Lid-Lab-NeRF: Generating Temporally Consistent, Labelled LiDAR Point Clouds using Neural Radiance Fields

Shrestha Srivastava[1]
shrestha20@iiserb.ac.in
Vaibhav Kumar[1,2]
vaibhav@iiserb.ac.in

[1] GeoAI4Cities Lab, Indian Institute of Science Education and Research, Bhopal

[2] SimDAAS Autonomy Pvt. Ltd.

Abstract

LiDAR point clouds are essential for autonomous driving and urban planning, but collecting and labelling high-quality data remains expensive and inflexible due to fixed vehicle trajectories. While synthetic data generation offers a potential solution, existing approaches struggle to bridge the reality gap and often produce temporally inconsistent or unlabelled results. We present Lid-Lab-NeRF, a novel framework that generates labelled, dynamic LiDAR point clouds by combining neural radiance fields with motion field prediction for better time-consistent outputs and semantic labelling through a semantic radiance field. Our model simultaneously produces intensity, depth, labels, and raydrop probability, while a feature alignment loss and raydrop optimisation pipeline ensure realistic scan patterns. Experiments on the KITTI-360 dataset demonstrate that Lid-Lab-NeRF effectively synthesizes labelled point clouds that closely match real LiDAR characteristics compared to existing works. Codes are available at https://github.com/geoai4cities/Lid-Lab-NeRF.git.

1 Introduction

To develop reliable perception and decision-making systems for fields such as spatial planning and autonomous navigation, extensive and well-annotated LiDAR datasets are often required. Capturing real-world LiDAR data that reflects the full spectrum of environmental variations across weather, illumination, and sensor parameters such as trajectory, field of view, and scan pattern remains a significant challenge. Moreover, even when such datasets are acquired, producing high-quality semantic labels involves substantial time, effort, and cost.

To address these challenges, researchers increasingly employ synthetic data generation to simulate diverse scenarios under controlled conditions. This enables systematic evaluation of factors such as trajectory variation, sensor type and extrinsics, label accuracy for perception, and the capture of dynamic objects from multiple viewpoints. Game engine-based approaches [4, 14, 18, 22] are the most prevalent for recreating outdoor environments and collecting data, yet they remain resource-intensive and cannot fully replicate real-world complexity.

To address these constraints, neural rendering techniques are increasingly applied to LiDAR data. Recent approaches such as LiDAR-NeRF [26], NeRF-LiDAR [30], and LiDAR4D [32] have made notable advances, yet most struggle to either generate labelled novel scans or capture dynamic objects within a scene. To overcome these limitations, we propose a NeRF-based framework that generates LiDAR scans from novel viewpoints while accounting for dynamic objects and producing semantic labels. This makes the generated data directly usable for downstream tasks such as object detection and segmentation. The proposed method incorporates the following components to accurately synthesize novel scans:

- Hybrid Grid and Hex-Plane Encoding generates representations from sparse LiDAR point clouds.
- Optical Flow-Based Temporal Modelling uses a dedicated MLP to track dynamic objects (e.g., moving cars) across frames.
- Multi-Property Prediction MLPs to predict depth, intensity, ray-drop probability, and label.
- Ray-Drop Refinement and Feature Alignment employs a U-Net with a feature alignment loss to generate raydrop masks that match real-world scenarios.

2 Related Works

The synthetic data generation methods for LiDAR point clouds are broadly categorized into physics-based simulators, generative models, and 3D synthesis techniques like NeRFs and Gaussian Splatting. Physics-based approaches use ray-casting and game engines to simulate LiDAR interactions. CARLA [4] is a common choice, with variants simulating adverse weather conditions [14], pedestrian behaviour [22], and semantic tasks [18]. PCGen [11] models sensor effects like ray-drop and noise, while LiDARsim [16] merges ray-tracing with learning. These systems offer customizability but face domain gaps, asset reliance, and high costs.

Data-driven methods reduce these issues by synthesizing or modifying scans using VAEs [29, 33], GANs [28], diffusion models [9, 21], and transformers [13]. UltraLiDAR [29] and LiDAR-GRIT [8] use VQ-VAE and transformers for novel view synthesis, while RangeLDM [21] improves structure via geometric priors. Text2LiDAR [31] enables text-driven generation. Yet, these models are limited to static scenes and fixed sensors configuration.

Techniques like neural rendering [17] and Gaussian Splatting [10] offer promising alternatives that address the issues mentioned above and generate highly realistic data. Further enhancements to NeRFs, such as Mip-NeRF [1], Zip-NeRF [3], and Mip-NeRF 360 [2], improve their performance by further addressing aliasing, rendering efficiency, and support for unbounded scenes.

Due to these reasons, a recent body of work has focused on using NeRFs for generating novel LiDAR scans. LiDAR-NeRF [26] is the first work that uses neural rendering on LiDAR data. It also supports scene editing by inserting new objects in the generated scans. NeRF-LiDAR [30], on the other hand, uses RGB camera images as conditional input and a feature alignment loss to generate labelled LiDAR scans. However, it remains limited to static scenes. LiDAR4D [32] addresses dynamic environments by introducing a spatiotemporal framework that models geometry and motion jointly, though it does not support label generation.

3 Methodology

3.1 Label Transfer

We have implemented our model on the KITTI-360 [15] dataset. The dataset provides detailed annotations for both static and dynamic outdoor elements. It includes 3D semantic LiDAR point cloud maps that span multiple timesteps. However, individual 3D scans corresponding to a single timestep are not annotated. To solve this, we have employed a technique for transferring semantic labels from the global map to each individual LiDAR point cloud scan. To transfer labels from the full map to individual scans, we start with each raw LiDAR point cloud from the KITTI-360 dataset, containing spatial coordinates (x, y, z) and intensity. These scans are transformed from the LiDAR's local frame to the global world frame using the dataset's transformation matrices. Labels are assigned via nearest neighbor matching: for each transformed point, we find the closest point in the full semantic map using k-d trees(a spatial data structure optimized for fast nearest neighbor search in large datasets), then transfer the corresponding label. The labelled scans are then transformed back to the LiDAR coordinate system to preserve the sensor perspective.

3.2 Adapting Neural Radiance Fields for LiDAR Data

The problem formulation of NeRFs for LiDAR data is similar to the traditional NeRF model. Here, instead of casting camera rays from a specific camera position, we cast LiDAR rays. The rays originate from a set of sensor positions, $X = \{X_1, X_2, \ldots, X_n\}$, at different timestamps, $T = \{T_1, T_2, \ldots, T_n\}$. The rays can be depicted as $r(t) = o + td$, where o represents the ray origin, and d is the ray direction. We then sample points along these cast rays and find the density (σ) for each of these points (p_i). Using the densities, we then calculate the opacity (α) and visibility weights (w_i) as shown in Equation 1.

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i), \quad w_i = \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{1}$$

The opacity values determine how much light a sampled point blocks, while the visibility weights represent its contribution to the final predicted property (e.g., depth, intensity). Since NeRFs typically compare rendered and ground-truth RGB images on a pixel-by-pixel basis, they cannot be directly applied to LiDAR data, which is inherently 3D. To bridge this gap, we project point clouds into range-view images. These images represent the 3D data via 2D panoramic or spherical projections, where each pixel represents an angular direction and encodes depth, intensity, and semantic label. To project a 3D point (x, y, z) to a 2D image coordinate (h, w), we compute the azimuth angle $\phi = \pi - \arctan 2(y, x)$, representing the horizontal direction, and the elevation angle $\theta = \arctan 2(z, \sqrt{x^2 + y^2})$, representing the vertical direction. These angles are then mapped to a 2D image plane of dimension (H, W) using Equation 2.

$$h = \left\lfloor \left(1 - \frac{\theta - \theta_{\text{down}}}{\theta_{\text{up}} - \theta_{\text{down}}}\right) \cdot H \right\rfloor, \quad w = \left\lfloor \frac{\phi}{2\pi} \cdot W \right\rfloor, \tag{2}$$

Here, θ_{up} and θ_{down} are the vertical field of view range of the LiDAR sensor, provided with the dataset.

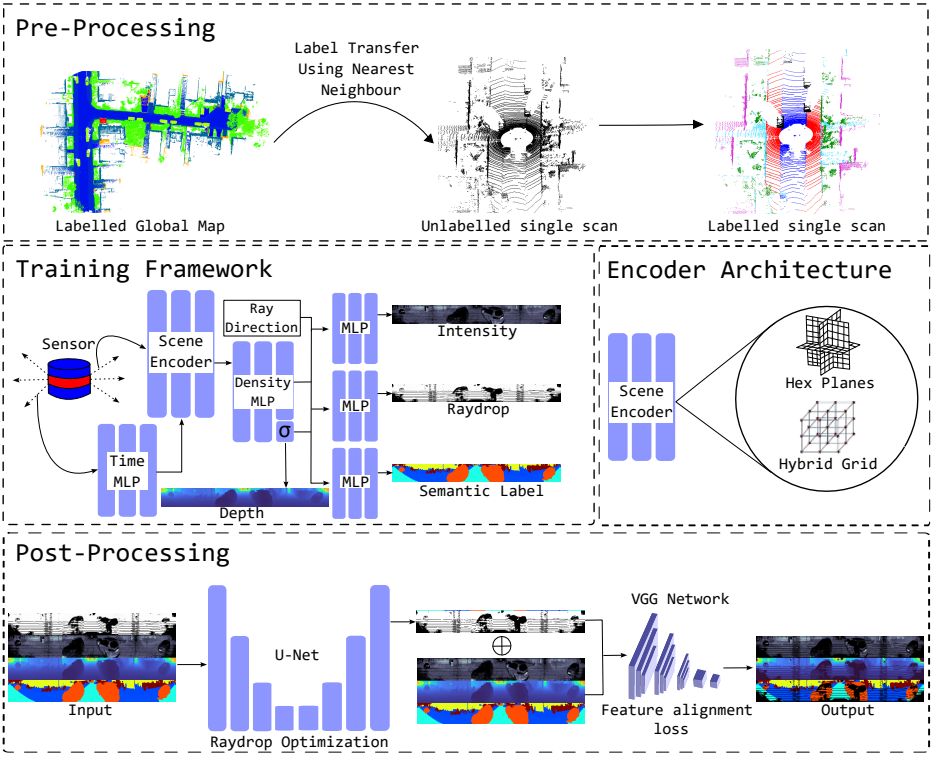

Figure 1: The figure illustrates our three-step methodology. In the preprocessing stage (top), we project individual scans and the global labelled map into a common coordinate frame, then assign labels by finding the nearest global point for each point in the scan. The core framework (middle) employs a scene encoder with hex planes and hybrid grids, along with a time-aware MLP, to model both spatial and dynamic components of the scenes. These encodings are used to predict depth, intensity, raydrop, and semantic labels. In the post-processing stage (bottom), a U-Net with VGG-based feature alignment loss refines the output by generating realistic raydrop masks.

3.3 Lid-Lab-NeRF Framework

3.3.1 Scene Encoders

We begin by generating encodings of the 4D LiDAR inputs (x, y, z, t) sampled from the cast ray at each timestep. Since NeRF training is computationally intensive, we design a specialized encoder that combines hex planes [7] with hybrid grids [19]. This design addresses the limitations of traditional sinusoidal positional encodings, which require dense and uniform sampling to capture multi-scale structures effectively. In contrast, LiDAR data is inherently sparse and uneven, leading sinusoidal encodings to introduce artifacts in under-sampled regions and amplify high-frequency sensor noise [25].

We employ **Hex-Planes** to model both static and dynamic scene components using six orthogonal planes: three spatial planes ($\mathcal{P}_{xy}$, $\mathcal{P}_{xz}$, $\mathcal{P}_{yz}$) and three spatio-temporal planes ($\mathcal{P}_{xt}$, $\mathcal{P}_{yt}$, $\mathcal{P}_{zt}$). A sampled point $q = (x, y, z, t)$ is projected onto each plane, and its encoding is

obtained by bilinearly interpolating features from the plane's nodes at multiple spatial and temporal resolutions, as formulated in Equation 3.

$$f_{\text{plane}}(q) = \prod_{p \in \mathcal{P}} \psi(p, \pi_p(q)), \tag{3}$$

Here, π_p denotes projection onto plane p, and ψ represents bilinear interpolation. Features from multiple resolutions within each plane are concatenated, and the resulting vectors are fused across planes using the Hadamard product.

Since Hex-Planes alone cannot capture the fine-grained details of LiDAR data, we incorporate **multi-resolution hash grids** $\mathcal{G} = \{\mathcal{G}_{xyz}, \mathcal{G}_{xyt}, \mathcal{G}_{xzt}, \mathcal{G}_{yzt}\}$, separated into static and dynamic encoders. As with Hex-Planes, points are projected into these grids at various spatial and spatio-temporal resolutions, and encodings are obtained via trilinear interpolation, as formulated in Equation 4.

$$f_{grid}(q) = \prod_{g \in \mathcal{G}} \phi(g, \pi_g(q)), \tag{4}$$

Here, ϕ denotes multi-resolution sampling. All features are concatenated and combined similarly as in the case of hex planes.

3.3.2 Flow-based Temporal Aggregation

To capture the motion of dynamic objects across scans, we adopt an MLP-based scene flow model that predicts a 3D flow field $\Delta\mathbf{x}$ from 4D $((x,y,z,t))$ inputs. Given a 4D point $\mathbf{q} = (\mathbf{x}, t)$, where $\mathbf{x} \in \mathbb{R}^3$ and $t \in [0,1]$, the input is encoded (separately from the scene encoders) and passed to a flow network to estimate forward ($\Delta\mathbf{x}+$) and backward ($\Delta\mathbf{x}-$) displacements as shown in Equation 5.

$$\mathbf{x}\pm 1 = \mathbf{x} + \Delta\mathbf{x}\pm, \tag{5}$$

Here, $\mathbf{x}+1$ and $\mathbf{x}-1$ denote predicted positions at adjacent timestamps $t+1$ and $t-1$, respectively. Using these estimates, we query scene encoders to extract encodings at the original and displaced coordinates, where $f_0 = f(\mathbf{x}, t)$, $f_{+1} = f(\mathbf{x}+1, t+1)$, and $f-1 = f(\mathbf{x}-1, t-1)$. Once the features are extracted, we then combine them in a weighted combination using Equation 6.

$$f_{\text{agg}} = 0.5 f_0 + 0.25(f_{+1} + f_{-1}). \tag{6}$$

To supervise the predicted displacements, we calculate the loss between predicted adjacent frames and the actual adjacent frames using the Chamfer Distance (CD) loss [5], which has been widely used in scene flow estimation tasks [12], as shown in Equation 7.

$$\mathcal{L}_{CD} = \frac{1}{K} \sum_{\hat{p}i \in S} \min p_i \in S' |\hat{p}i - p_i|_2^2 + \frac{1}{K} \sum p_i \in S' \min_{\hat{p}i \in S} |p_i - \hat{p}i|_2^2, \tag{7}$$

Here, $\hat{p}_i \in S$ denotes a predicted point from the generated point set S (predicted adjacent frames), and $p_i \in S'$ denotes a ground truth point from the reference set S' (actual adjacent frames).

3.3.3 LiDAR Data Attribute Prediction

To predict LiDAR attributes such as depth, raydrop, intensity, and semantic labels, a density MLP first outputs a scalar density value (σ) and a 15D geometric feature vector for each sampled 3D point along the sensor ray. Depth is then computed by integrating all sampled points along the ray using the visibility weights. After this, the geometric feature vector together with the ray direction $\mathbf{d} = (\cos\theta\cos\phi, \sin\theta\cos\phi, \sin\phi)$ is passed to three separate MLPs to predict intensity, raydrop probability, and semantic labels. The output for the intensity and raydrop probability is a scalar value, while for the semantic label, it is a probability distribution vector over the label classes. The overall attribute prediction process is formulated in Equation 8.

$$Depth = \sum_i w_i d_i, \quad Intensity = \sum_i w_i i_i, \quad Raydrop = \sum_i w_i r_i, \quad Label = \sum_i w_i l_i, \tag{8}$$

Here, d_i, i_i, r_i, and l_i represent depth, intensity, raydrop value and label probability for each of the sampled points. From the raydrop prediction, a raydrop mask is generated by thresholding at 0.5, and the semantic labels for a point (k) are assigned using $\arg\max_k Label_k$.

We employ Mean Squared Error (MSE) loss for depth, intensity, and raydrop predictions and Categorical Cross Entropy (CCE) loss for label prediction between the rendered value and the actual ground truth, as depicted in Equation 9.

$$\begin{gathered}\mathcal{L}_{\text{depth}} = \sum_{r\in R} \|\hat{D} - D(r)\|_2^2, \; \mathcal{L}_{\text{intensity}} = \sum_{r\in R} \|\hat{I} - I(r)\|_2^2, \; \mathcal{L}_{\text{raydrop}} = \sum_{r\in R} \|\hat{R} - R(r)\|_2^2, \\ \mathcal{L}_{\text{label}} = -\sum_{i=1}^{C} L_i \log(\hat{L}_i),\end{gathered} \tag{9}$$

Therefore, for the training of the main framework, the loss function can be written as, $\mathcal{L}_{\text{total}} = \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{intensity}}\mathcal{L}_{\text{intensity}} + \lambda_{\text{raydrop}}\mathcal{L}_{\text{raydrop}} + \lambda_{\text{label}}\mathcal{L}_{\text{label}} + \lambda_{\text{CD}}\mathcal{L}_{\text{CD}}$

3.4 Post-Processing via Raydrop Optimization and Feature Alignment

To generate more realistic scans that match actual LiDAR sensors, we introduce a post-training refinement phase known as raydrop optimization, along with a feature alignment loss. This refinement improves the raydrop mask quality and ensures high-level feature similarity with the real data. During inference, a multi-channel tensor of the predicted maps is created by combining raydrop, intensity, depth and label rangeview maps. This tensor is passed through a UNet [23] to refine raydrop predictions. The input is partially masked with randomly zeroed patches. The UNet outputs a reconstructed raydrop mask $\hat{Y}$, supervised via Binary Cross Entropy loss as shown in Equation 10.

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{10}$$

Here, y_i and $\hat{y}_i$ denote ground truth and predicted raydrop values. Additionally, we apply a feature alignment loss to make the final predicted mask similar to the real-world data, even in higher-order feature space. To achieve this, we employ a VGG19 feature extractor[24] which takes intensity, depth and label rangeview maps as input after being multiplied by

the raydrop mask generated from the UNet. The loss is computed hierarchically with the corresponding ground truth scan across feature levels using the Equation 11.

$$\mathcal{L}_{\text{feat}} = \left\| \phi(X_{\text{VGG}} \odot \hat{Y}) - \phi(X_{\text{gt}}) \right\|_2^2, \tag{11}$$

Here, the function $\phi(\cdot)$ represents a fixed, pre-trained feature extractor (a VGG-style network that maps input tensors to high-level perceptual feature representations), X_{gt} is the ground truth feature tensor, and $\odot$ denotes element-wise multiplication. The combined loss for the training of the UNet can be written as, $\mathcal{L}_{\text{UNet}} = \lambda_{\text{BCE}}\mathcal{L}_{\text{BCE}} + \lambda_{\text{feat}}\mathcal{L}_{\text{feat}}$.

4 Results

We evaluate our framework on the KITTI-360 dataset. Training the full model takes approximately 150 minutes on an NVIDIA A100 GPU, with a peak memory usage of 20GB. During inference, generating 51 LiDAR scans requires about 3 minutes. We provide a comprehensive comparison with existing LiDAR NVS methods, with quantitative results reported in Table 1 and qualitative examples shown in Figures 2–6.

Table 1: Quantitative comparison on KITTI-360 dataset. We compare our method to different previous approaches and highlight the best results in **bold** and the second-best results in *italic*. Results are collected from the literature survey of [32].

Method	Point Cloud		Depth					Intensity				
	CD↓	F-score↑	RMSE↓	MedAE↓	LPIPS↓	SSIM↑	PSNR↑	RMSE↓	MedAE↓	LPIPS↓	SSIM↑	PSNR↑
LiDARsim [43]	3.2228	0.7157	6.9153	0.1279	0.2926	0.6342	21.4608	0.1666	0.0569	0.3276	0.3502	15.5853
NKSR [24]	1.8982	0.6855	5.8403	0.0996	*0.2752*	0.6409	23.0368	0.1742	0.0590	0.3337	0.3517	15.2081
PCGen [31]	0.4636	0.8023	5.6583	0.2040	0.5391	0.4903	23.1675	0.1970	0.0763	0.5926	0.1351	14.1181
D-NeRF [41]	0.1442	*0.9128*	*4.0194*	0.0508	0.3061	*0.6634*	*26.2344*	0.1369	*0.0440*	0.3409	0.3748	*17.3554*
TiNeuVox-B [8]	0.1748	0.9059	4.1284	*0.0502*	0.3427	0.6514	26.0267	*0.1363*	0.0453	0.4365	0.3457	17.3535
K-Planes [8]	*0.1302*	0.9123	4.1322	0.0539	0.3457	0.6385	26.0236	0.1415	0.0498	0.4081	0.3008	17.0167
LiDAR-NeRF [42]	0.1438	0.9091	4.1753	0.0566	0.2797	0.6568	25.9878	0.1404	0.0443	*0.3135*	*0.3831*	17.1549
Lid-Lab-Nerf (ours)	**0.1183**	**0.9154**	**2.9504**	**0.0317**	**0.0768**	**0.8629**	**28.6802**	**0.1056**	**0.0263**	**0.1466**	**0.6217**	**19.5330**

The proposed approach is quantitatively compared with prior methods using standard metrics for both point cloud quality and image reconstruction. For point clouds, Chamfer Distance (CD) and F-score (5 cm threshold) are reported to capture geometric accuracy and completeness. For image-based evaluation, RMSE, MedAE, PSNR, LPIPS, and SSIM are used to assess depth and intensity fidelity.

As shown in Table 1, the method outperforms all major LiDAR-NVS baselines across these metrics. Unlike LiDAR-NeRF, which relies solely on grid-based features and suffers from over-smoothing and blurred boundaries, the hybrid representation captures both global structure and local details, producing sharper and more consistent reconstructions in depth and intensity (Figure 2). Combining K-Planes with grid-based encoding further improves geometric and visual quality compared to using either alone, as illustrated in Figure 3, where outputs are more faithful to the ground truth.

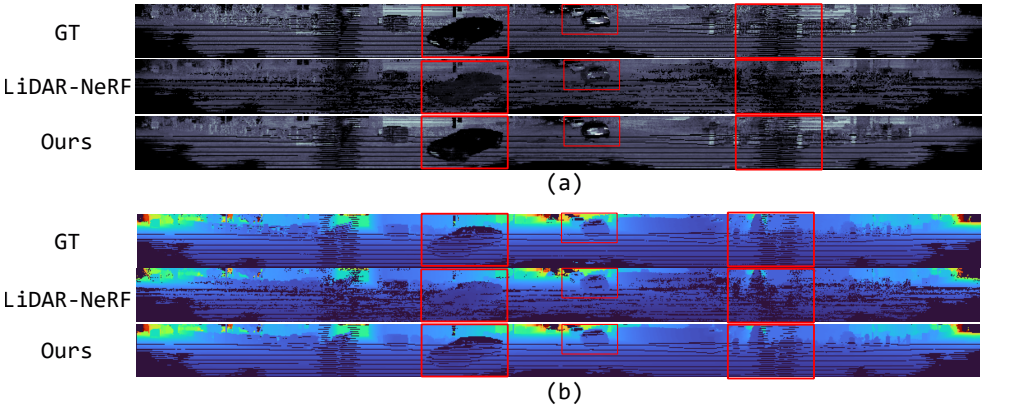

Figure 2: The range-view images offer a qualitative comparison of (a) simulated intensity and (b) simulated depth for LiDAR-NeRF and the proposed framework against ground truth. Highlighted regions show that the method produces sharper boundaries, less blurring, improved raydrop patterns, and better preservation of fine details, resulting in outputs that more closely match the ground truth.

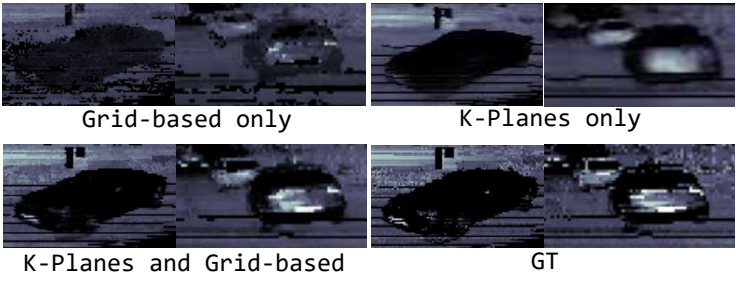

Figure 3: The above image shows how the generated output changes based on the different encoder configurations.

The reduction in CD and improvement in F-score over LiDAR-NeRF stem from the use of a UNet-based architecture and a feature alignment loss, which together enhance raydrop prediction and preserve high-level structure (Figure 4). The impact of this post-processing step is further quantified in Table 2, which compares model performance with and without the raydrop refinement pipeline.

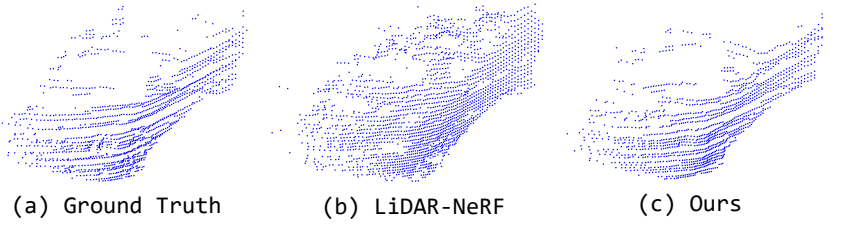

Figure 4: Comparison of point cloud patterns from real data, LiDAR-NeRF, and our method.

Table 2: Effect of the raydrop refinement pipeline (UNet + feature alignment loss) on output quality. Results show significant improvement with the refinement applied.

Method	Point Cloud		Depth					Intensity				
	CD↓	F-score↑	RMSE↓	MedAE↓	LPIPS↓	SSIM↑	PSNR↑	RMSE↓	MedAE↓	LPIPS↓	SSIM↑	PSNR↑
Without Raydrop refinement	0.2253	0.88214	6.4378	0.1137	0.5120	0.5175	21.8879	0.1533	0.0575	0.4140	0.2417	16.2914
With Raydrop refinement	**0.1183**	**0.9154**	**2.9504**	**0.0317**	**0.07687**	**0.86299**	**28.6802**	**0.1056**	**0.0263**	**0.1466**	**0.62177**	**19.533**

4.1 Semantic Label Quality and Downstream Usability

Our framework generates semantic labels for each pixel in the rangeview image. We segment the scene into six major classes: sidewalk, road, building, vehicle, vegetation, and miscellaneous. As shown in Table 3, our model assigns pixel-wise labels with high precision, achieving a mean class accuracy of 95.32%, indicating balanced performance across all six categories. The evidence for balanced performance can also be seen from Table 4

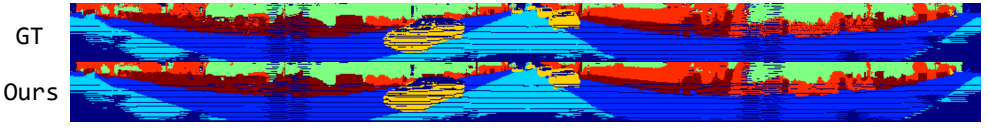

Figure 5: Visual comparison of per-pixel semantic label predictions between our ground truth annotations and our model.

Table 3: Per-pixel (or per-point) label prediction performance on the generated rangeview label images.

Metric	Accuracy	Mean Class Acc.	Precision	Recall	F1 Score
Value	0.9577	0.9532	0.9578	0.9577	0.9577

Table 4: Per-class pixel-wise label prediction accuracy on the generated rangeview images.

Class	Sidewalk	Road	Building	Vehicle	Vegetation	Others
Accuracy	0.9755	0.9839	0.9583	0.9488	0.9387	0.9138

The usability of the generated scans was assessed through a segmentation task. Two separate MinkUNet models were trained: one on real KITTI-360 data and another on an equal volume of synthetically generated data (39 scans each), produced using the same trajectory and configuration as the real ego-vehicle. Both models were evaluated on the same set of 12 real KITTI-360 scans for a fair comparison. As shown in Table 5, the real-data model achieved a mean Intersection over Union (IoU) of 27.49%, only slightly higher than the 27.00% obtained with synthetic data. The synthetic-data model outperformed in specific classes such as “vehicle,” “vegetation,” and “others,” demonstrating that the generated scans can rival real-world data in targeted segmentation tasks. These results also indicate a substantial reduction in the domain gap often observed with synthetic datasets.

Table 5: Segmentation performance of MinkUNet trained on real KITTI-360 data and our synthetically generated data. The real-data model attains a slightly higher overall mIoU, while the synthetic-data model shows competitive performance and leads in specific classes.

Model	Sidewalk IoU	Road IoU	Building IoU	Vehicle IoU	Vegetation IoU	Others IoU	Mean IoU
MinkUNet (trained on real KITTI-360)	35.45	36.55	33.89	40.25	14.92	3.84	27.49
MinkUNet (trained on our synthetic data)	34.85	33.67	30.41	40.39	16.70	5.93	27.00

4.2 Visual Comparison of Rendered Dynamic Objects

The usage of the time flow MLP along with Chamfer Distance loss allows the framework to learn and capture the spatiotemporal point-wise motion dynamics. Moreover, as shown in Figure 6, the moving vehicle captured in a single scan is not accurately reconstructed by LiDAR-NeRF, resulting in an incomplete and distorted rendering. In contrast, our model captures significantly more information, providing a more complete and coherent representation of the vehicle.

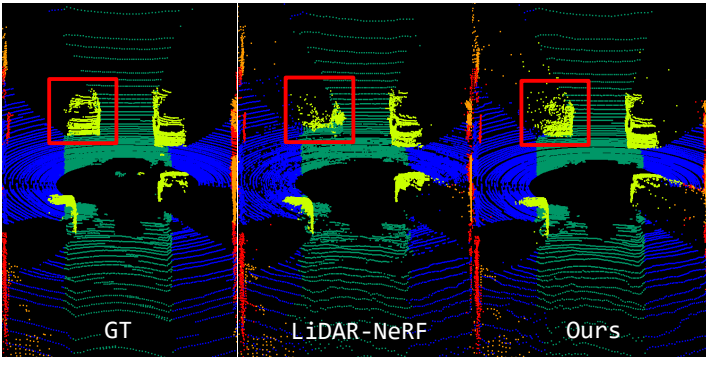

Figure 6: Scan no. 4965 from the KITTI-360 dataset compared across LiDAR-NeRF as well as our model. The red box indicates the dynamic or moving vehicle. As visible from the figure, our model captures moving objects better than LiDAR-NeRF and is closer to the ground truth.

5 Conclusion

The work introduces Lid-Lab-NeRF, a novel framework that synthesizes temporally consistent, labelled LiDAR point clouds using neural radiance fields. By incorporating hybrid encoding strategies, motion-aware modelling, and specialized post-processing, our approach bridges the gap between synthetic and real LiDAR data. Extensive experiments on the KITTI-360 dataset demonstrate superior performance across visual, geometric and semantic benchmarks. Unlike prior methods, our model accurately captures dynamic scenes as well as delivers high-quality semantic labels. The results highlight its effectiveness in both data generation and downstream perception tasks. We believe Lid-Lab-NeRF sets a new direction for scalable, label-rich LiDAR simulation.

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.

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022.

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19697–19705, 2023.

[4] 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*, volume 78 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, 2017. URL https://proceedings.mlr.press/v78/dosovitskiy17a.html.

[5] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 605–613. IEEE, 2017. doi: 10.1109/CVPR.2017.70.

[6] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[7] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023.

[8] Hamed Haghighi, Amir Samadi, Mehrdad Dianati, Valentina Donzella, and Kurt Debattista. Taming transformers for realistic lidar point cloud generation. *arXiv preprint arXiv:2404.05505*, 2024.

[9] Qianjiang Hu, Zhimin Zhang, and Wei Hu. Rangeldm: Fast realistic lidar point cloud generation. In *European Conference on Computer Vision*, pages 115–135. Springer, 2024.

[10] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4): 139–1, 2023.

[11] Chenqi Li, Yuan Ren, and Bingbing Liu. Pcgen: Point cloud generator for lidar simulation. *arXiv preprint arXiv:2210.08738*, 2022. URL https://doi.org/10.48550/arXiv.2210.08738.

[12] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f9e3fbd4974c93b04fe2b9d3acb206c2-Abstract.html.

[13] Y. Li et al. Generative ai empowered lidar point cloud generation with multimodal transformer. *arXiv preprint arXiv:2406.18542*, 2024.

[14] Yifan Li, Yifan Wang, Yifan Liu, Yifan Zhang, Yifan Chen, Yifan Yu, Yifan Wang, and Yifan Wang. Realistic rainy weather simulation for lidars in carla simulator. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024. URL https://arxiv.org/abs/2312.12772. arXiv:2312.12772.

[15] 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*, 45(3):3292–3310, 2022.

[16] 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. *arXiv preprint arXiv:2006.09348*, 2020. URL https://doi.org/10.48550/arXiv.2006.09348.

[17] 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.

[18] Javier Montalvo, Pablo Carballeira, and Álvaro García-Martín. Synthmanticlidar: A synthetic dataset for semantic segmentation on lidar imaging. In *2024 IEEE International Conference on Image Processing (ICIP)*, pages 137–143. IEEE, 2024.

[19] 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.

[20] 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.

[21] Haoxi Ran, Vitor Guizilini, and Yue Wang. Towards realistic scene generation with lidar diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. URL https://arxiv.org/abs/2404.00815.

[22] Muhammad Naveed Riaz, Maciej Wielgosz, Abel García Romera, and Antonio M. López. Synthetic data generation framework, dataset, and efficient deep model for pedestrian intention prediction. *arXiv preprint arXiv:2401.06757*, 2024. URL https://arxiv.org/abs/2401.06757.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28. URL https://arxiv.org/abs/1505.04597.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] Shanlin Sun, Bingbing Zhuang, Ziyu Jiang, Buyu Liu, Xiaohui Xie, and Manmohan Chandraker. Lidarf: Delving into lidar for neural radiance field on street scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19563–19572, 2024.

[26] Tang Tao, Longfei Gao, Guangrun Wang, Yixing Lao, Peng Chen, Hengshuang Zhao, Dayang Hao, Xiaodan Liang, Mathieu Salzmann, and Kaicheng Yu. Lidar-nerf: Novel lidar view synthesis via neural radiance fields. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 390–398, 2024.

[27] Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural fields as learnable kernels for 3d reconstruction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18479–18489, New Orleans, LA, USA, 2022. IEEE. doi: 10.1109/CVPR52688.2022.01795.

[28] Zhengkang Xiang, Zexian Huang, and Kourosh Khoshelham. Synthetic lidar point cloud generation using deep generative models for improved driving scene object recognition. *Image and Vision Computing*, 2024.

[29] Yuwen Xiong, Wei-Chiu Ma, Jingkang Wang, and Raquel Urtasun. Ultralidar: Learning compact representations for lidar completion and generation. *arXiv preprint arXiv:2311.01448*, 2023.

[30] Junge Zhang, Feihu Zhang, Shaochen Kuang, and Li Zhang. Nerf-lidar: Generating realistic lidar point clouds with neural radiance fields. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(7):7178–7186, 2024. doi: 10.1609/aaai.v38i7.28546.

[31] Yifan Zhang, Yiming Wang, Xinyu Zhang, Yifan Chen, Ping Luo, Yu Wang, and Cewu Lu. Text2lidar: Text-guided lidar point cloud generation via equirectangular transformer. *arXiv preprint arXiv:2407.19628*, 2024. URL https://arxiv.org/abs/2407.19628.

[32] Zehan Zheng, Fan Lu, Weiyi Xue, Guang Chen, and Changjun Jiang. Lidar4d: Dynamic neural fields for novel space-time view lidar synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5145–5154, 2024.

[33] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *European Conference on Computer Vision (ECCV)*, 2022.