66	92.66	0.428	10.88
Warning sign	0.033	90.87	4.296	92.62	92.62	92.62	0.424	0.082
Average	0.028	92.81	3.864	93.65	93.65	93.65	0.462	2.642
<b><i>LiDAR-NeRF (w/ iNGP)</i></b>								
Bollard	0.007	98.54	0.974	99.13	99.13	99.13	0.786	0.723
Car	0.005	99.33	2.256	97.85	97.85	97.85	0.842	0.436
Pedestrian	0.001	99.97	1.381	99.24	99.24	99.24	0.941	1.650
Pier	0.004	98.14	1.047	99.08	99.08	99.08	0.889	0.671
Plant	0.001	99.36	0.415	99.80	99.80	99.80	0.976	0.464
Safety barrier	0.018	92.41	2.624	96.63	96.63	96.63	0.622	2.640
Tire	0.001	100.0	0.563	99.64	99.64	99.64	0.965	0.174
Traffic cone	0.002	100.0	1.221	99.32	99.32	99.32	0.949	2.493
Warning sign	0.006	98.77	1.266	99.13	99.13	99.13	0.948	0.259
Average	0.005	98.50	1.305	98.86	98.86	98.86	0.879	1.057

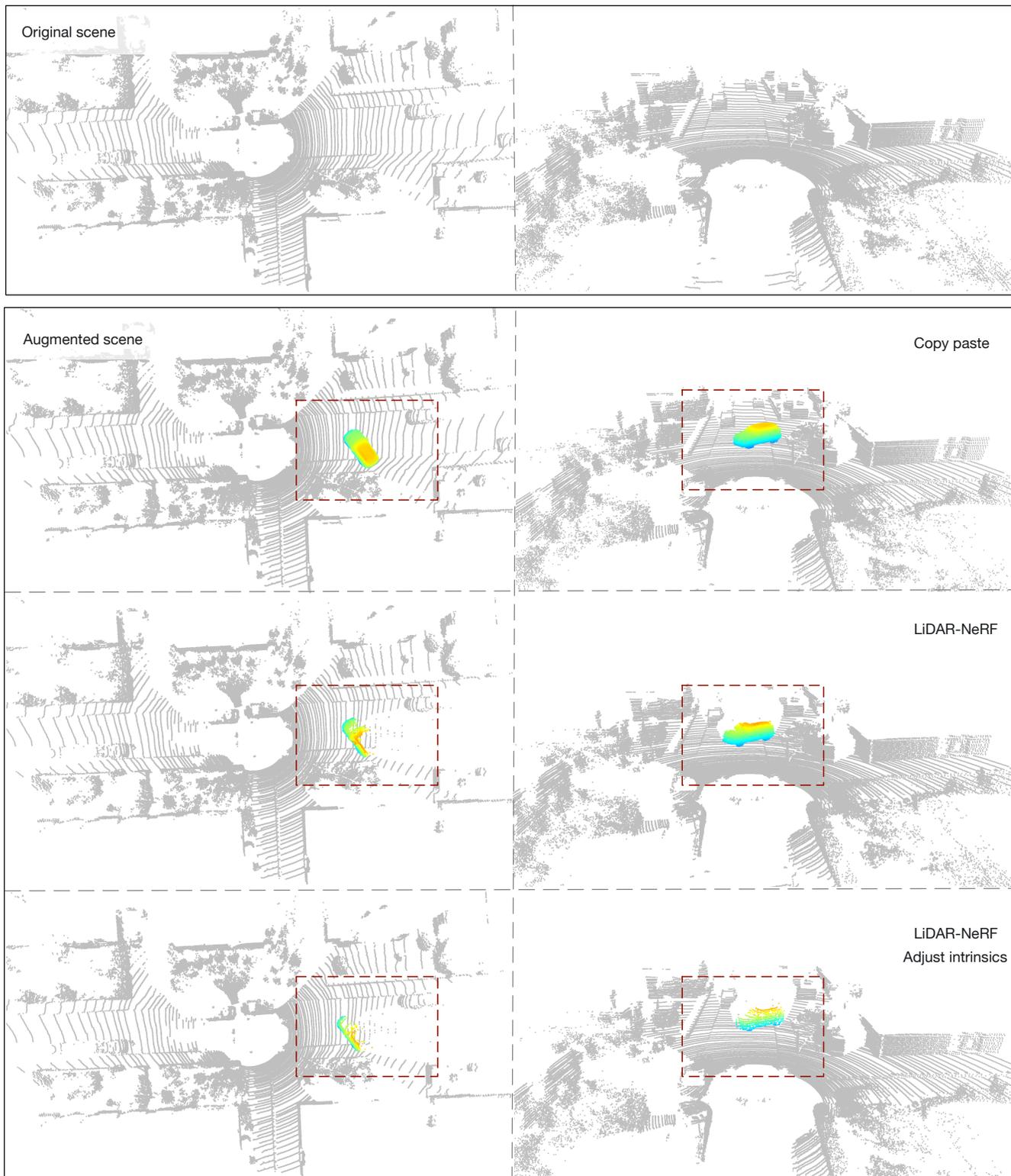


Figure 3: Scene editing. The augmented scene from our LiDAR-NeRF has realistic occlusion effects and consistent LiDAR pattern thanks to our differentiable LiDAR rendering formalism, compared with the common cope-paste strategy.

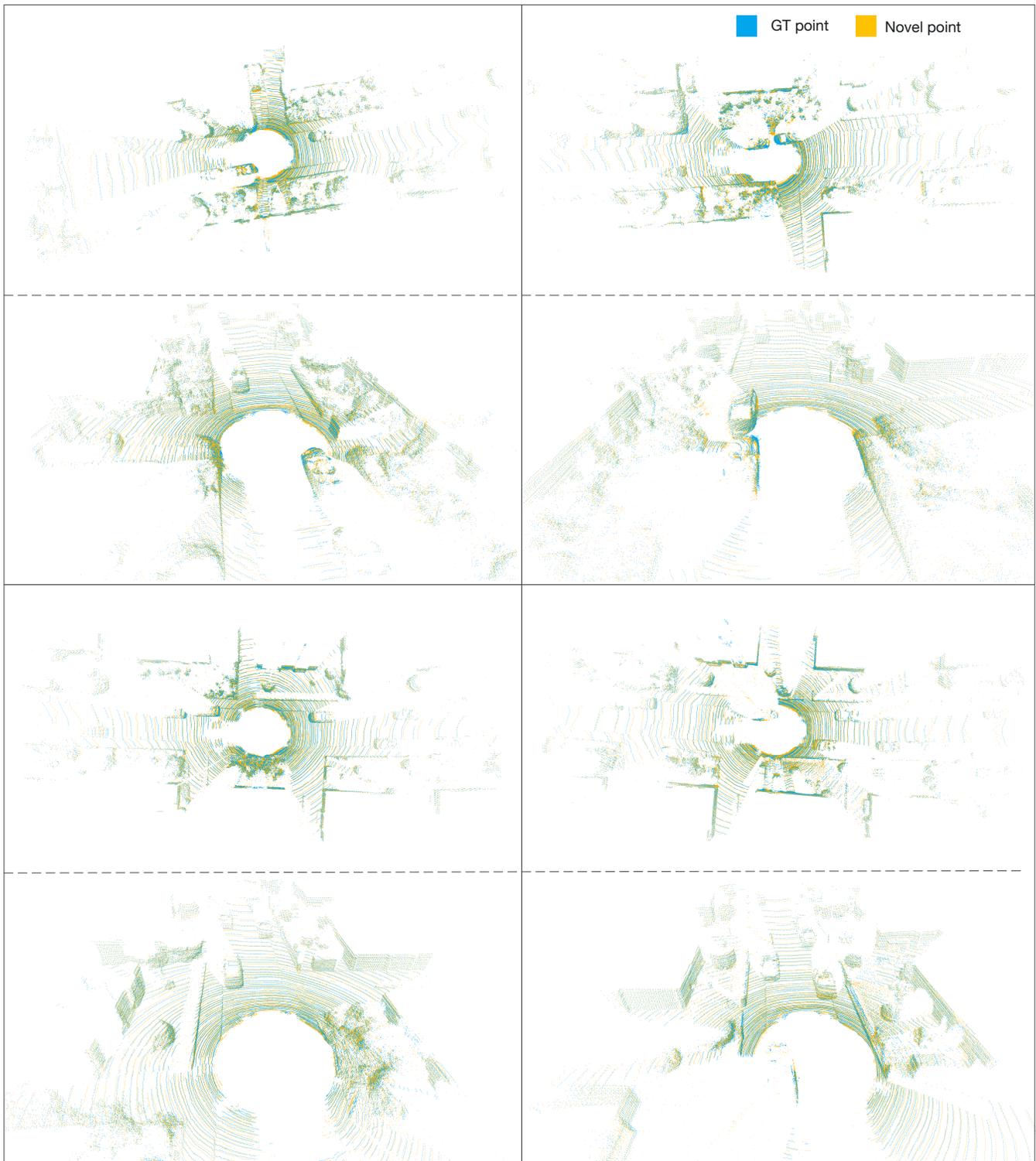


Figure 4: Qualitative results on KITTI-360. The high quality of the results from different view-points demonstrates the effectiveness of our method.

697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754

## 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. 755
- [2] Hankyu Jang and Daeyoung Kim. D-tensorf: Tensorial radiance fields for dynamic scenes. *arXiv preprint arXiv:2212.02375*, 2022. 756
- [3] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, page 0, 2006. 757
- [4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 758
- [5] 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. 759
- [6] Chenqi Li, Yuan Ren, and Bingbing Liu. Pcgcn: Point cloud generator for lidar simulation. *arXiv preprint arXiv:2210.08738*, 2022. 760
- [7] 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. 761
- [8] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, 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. 762
- [9] 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. 763
- [10] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d representation and rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15182–15192, 2021. 764
- [11] 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. 765
- [12] 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. 766
- [13] 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. 767
- [14] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 768

755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812