Dynamic LiDAR Re-simulation using Compositional Neural Fields

Hanfeng Wu[1,2] Xingxing Zuo[2,*] Stefan Leutenegger[2]
Or Litany[3,4] Konrad Schindler[1] Shengyu Huang[1,*]
[1] ETH Zurich [2] TU Munich [3] Technion [4] Nvidia

Abstract

We introduce DyNFL, a novel neural field-based approach for high-fidelity re-simulation of LiDAR scans in dynamic driving scenes. DyNFL processes LiDAR measurements from dynamic environments, accompanied by bounding boxes of moving objects, to construct an editable neural field. This field, comprising separately reconstructed static backgrounds and dynamic objects, allows users to modify viewpoints, adjust object positions, and seamlessly add or remove objects in the re-simulated scene. A key innovation of our method is the neural field composition technique, which effectively integrates reconstructed neural assets from various scenes through a ray drop test, accounting for occlusions and transparent surfaces. Our evaluation with both synthetic and real-world environments demonstrates that DyNFL substantial improves dynamic scene simulation based on LiDAR scans, offering a combination of physical fidelity and flexible editing capabilities.

1. Introduction

We introduce a neural representation for the purpose of reconstructing and manipulating LiDAR scans of dynamic driving scenes. Counterfactual re-simulation is an emerging application in the realm of autonomous driving, offering a unique approach to examining "what if" scenarios. This method involves creating a reconstruction of a real-world event, termed as *digital twin* and then applying various modifications to it. These could include altering the environmental conditions, changing the action of some agent, or introducing additional scene elements. Analyzing the outcomes of these edited scenarios provides insights into the functioning of the perception system, moreover they can be used to obtain training data for rare situations.

The essence of counterfactual re-simulation is the capability to authentically recreate variations of the original, factual observation. We address this challenge in the context of LiDAR on autonomous vehicles (AV). Existing approaches to LiDAR re-simulation have important limitations. Conventional simulators such as CARLA [9] and NVIDIA DRIVE Sim are capable of modeling LiDAR sensors. However, their reliance on manually designed 3D simulation assets requires significant human effort. LiDAR-sim [27] aims to remedy this by reconstructing vehicles and scenes from real measurements. While producing encouraging results, its two-stage LiDAR modeling lacks realism, particularly in terms of physical effects like multi-returns and reflected intensity, which were shown to matter for downstream processing [15]. Following NeRF's [28] success in camera view synthesis, some works have applied neural fields for LiDAR modeling [19, 44, 59]. In particular, Neural LiDAR Fields (NFL)[19] developed a physically inspired LiDAR volumetric rendering scheme that accounts for two-way transmittance and beam width, allowing faithful recovery of secondary returns, intensity, and ray drops. These models are, however, limited to static scenes that do not change while multiple input views are scanned, and are thus of limited use for re-simulation in the presence of moving traffic. Recently, UniSim [55] followed Neural Scene Graph [32] in modeling road scenes as sets of movable NeRF instances on top of a static background. UniSim introduced a unified synthesis approach for camera and LiDAR sensors, but ignored physical sensor properties like two-way transmittance and beam width [19].

We present DyNFL, a novel approach for re-simulating LiDAR views of driving scenarios. Our method builds upon a neural SDF that enables an accurate representation of scene geometry, while at the same time enforcing physical accuracy by modeling two-way transmittance, like NFL [19]. Our primary contribution is a method for compositing neural fields that accurately integrates LiDAR measurements from individual fields representing different scene assets. With the help of a ray drop test, we effectively manage occlusions and transparent surfaces. This not only ensures physical accuracy, but also facilitates the inclusion of assets reconstructed from a variety of static and dynamic scenes, thereby enhancing control over the simulated content. Our method bridges the gap between the physical fidelity of the re-simulation and flexible editing of

*Corresponding authors

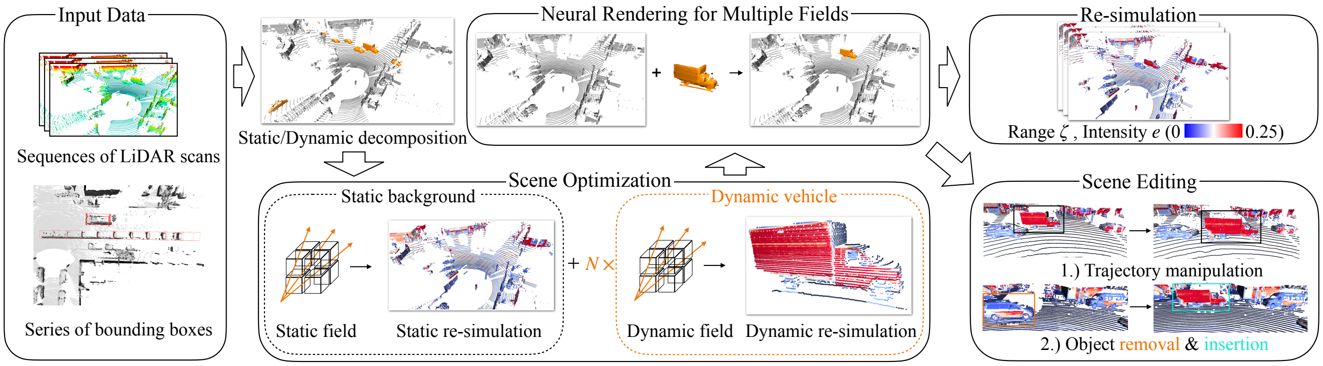

Figure 1. Overview of DyNFL. Our method takes LiDAR scans and tracked bounding boxes of dynamic vehicles as input. DyNFL first decomposes the scene into a static background and N dynamic vehicles, each modelled using a dedicated neural field. These neural fields are then composed to re-simulate LiDAR scans in dynamic scenes. Our composition technique supports various scene edits, including altering object trajectories, removing and adding reconstructed neural assets between scenes.

dynamic scenes. We validate DyNFL with both synthetic and real-world data, focusing on three key areas: *(i)* high-quality view synthesis, *(ii)* perceptual fidelity, and *(iii)* asset manipulation. We find that our approach outperforms baseline models w.r.t. both range and intensity. Its synthetic outputs also show higher agreement with real scans in terms of object detection and segmentation. Furthermore, DyNFL enables not only removal, duplication and repositioning of assets within the same scene, but also the inclusion of assets reconstructed in other scenes, paving the way for new applications.

2. Related work

2.1. Neural radiance fields and volume rendering

Neural Radiance Fields (NeRF) [28] have demonstrated remarkable success in novel-view image synthesis through neural volume rendering. These fields are characterized by the weights of Multilayer Perceptrons (MLPs), which enable the retrieval of volume density and RGB colors at any specified point within the field for image compositing via volume rendering. Several studies [2, 3, 7, 12, 46] have subsequently advanced NeRF's rendering quality by addressing challenges such as reducing aliasing artifacts [2], scaling to unbound large-scale scenarios [3], and capturing specular reflections on glossy surfaces [46]. Certain works [7, 12, 20, 29] have explored more effective representations of radiance fields. TensorsRF [7] employs multiple compact low-rank tensor components, such as vectors and matrices, to represent the radiance field. Plenoxels [12] accelerates NeRF training by replacing MLPs with explicit plenoptic elements stored in sparse voxels and factorizing appearance through spherical-harmonic functions. Müller et al. [29] achieved a substantial acceleration in rendering speed by employing a representation that combines trainable multi-resolution hash encodings (MHE) with shared shallow MLP networks. Kerbel et al. [20] introduce a novel volume rendering method utilizing 3D Gaussians to represent the radiance field and rendering images based on visibility-aware splatting of 3D Gaussians.

2.2. Dynamic neural radiance fields

Neural fields [53] can be extended to represent dynamic scenes. On top of the *canonical* scene representation, some methods [33–35, 58] additionally model the 4D deformation fields. Meanwhile, some other works learn a space-time correlated [1, 24, 26, 38], or decomposed [45, 52, 54] neural field to encode the 4D scenes, achieving fine-grained reconstruction of the geometry and the appearance. Some other methods decompose the scene into static and dynamic parts, and model each dynamic actor with dedicated neural fields. Neural Scene Graph [32] and Panoptic Neural Fields [21] treat every dynamic object in the scene as a node, and synthesize photo-realistic RGB images by jointly rendering from both dynamic nodes and static background. UniSim[55] employs neural SDF representation to model dynamic scenes in driving scenarios, and render in a similar way to Neural Scene Graph [32].

2.3. Neural surface representation

A fundamental challenge for NeRF and its variants involves accurately recovering the underlying 3D surface from the implicit radiance field. Surfaces obtained by thresholding on the volume density of NeRF often exhibit noise [47, 56]. To address this, implicit surface representations like Occupancy [30, 31] and signed distance functions (SDF) [25, 41, 47–49, 56, 57, 60] in grid maps are commonly integrated into neural volume rendering techniques.

NeuS [47] introduces a neural SDF representation for surface reconstruction, proposing an unbiased weight function for the appearance composition process in volume rendering. Similarly, VolSDF [56] models scenes with a neural SDF and incorporates the SDF into the volume rendering process, advocating a sampling strategy of the viewing ray to bound opacity approximation error. Neuralangelo [25] improves surface reconstruction using the multi-resolution hash encoding (MHE) [29] and SDF-based volume rendering [47]. While these methods might deliver satisfying dense surface reconstructions, their training is time-consuming, taking hours for a single scene. Voxurf [51] offers a faster surface reconstruction method through a two-stage training procedure, recovering the coarse shape first and refining details later. Wang et al. [49] expedites NeuS training to several minutes by predicting SDFs through a pipeline composed of MHE and shallow MLPs.

Many works also incorporate distances measured by LiDAR as auxiliary information to constrain the radiance field. For instance, works [6, 50] render depth by accumulating volume density and minimizing depth discrepancies between LiDAR and render depth during training. Rematas et al. [37] enforces empty space between the actual surface and the ray origin.

2.4. LiDAR simulation

While simulators like CARLA [9] and AirSim [39] can simulate LiDAR data, they suffer from expensive human annotation requirements and a notable sim-to-real gap due to limited rendering quality. Generative model-based methods for LiDAR synthesis [5, 61] offer an alternative but often lack control and produce distorted geometries [23]. Learning-based approaches [11, 23, 27] try to enhance realism by transferring real scan properties to simulations. For example, [15] uses a RINet trained on RGB and real LiDAR data to augment simulated scan qualities. LiDARsim [27] employs ray-surfel casting with explicit disk surfels for more accurate simulations. Huang et al. [19] proposed Neural LiDAR Fields (NFL), combining neural fields with a physical LiDAR model for high-quality synthesis, although it's limited to static scenes and can produce noisy outputs due to its unconstrained volume density representation. UniSim [55] constructs neural scene representations from realistic LiDAR and camera data, using SDF-based volume rendering for sensor measurement generation at novel viewpoints.

3. Dynamic neural scene representation

Problem statement. Consider a set of LiDAR scans $\mathcal{X} = \{\mathbf{X}_t\}_{t=1}^T$ that have been compensated for ego-motion, along with tracked bounding boxes* for dynamic vehicles $\mathcal{B} = \{\mathbf{B}_t^v\}_{v=1}^N$, where T represents the total number of LiDAR scans, and N is the count of dynamic vehicles. Each scan $\mathbf{X}_t$ is composed of n_t rays, each ray $\mathbf{r}$ is described by the tuple $(\mathbf{o}, \mathbf{d}, \zeta, e, p_d)$, where $\mathbf{o}$ and $\mathbf{d}$ denote the ray's origin and direction, ζ and e represent range and intensity values, and $p_d \in \{0, 1\}$ indicates whether the ray is dropped or not due to insufficient returned radiant power.

The goal is to reconstruct the scene with a static-dynamic decomposed neural representation, that can enable the rendering of LiDAR scan $\mathbf{X}_{\text{tgt}}$ from novel viewpoint $\mathbf{T}_{\text{tgt}}$. This setup also facilitates various object manipulations, including altering object trajectories, and inserting or removing objects from the scene. The overview of our method is given in Fig. 1.

3.1. Neural scene decomposition

We leverage the inductive bias that driving scenes can be decomposed into a static component and N rigidly-moving dynamic components [13, 18]. Consequently, we establish $N+1$ neural fields. The neural field $\mathbf{F}_{\text{static}}$ is designated for the static component of the scene, capturing the unchanging background elements. Concurrently, the set of neural fields $\{\mathbf{F}^v\}_{v=1}^N$ is used to model the N dynamic entities, specifically the vehicles in motion.

Neural field for static background. The static background is encoded into a neural field $\mathbf{F}_{\text{static}} : (\mathbf{x}, \mathbf{d}) \mapsto (s, e, p_d)$ that estimates the signed distance s, intensity e, and ray drop probability $p_d \in [0, 1]$ given the point coordinates $\mathbf{x}$ and the ray direction $\mathbf{d}$. In practice, we first use a multi-resolution hash encoding (MRH) [29] to map each point to its positional feature $\mathbf{f}_{\text{pos}} \in \mathbb{R}^{32}$, and project the view direction onto the first 16 coefficients of the spherical harmonics basis, resulting in $\mathbf{f}_{\text{dir}}$. Subsequently, we utilize three Multilayer Perceptrons (MLPs) to estimate the scene properties as follows:

$$(s, \mathbf{f}_{\text{geo}}) = f_s(\mathbf{f}_{\text{pos}}), \quad e = f_e(\mathbf{f}_{\text{ray}}), \quad p_d = f_{\text{drop}}(\mathbf{f}_{\text{ray}}). \tag{1}$$

Here, f_s, f_e, and f_{drop} are three MLPs, $\mathbf{f}_{\text{ray}} \in \mathbb{R}^{31}$ represents the ray feature and is constructed by concatenating the per-point geometric feature and the directional feature. The geometric feature is denoted as $\mathbf{f}_{\text{geo}} \in \mathbb{R}^{16}$. For more implementation details, please refer to the supplementary materials.

Neural fields for dynamic vehicles. LiDAR scans collected over time are often mis-aligned due to the motion of both the sensor and other objects in the scene. Despite applying ego-motion for aligning static background points, dynamic object points remain blurred along their trajectories. Our approach to constructing a dynamic neural scene representation is grounded in the assumption that each dynamic object only undergoes rigid motion. Therefore, we can first align them over time and reconstruct them in their *canonical* coordinate frame, and then render them over time by reversing the alignment of the neural field.

*We assume that the ground truth object detection and tracking annotations are available.

Specifically, consider a dynamic vehicle v occurring in LiDAR scans $\{\mathbf{X}_t^v\}_{t=1}^T$ along with the associated bounding boxes $\{\mathbf{B}_t^v \in \mathbb{R}^{3\times 8}\}_{t=1}^T$ in the world coordinate framework. Here each bounding box is defined by its eight corners, and the first bounding box $\mathbf{B}_1^v$ is considered as the *canonical* box. We estimate the relative transformations $\{\mathbf{T}_t \in \mathrm{SE}(3)\}_{t=2}^T$ between the remaining $T-1$ bounding boxes and the canonical box, expressed as $\mathbf{B}_1^v = \mathbf{T}_t \mathbf{B}_t^{v*}$. Subsequently, all LiDAR measurements on the object are transformed and accumulated in its canonical coordinate frame. The vehicle v is then reconstructed in its canonical space, akin to the static background, using a neural field $\mathbf{F}^v$. To render the dynamic vehicle at timestamp t, the corresponding rigid transformation is applied to the queried rays. The dynamic neural field can thus be expressed as: $\mathbf{F}_t^v : (\mathbf{T}_t\mathbf{x}, \mathbf{T}_t\mathbf{d}) \mapsto (s, e, p_d)$. The rendering process for $\mathbf{F}^v$ is the same as rendering for static neural field $\mathbf{F}_{\text{static}}$.

4. Neural rendering of the dynamic scene

In this section, we present the methodology for rendering LiDAR scans from the neural scene representation. We begin by revisiting the density-based volume rendering formulation for active sensors [19] in Sec. 4.1. Subsequently, we explore the extension of this formulation to SDF-based neural scene representation in Sec. 4.2. Finally, we provide a detailed discussion on rendering LiDAR measurements from individual neural fields in Sec. 4.3 and the process of composing results from different neural fields in Sec. 4.4.

4.1. Volume rendering for active sensor

LiDAR utilizes laser beam pulses to determine the distance to the nearest reflective surface by analyzing full waveform profile of the returned radiant power. The radiant power $P(\zeta)$ from range ζ is the result of a convolution between the pulse power $P_e(t)$ and the impulse response $H(\zeta)$, defined as [16, 17, 19]:

$$P(\zeta) = \int_0^{2\zeta/c} P_e(t) H(\zeta - \frac{ct}{2})\, dt \,. \tag{2}$$

The impulse response $H(\zeta)$ is a product of the target and sensor impulse responses: $H(\zeta) = H_T(\zeta) \cdot H_S(\zeta)$, and the individual components are expressed as:

$$H_T(\zeta) = \frac{\rho}{\pi}\cos(\theta)\delta(\zeta - \bar{\zeta}) \,, \quad H_s(\zeta) = T^2(\zeta)\frac{A_e}{\zeta^2} \,, \tag{3}$$

where ρ represents the surface reflectance, θ denotes incidence angle, $\bar{\zeta}$ is the ground truth distance to the nearest reflective surface, $T(\zeta)$ and A_e describe the transmittance at range ζ and sensor's effective area, respectively. Due to the non-differentiability introduced by the indicator function $\delta(\zeta - \bar{\zeta})$, Eq. (2) is non-differentiable and is thus not suitable for solving the inverse problem. NFL [19] solves it by extending it into a probabilistic formulation given by:

$$P(\zeta) = C \cdot \frac{T^2(\zeta)\cdot \sigma_\zeta \rho_\zeta}{\zeta^2}\cos(\theta) \,. \tag{4}$$

Here, C accounts for the constant values, and σ_ζ represents the density at range ζ. The radiant can be reconstructed using the volume rendering formulation:

$$P = \sum_{j=1}^{N} \int_{\zeta_j}^{\zeta_{j+1}} C \frac{T^2(\zeta)\cdot \sigma_\zeta \rho_\zeta}{\zeta^2}\cos(\theta_j)\, d\zeta = \sum_{j=1}^{N} w_j \rho'_{\zeta_j} , \tag{5}$$

where the weights $w_j = 2\alpha_{\zeta_j} \cdot \prod_{i=1}^{j-1}(1 - 2\alpha_{\zeta_i})$. Here α_{ζ_j} is the discrete opacity at range ζ_j. Please refer to [19] for more details.

4.2. SDF-based volume rendering for active sensor

A neural scene representation based on probabilistic density often results in surfaces with noticeable noise due to insufficient surface regularization [47]. To address this, we opt for a signed distance-based scene representation and establish the volume rendering formulation within the framework of an active sensor. Building upon SDF-based volume rendering for passive sensors [47], we compute the opaque density $\tilde{\sigma}_{\zeta_i}$ as follows:

$$\tilde{\sigma}_{\zeta_i} = \max\left(\frac{-\frac{\mathrm{d}\Phi_s}{\mathrm{d}\zeta_i}(f(\zeta_i))}{\Phi_s(f(\zeta_i))}, 0 \right), \tag{6}$$

where $\Phi_s(\cdot)$ represents the Sigmoid function, $f(\zeta)$ evaluates the signed distance to the surface at range ζ along the ray $\mathbf{r}$.

Next, we substitute the density σ in Eq. (5) with opaque density from Eq. (6) and re-evaluate the radiant power and weights as:

$$P = \sum_{j=1}^{N} \mathcal{T}_{\zeta_j}^2 \tilde{\alpha}_{\zeta_j} \rho'_{\zeta_j}, \quad \tilde{w}_j = 2\tilde{\alpha}_{\zeta_j} \cdot \prod_{i=1}^{j-1}(1 - 2\tilde{\alpha}_{\zeta_i}) \,. \tag{7}$$

In this context, $\tilde{\alpha}_{\zeta_j}$ is computed as:

$$\tilde{\alpha}_{\zeta_j} = \max\left(\frac{\Phi_s(f(\zeta_j))^2 - \Phi_s(f(\zeta_{j+1}))^2}{2\Phi_s(f(\zeta_j))^2}, 0 \right). \tag{8}$$

Please refer to the supplementary for more details.

4.3. Volume rendering for LiDAR measurements

Consider rendering the LiDAR measurements from a single neural field, we employ the hierarchical sampling[47] technique to sample a total of $N_s = N_u + N_i$ points along each

*$\mathbf{TB} = \mathbf{RB} + \mathbf{t}$, where $\mathbf{R}$ and $\mathbf{t}$ are the rotation and translation components of $\mathbf{T}$.

ray, where N_u points are uniformly sampled, and N_i points are probabilistically sampled based on the weights along the ray, facilitating denser sampling in proximity to the surface. Subsequently, we compute the weights for the N_s points following Eq. (8). The rendering of range, intensity, and ray drop for each ray can be expressed through volume rendering as follows: $y_{\text{est}} = \sum_{j=1}^{N_s} w_j y_j$, where $y \in \{\zeta, e, p_d\}$.

4.4. Neural rendering for multiple fields

Our full neural scene representation comprises $N+1$ neural fields as discussed in Sec. 3.1. Rendering from all these fields for each ray during inference is computationally intensive. To address this, we implement a two-stage method. In the first stage, we identify the $k+1$ neural fields, where $k \geq 0$ represents the number of dynamic fields, that are likely to intersect with a given ray. The second stage involves rendering LiDAR measurements from these selected fields individually and then integrating them into a unified set of measurements.

Ray intersection test. As outlined in Sec. 3.1, each dynamic neural field is reconstructed in its unique canonical space, defined by a corresponding canonical box. To determine neural fields intersecting with a ray at inference time, we begin by estimating the transformations $\{\mathbf{T}_t^v\}_{v=1}^N$, which convert coordinates from the world framework to each vehicle's canonical space at timestamp t. These transformations are determined by interpolating the training set transformations using spherical linear interpolation (SLERP) [40]. Following this, we apply transformations to the queried ray and run intersection tests with the canonical boxes of the scenes.

Neural rendering from multiple neural fields. After identifying the $k+1$ neural fields that potentially intersect with a ray, we perform volume rendering on each field separately, yielding $k+1$ distinct sets of LiDAR measurements. Next, we evaluate the ray drop probabilities across these fields. A ray is deemed *dropped* if all neural fields indicate a drop probability $p_d > 0.5$. For rays not classified as dropped, we sort the estimated ranges in ascending order and select the nearest one as our final range prediction. Correspondingly, the intensity value is extracted from the same neural field associated with this closest range.

5. Neural scene optimisation

Given the set of LiDAR scans and the associated tracked bounding boxes of the dynamic vehicles, we optimise our neural scene representation by minimising the loss:

$$\mathcal{L} = w_\zeta \mathcal{L}_\zeta + w_s \mathcal{L}_s + w_{\text{eik}} \mathcal{L}_{\text{eik}} + w_e \mathcal{L}_e + w_{\text{drop}} \mathcal{L}_{\text{drop}}, \quad (9)$$

where w_* denotes respective weights, and each individual loss term $\mathcal{L}_*$ is explained below.

Range reconstruction loss. For range estimation, we employ L1 loss, defined as: $\mathcal{L}_\zeta = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r} \in \mathcal{R}} |\zeta_{est} - \zeta_{gt}|$, where $\mathcal{R}$ denotes the set of LiDAR rays, ζ_{est} and ζ_{gt} correspond to the estimated and actual ranges, respectively.

Surface points' SDF regularisation. Acknowledging that LiDAR points mostly come from actual surface, we introduce an additional SDF regularisation term $\mathcal{L}_s$ that penalizes surface points' SDF values: $\mathcal{L}_s = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} |s(\mathbf{p})|$. Here $\mathcal{P}$ denotes the set of surface points and $s(\mathbf{p})$ represents the SDF value of the point $\mathbf{p}$.

Eikonal constraint. Following [14], we utilize the Eikonal loss, $\mathcal{L}_{\text{eik}}$, to regularize the SDF level set. This ensures the gradient norm of the SDF is approximately one at any queried point. The loss is computed as: $\mathcal{L}_{\text{eik}} = \frac{1}{|\mathcal{Z}|} \sum_{\mathbf{p} \in \mathcal{Z}} (\|\nabla s(\mathbf{p})\|_2 - 1)^2$, where $\mathcal{Z}$ is the set of all the sampled points. To stablise the training procedure, we adopt a numerical approach [25] to compute $\nabla s(\mathbf{p})$ as:

$$\nabla s(\mathbf{p}) = \frac{s(\mathbf{p} + \boldsymbol{\epsilon}) - s(\mathbf{p} - \boldsymbol{\epsilon})}{2\epsilon}, \quad (10)$$

where the numerical step size ϵ is set to be 10^{-3} meters.

Intensity Loss. For intensity reconstruction, we apply L2 loss, defined as: $\mathcal{L}_e = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r} \in \mathcal{R}} (e_{est} - e_{gt})^2$.

Ray drop loss. We follow [19] to supervise the ray drop estimation with a combination of a binary cross entropy loss $\mathcal{L}_{bce}$ and a Lovasz loss $\mathcal{L}_{ls}$ [4] as:

$$\mathcal{L}_{\text{drop}} = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r} \in \mathcal{R}} \left(\mathcal{L}_{bce}(p_{d,est}, p_{d,gt}) + \mathcal{L}_{ls}(p_{d,est}, p_{d,gt})\right). \quad (11)$$

It's worth noting that in the context of dynamic neural fields, during training, we incorporate all LiDAR rays that intersect with the objects' bounding boxes of the scenes. A ray is classified as *dropped* either if it is labeled as such in the dataset or if it does not intersect with the actual surfaces of the dynamic vehicles (*e.g.* rays that are close but in parallel to the surfaces). This approach enhances the accuracy and realism of the reconstructed dynamic neural fields, improving the rendering fidelity at inference time.

6. Experiments

6.1. Datasets and evaluation protocol

Real-world Dynamic scenes. We construct *Waymo Dynamic* dataset by selecting four representative scenes from Waymo Open dataset [42], with multiple moving vehicles inside. These scenes are comprised of sequences of 50 consecutive frames. For evaluation purposes, every fifth frame is designated for testing, while the other 40 frames are allocated for training.

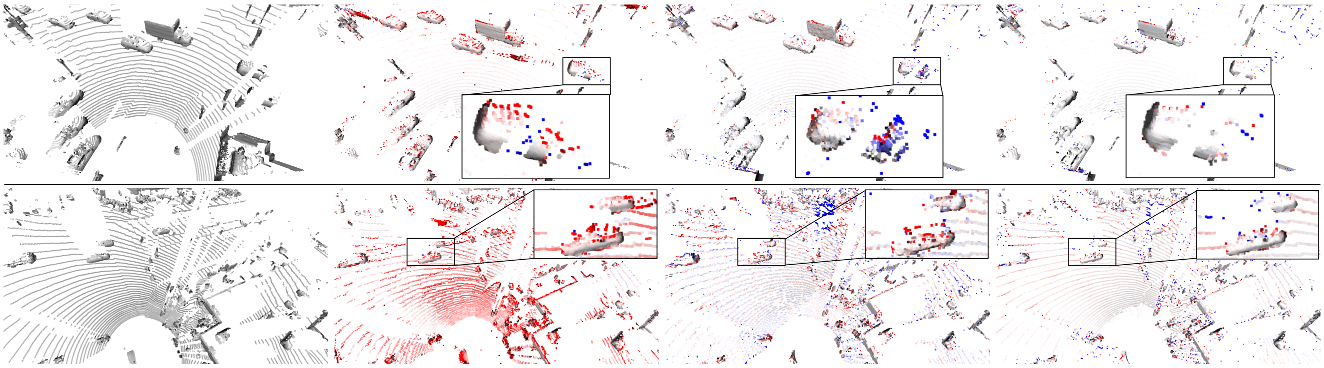

Figure 2. Qualitative comparison of range estimation on *Waymo Dynamic* dataset. Dynamic vehicles are zoomed in, and points are color-coded by range errors (-100 100 cm).

Real-world static scenes. We also evaluate our method on four static scenes as introduced in [19]. There are two settings, *Waymo Interp* applies the same evaluation protocol as *Waymo Dynamic*, while *Waymo NVS* employs a dedicated closed-loop evaluation to validate the real novel view synthesis performance. Please refer to NFL [19] for more details about this setting.

Synthetic static scenes. *TownClean* and *TownReal* are synthetic static scenes introduced in NFL [19]. They consist of 50 LiDAR scans simulated in urban street environment, using non-diverging and diverging beams, respectively.

Evaluation metrics. To evaluate the LiDAR range accuracy, we employ a suite of four metrics: mean absolute errors (MAE [cm]), median absolute errors (MedAE [cm]), Chamfer distance (CD[cm]) and MedAE for dynamic vehicles (MedAE Dyn[cm]). For intensity evaluation, We report root mean square error (RMSE). In addition to our primary evaluations, we assess the re-simulated LiDAR scans' realism through two auxiliary tasks: object detection and semantic segmentation. For object detection, we calculate the *detection agreement* [27], both for all vehicles (Agg. [%]) and specifically for dynamic vehicles (Dyn. Agg. [%]). Regarding semantic segmentation, we measure and report recall, precision, and the intersection over union (IoU[%]). It's important to note that the predictions on the original LiDAR scans serve as our *ground truth*, against which we compare the results obtained from the re-simulated scans.

Baseline methods. Regarding LiDAR simulation on static scenes, NFL [19] and LiDARsim[27] are two closest baselines to compare to. Additionally, we include i-NGP [29], DS-NeRF [8], and URF [37] for comparison. As for simulation on dynamic scenes, we compare to LiDARsim [27] and UniSim [55][*]. Please refer to the supplementary for implementation details.

Method	MAE ↓	MedAE ↓	CD ↓	MedAE Dyn ↓	Intensity RMSE ↓
LiDARsim [27]	170.1	11.5	31.1	16.0	0.10
Unisim [55]	35.6	6.1	14.3	14.3	**0.05**
Ours	**30.8**	**3.0**	**10.9**	**8.5**	**0.05**

Table 1. Evaluation of LiDAR NVS on *Waymo Dynamic* dataset.

	TownClean			TownReal			Waymo interp.			Waymo NVS		
Method	MAE ↓	MedAE ↓	CD ↓	MAE ↓	MedAE ↓	CD ↓	MAE ↓	MedAE ↓	CD ↓	MAE ↓	MedAE ↓	CD ↓
i-NGP [29]	42.2	4.1	17.4	49.8	4.8	19.9	**26.4**	5.5	**11.6**	<u>30.4</u>	7.3	15.3
DS-NeRF [8]	41.7	3.9	16.6	48.9	4.4	18.8	<u>28.2</u>	6.3	14.5	30.4	7.2	16.8
URF [37]	43.3	4.2	16.8	52.1	5.1	20.7	28.2	5.4	12.9	43.1	10.0	21.2
LiDARsim [27]	159.6	<u>0.8</u>	23.5	162.8	3.8	27.4	116.3	15.2	27.6	160.2	16.2	34.7
NFL[19]	<u>32.0</u>	2.3	<u>9.0</u>	<u>39.2</u>	<u>3.0</u>	<u>11.5</u>	30.8	<u>5.1</u>	<u>12.1</u>	32.6	<u>5.5</u>	<u>13.2</u>
Ours	**26.7**	**0.7**	**6.7**	**33.9**	**2.1**	**10.4**	28.3	**4.7**	12.5	**28.6**	**4.9**	**13.0**

Table 2. Evaluation of LiDAR NVS on static scenes.

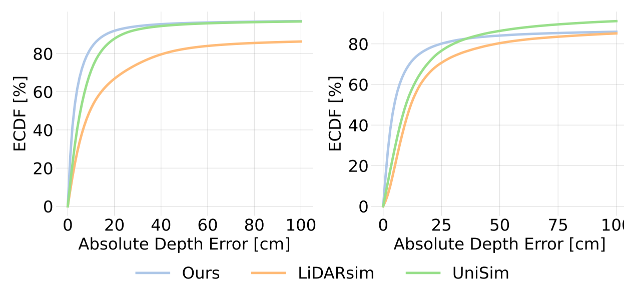

Figure 3. ECDF plots showcasing range errors across all the points (left) and specifically for points associated with dynamic vehicles (right). Our neural fields composition demonstrates superior performance over LiDARsim [27] and UniSim [55], especially in the context of dynamic vehicles.

6.2. LiDAR novel view synthesis evaluation

LiDAR NVS in dynamic scenes. Quantitative comparisons with baseline methods are detailed in Tab. 1. DyNFL notably outperforms LiDARsim [27] and UniSim [55] in range reconstruction. This improvement is largely due to our SDF-based neural scene representation, which incorporates the physical aspects of LiDAR sensing. Additionally,

[*]We re-implement LiDARsim [22] and UniSim [55] as they are not open-sourced.

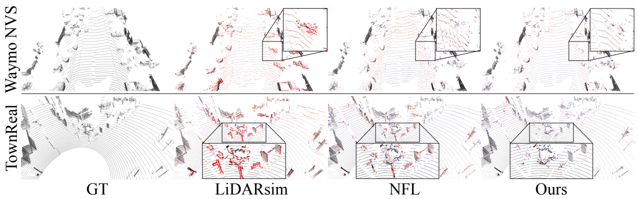

Figure 4. Qualitative results of range estimation. Regions with gross errors (-100 100 cm) are highlighted.

Datasets	MAE ↓	MedAE ↓	CD ↓
TownClean	26.7(-1.5)	0.7(-0.2)	6.7(-0.5)
Waymo Interp	28.3 (0.1)	4.7 (-0.2)	12.5 (-0.1)
Waymo Dynamic	30.8 (-0.3)	3.0 (-0.2)	10.9 (-0.3)

Table 3. Ablation study of volume rendering for active sensing.

Datasets	MAE ↓	MedAE ↓	CD ↓
TownReal	33.9(-3.3)	2.1(-0.0)	10.4(-1.2)
Waymo Interp	28.3 (-0.3)	4.7 (-0.1)	12.5 (-0.3)

Table 4. Ablation study of the surface points' SDF regularisation.

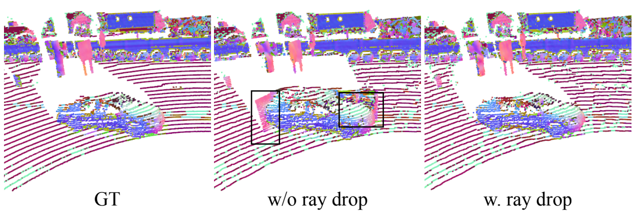

Figure 5. Qualitative results on *Waymo Dynamic* dataset. Our model equipped with a ray drop module effectively composites multiple neural fields, re-simulating LiDAR scans of high quality.

our method employs a ray drop test when rendering multiple neural fields, leading to a more accurate reconstruction of dynamic vehicles, as evidenced in Fig. 2 and further supported by the data in Fig. 3.

LiDAR NVS in static scenes. In addition to dynamic scenes, we evaluate DyNFL against baseline methods in static scenarios, with the results detailed in Tab. 2 and Fig. 4. DyNFL excels in reconstructing geometry in most cases. A key observation is its superior performance in reconstructing planar regions (*e.g.* the ground shown in Fig. 4), especially when compared to NFL [19], which also uses a neural field for surface representation. This improvement is largely due to the enhanced surface regularizations provided by our advanced SDF-based surface modeling approach.

6.3. Ablation study

SDF-based volume rendering for active sensing. We begin by assessing the efficacy of our SDF-based volume rendering for active sensor, the results are shown in Tab. 3. When compared to our baseline that uses the SDF-based volume rendering for passive sensing, DyNFL demonstrates enhanced performance in both synthetic (*TownClean*) and real-world (*Waymo Interp* and *Waymo Dynamic*) datasets, indicating the importance of incorporating the physical sensing process of LiDAR in addressing the inverse problem.

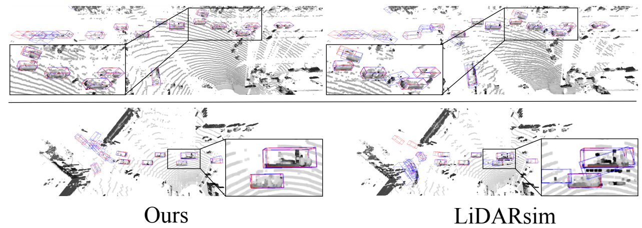

Figure 6. Object detection results on *Waymo Dynamic* dataset. The ground truth and predicted bounding boxes are marked in red and blue, respectively.

	GT	Ours			LiDARSim[27]		
Threshold	AP↑	AP↑	Agg.↑	Dyn. Agg.↑	AP↑	Agg.↑	Dyn. Agg.↑
IoU>0.7	0.85	0.86	**0.77**	**0.71**	**0.90**	0.76	0.68
IoU>0.5	**0.98**	0.96	**0.87**	**0.76**	0.95	0.86	**0.76**

Table 5. Object detection results on *Waymo Dyanmic* datasets.

	Vehicle			Background		
Method	Recall ↑	Precision ↑	IoU ↑	Recall ↑	Precision ↑	IoU ↑
i-NGP [29]	91.8	83.6	78.1	97.9	99.2	97.1
DS-NeRF [8]	89.3	84.8	77.3	98.1	98.8	97.0
URF [36]	86.9	79.8	72.0	97.7	98.5	96.2
Lidarsim [27]	89.6	68.9	64.0	94.5	98.9	93.5
NFL [19]	**94.5**	84.8	80.9	97.8	**99.4**	**97.3**
Ours	90.5	**88.4**	**81.1**	**98.5**	98.7	**97.3**

Table 6. Semantic segmentation results on *Waymo NVS* dataset.

Neural fields composition. To validate the efficacy of our two-stage neural field composition approach, we compare it with an alternative approach utilized in UniSim [55]. The results are shown in Tab. 1. UniSim [55] blends different neural fields by sampling points from all intersected neural fields, followed by a single evaluation of volume rendering to produce the final LiDAR scan. In contrast, our method independently renders from each intersecting neural field first, and then combines these measurements into a final measurement using a ray drop test (*cf*. Fig. 5). This approach leads to a notable improvement in geometry reconstruction over UniSim [55], exemplified by our method halving the Median Absolute Error (MedAE) across all points. This enhancement is even more evident when focusing solely on points related to dynamic vehicles (*cf*. Fig. 3).

Surface points' SDF constraint. We examine the importance of the surface points' SDF constraint discussed in

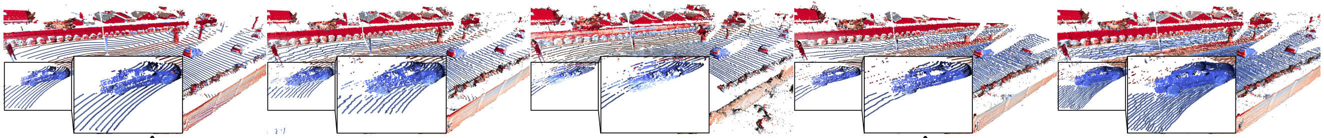

Figure 7. LiDAR novel view synthesis by changing sensor elevation angle (θ), poses (x, y, z) and number of beams on *Waymo Dynamic* dataset. The points are color-coded by the intensity values (0 0.25).

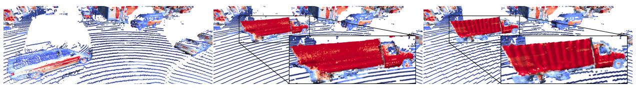

Figure 8. Qualitative results of object removal and insertion. DyNFL seamlessly inserts the neural asset (truck) into a new scene attributed to our superior compositional rendering scheme. In contrast, UniSim [55] struggles to accurately model geometry.

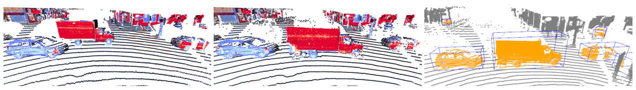

Figure 9. Qualitative results of object trajectory manipulation. The truck can be successfully detected after manipulation, indicating high-realism LiDAR re-simulation achieved by DyNFL.

Sec. 5 on *Town Real* and *Waymo Interp* datasets. The results shown in Tab. 4 suggest that our method yields improved geometry reconstruction quality by additionally enforcing LiDAR points to have zero SDF values.

6.4. Auxiliary task evaluations

To assess the fidelity of our neural re-simulation and gauge the domain gap between re-simulated and real scans, we evaluate their applicability in two downstream tasks: object detection and semantic segmentation.

Object detection. We utilize the pre-trained FSDv2 [10] model for object detection and conduct evaluations on the re-simulated LiDAR scans within the *Waymo Dynamic* dataset. Our results are compared against those from LiDARsim [27], with the findings detailed in Tab. 5 and Fig. 6. Notably, DyNFL exhibits a more substantial detection agreement with the predictions on real LiDAR scans. This indicates a higher fidelity in our re-simulations and a reduced domain gap relative to actual scans.

Semantic segmentation. For semantic segmentation, we use the pre-trained SPVNAS model [43], with the results presented in Tab. 6. DyNFL improves over baseline methods according to most evaluation metrics, underscoring the realism of our re-simulated LiDAR scans.

6.5. Scene editing

Beyond LiDAR novel view synthesis by adjusting the sensor configurations (*cf*. Fig. 7), we additionally demonstrate the practicality of our compositional neural fields approach through two scene editing applications.

Insert object from one scene into another. Our explicit neural scene de-composition and flexible composition technique enable seamless insertion and removal of neural assets across scenes. As demonstrated in Fig. 8, we are able to replace a car from one scene with a truck from another scene, achieving accurate reconstruction of both geometry and intensity. In contrast, UniSim [55] struggles to preserve high quality geometry. This highlights the significant potential of our approach in generating diverse and realistic LiDAR scans for autonomous driving scenarios.

Manipulate the trajectory of dynamic objects. DyNFL also facilitates the manipulation of moving objects' trajectories by simply adjusting their relative poses to the canonical bounding box. Representative results are shown in Fig. 9. The high realism of our re-simulation is also indicated by the successful detection of inserted virtual objects.

7. Limitations and future work

We present DyNFL, a compositional neural fields approach for LiDAR re-simulation. Our method excels previous art in both static and dynamic scenes, offering powerful scene editing capabilities that open up opportunities for generating diverse and high-quality scenes, to evaluate an autonomy system trained only on real data in closed-loop.

Despite achieving the state-of-the-art performance, there are still limitations we aim to address in future work. Firstly, DyNFL faces challenges in view synthesis of dynamic vehicles from unseen angles. This difficulty arises from the complexity of creating an a-priori model that can accurately complete unseen regions and simulate point cloud noise, ray drops patterns etc. Secondly, our method currently relies on object detection and tracking annotations, and its performance may be compromised when given inaccurate labels. Overcoming this dependency, exploring 4D representations while retaining scene editing flexibility, stands out as a crucial challenge for future research.

References

[1] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. HyperReel: High-fidelity 6-DoF video with ray-conditioned sampling. In *CVPR*, 2023. 2

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021. 2

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 2

[4] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The Lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *CVPR*, 2018. 5

[5] Lucas Caccia, Herke Van Hoof, Aaron Courville, and Joelle Pineau. Deep generative modeling of lidar data. In *IROS*. IEEE, 2019. 3

[6] MingFang Chang, Akash Sharma, Michael Kaess, and Simon Lucey. Neural radiance field with LiDAR maps. In *ICCV*, 2023. 3

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensorRF: Tensorial radiance fields. In *ECCV*. Springer, 2022. 2

[8] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *CVPR*, 2022. 6, 7

[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*. PMLR, 2017. 1, 3

[10] Lue Fan, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. FSD V2: Improving fully sparse 3d object detection with virtual voxels. *arXiv preprint arXiv:2308.03755*, 2023. 8

[11] Jin Fang, Dingfu Zhou, Feilong Yan, Tongtong Zhao, Feihu Zhang, Yu Ma, Liang Wang, and Ruigang Yang. Augmented lidar simulator for autonomous driving. *IEEE Robotics and Automation Letters*, 5(2):1931–1938, 2020. 3

[12] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2

[13] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3d scene flow. *CVPR 2021*, 2021. 3

[14] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of Machine Learning and Systems 2020*, pages 3569–3579. 2020. 5

[15] Benoît Guillard, Sai Vemprala, Jayesh K Gupta, Ondrej Miksik, Vibhav Vineet, Pascal Fua, and Ashish Kapoor. Learning to simulate realistic lidars. In *IROS*, pages 8173–8180. IEEE, 2022. 1, 3

[16] Martin Hahner, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Fog simulation on real LiDAR point clouds for 3d object detection in adverse weather. In *CVPR*, 2021. 4

[17] Martin Hahner, Christos Sakaridis, Mario Bijelic, Felix Heide, Fisher Yu, Dengxin Dai, and Luc Van Gool. LiDAR snowfall simulation for robust 3d object detection. In *CVPR*, 2022. 4

[18] Shengyu Huang, Zan Gojcic, Jiahui Huang, and Konrad Schindler Andreas Wieser. Dynamic 3d scene analysis by point cloud accumulation. In *ECCV*, 2022. 3

[19] Shengyu Huang, Zan Gojcic, Zian Wang, Francis Williams, Yoni Kasten, Sanja Fidler, Konrad Schindler, and Or Litany. Neural lidar fields for novel view synthesis. In *ICCV*, 2023. 1, 3, 4, 5, 6, 7

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 2

[21] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *CVPR*, 2022. 2

[22] Seongjo Lee, Dahyeon Kang, Seoungjae Cho, Sungdae Sim, Yong Woon Park, Kyhyun Um, and Kyungeun Cho. Lidar simulation method for low-cost repetitive validation. In *Advances in Computer Science and Ubiquitous Computing: CSA & CUTE*, pages 237–242. Springer, 2015. 6

[23] Chenqi Li, Yuan Ren, and Bingbing Liu. Pcgen: Point cloud generator for lidar simulation. In *ICRA*, pages 11676–11682. IEEE, 2023. 3

[24] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 2

[25] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023. 2, 3, 5

[26] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *CVPR*, 2023. 2

[27] 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 *CVPR*, 2020. 1, 3, 6, 7, 8

[28] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NerF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2

[29] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 2, 3, 6, 7

[30] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, 2020. 2

[31] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *ICCV*, 2021. 2

[32] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *CVPR*, 2021. 1, 2

[33] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, 2021. 2

[34] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. HyperNeRF: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Transactions on Graphics*, 40(6), 2021.

[35] 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. 2

[36] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. *arXiv preprint arXiv:2111.14643*, 2021. 7

[37] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *CVPR*, 2022. 3, 6

[38] Sara Fridovich-Keil and Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 2

[39] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *International Conference on Field and Service Robotics*, pages 621–635. Springer, 2018. 3

[40] Ken Shoemake. Animating rotation with quaternion curves. In *Conference on Computer Graphics and Interactive Techniques*, page 245–254. Association for Computing Machinery, 1985. 5

[41] Jiaming Sun, Xi Chen, Qianqian Wang, Zhengqi Li, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. Neural 3d reconstruction in the wild. In *ACM SIGGRAPH*, 2022. 2

[42] 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 *CVPR*, 2020. 5

[43] Haotian* Tang, Zhijian* Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *ECCV*, 2020. 8

[44] Tang Tao, Longfei Gao, Guangrun Wang, Peng Chen, Dayang Hao, Xiaodan Liang, Mathieu Salzmann, and Kaicheng Yu. Lidar-nerf: Novel lidar view synthesis via neural radiance fields. *arXiv preprint arXiv:2304.10406*, 2023. 1

[45] Haithem Turki, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. Suds: Scalable urban dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12375–12385, 2023. 2

[46] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *CVPR*, 2022. 2

[47] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 2, 3, 4

[48] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Hf-neus: Improved surface reconstruction using high-frequency details. In *NeurIPS*, 2022.

[49] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *ICCV*, 2023. 2, 3

[50] Zian Wang, Tianchang Shen, Jun Gao, Shengyu Huang, Jacob Munkberg, Jon Hasselgren, Zan Gojcic, Wenzheng Chen, and Sanja Fidler. Neural fields meet explicit geometric representations for inverse rendering of urban scenes. In *CVPR*, 2023. 3

[51] Tong Wu, Jiaqi Wang, Xingang Pan, Xudong Xu, Christian Theobalt, Ziwei Liu, and Dahua Lin. Voxurf: Voxel-based efficient and accurate neural surface reconstruction. *arXiv preprint arXiv:2208.12697*, 2022. 3

[52] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. Dˆ 2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. *Advances in Neural Information Processing Systems*, 35:32653–32666, 2022. 2

[53] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, pages 641–676, 2022. 2

[54] Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, and Yue Wang. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. *arXiv preprint arXiv:2311.02077*, 2023. 2

[55] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *CVPR*, 2023. 1, 2, 3, 6, 7, 8

[56] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021. 2

[57] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. MonoSDF: Exploring monocular geometric cues for neural implicit surface reconstruction. In *NeurIPS*, 2022. 2

[58] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *CVPR*, 2021. 2

[59] Junge Zhang, Feihu Zhang, Shaochen Kuang, and Li Zhang. Nerf-lidar: Generating realistic lidar point clouds with neural radiance fields. *arXiv preprint arXiv:2304.14811*, 2023. 1

[60] Xingxing Zuo, Nan Yang, Nathaniel Merrill, Binbin Xu, and Stefan Leutenegger. Incremental dense reconstruction from monocular video with guided sparse feature volume fusion. *IEEE Robotics and Automation Letters*, 2023. 2

[61] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *ECCV*. Springer, 2022. 3