DISS. ETH NO. 27078

Point Cloud to Pose, Pose to Point Cloud

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

Shengyu Huang

MSc in Geomatik, ETH Zurich

born on 17.11.1995

citizen of China

accepted on the recommendation of

Prof. Dr. Konrad Schindler

Prof. Dr. Andreas Wieser

Prof. Dr. Or Litany

Prof. Dr. Simon Lucey

2024

Abstract

placeholder for abstract

Kurzfassung

german text

german text

Acknowledgements

Thank you

Konrad, Andreas, Zan, Iro, Or, Sanja, Federico Michael, Michael, Keisuke, Titas, Lukas

Jiahui, Zian

Mikhail, Anton, Nico, Nikolai, Riccardo, Rodrigo, Alex, Nando, Ozgur, Manu, Corrine, Stefano, Bingxin, Binbin, Yujia, Torben,

Liyuan, Yuru, Hanfeng, Tao, Yuanwen,

Jixuan, Ye, Jimeng, Yikui, Lukas.

Lei Ke

Contents

1 Introduction

paper dissertations require an introduction which includes the overriding research question, the methodology used and the relevance of the thesis to science and economy the conclusion should merge the results of the publications and include suggestions for ongoing research

1.1. Outline & Contributions

The work presented in this thesis comprises articles published to journals and conferences of the fields of Computer Vision and Experimental Fluid Dynamics.

1.1.1. Predator

In Chapter ...

Contributions.

- A warping-based variational optical flow approach is adapted to volumetric particle flow estimation. To account also for larger displacements between time steps, a coarse-to-fine pyramid scheme is implemented. An efficient primal-dual optimization approach is used to minimize the energy, which is formulated as a saddle-point problem.
- The energy formulation is derived from the stationary Stokes equations, which model the incompressibility and viscosity of the fluid. Both, a hard constraint and an alternative soft constraint variant on the flow divergence are integrated into the variational framework. The Euler-Lagrange equations of the energy reveal that the viscosity of the fluid can be accounted for via quadratic regularization of the flow gradient.
- To handle also large measurement volumes, a semi-dense formulation is introduced. It takes into account the individual requirements of particle reconstruction and flow estimation. While the data term is evaluated at full resolution, the flow field is regularized at a lower spatial resolution.
- Extensive evaluations are carried out for different data terms and regularizers. Further, ablation studies on the matching window size, seeding density and stepsize of the semi-dense formulation are performed, justifying the chosen parameters and the effectiveness of the presented approach.

1.2. Relevance to Science and Economy

xxx

Open Source.. xxx

Scientific Recognition.. xxx

2 Background

placeholder

2.1. Point cloud register

3 PREDATOR: Registration of 3D Point Clouds with Low Overlap

Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, Konrad Schindler
IEEE Computer Vision and Pattern Recognition, 2021

(Author version; for typeset version please refer to the original conference paper.)

Abstract

We introduce PREDATOR, a model for pairwise **p**oint-cloud **re**gistration with **d**eep **at**tention to the **o**verlap **r**egion. Different from previous work, our model is specifically designed to handle (also) point-cloud pairs with low overlap. Its key novelty is an overlap-attention block for early information exchange between the latent encodings of the two point clouds. In this way the subsequent decoding of the latent representations into per-point features is conditioned on the respective other point cloud, and thus can predict which points are not only salient, but also lie in the overlap region between the two point clouds. The ability to focus on points that are relevant for matching greatly improves performance: PREDATOR raises the rate of successful registrations by more than 15 percent points in the low-overlap scenario, and also sets a new state of the art for the *3DMatch* benchmark with 90.6% registration recall. [Code release]

a) Input point clouds b) Inferred overlap region c) Estimated registration

Figure 3.1.: PREDATOR is designed to focus attention on the overlap region, and to prefer salient points in that region, so as to enable robust registration in spite of low overlap.

3.1. Introduction

Recent work has made substantial progress in fully automatic, 3D feature-based point cloud registration. At first glance, benchmarks like *3DMatch* Zeng et al. (2017) appear to be saturated, with multiple state-of-the-art (SoTA) methods Gojcic et al. (2019); Choy et al. (2019b); Bai et al. (2020) reaching nearly 95% feature matching recall and successfully registering >80% of all scan pairs. One may get the impression that the registration problem is solved—but this is actually not the case. We argue that the high success rates are a consequence of lenient evaluation protocols. We have been making our task too easy: existing literature and benchmarks Choi et al. (2015); Zeng et al. (2017); Khoury et al. (2017) consider only pairs of point clouds with ≥30% overlap to measure performance. Yet, the low-overlap regime is very relevant for practical applications. On the one hand, it may be difficult to ensure high overlap, for instance when moving along narrow corridors, or when closing loops in the presence of occlusions (densely built-up areas, forest, etc.). On the other hand, data acquisition is often costly, so practitioners aim for a low number of scans with only the necessary overlap Yang et al. (2019, 2020).

Driven by the evaluation protocol, the high-overlap scenario became the focus of research, whereas the more challenging low-overlap examples were largely neglected (*c.f.* Fig. 3.1). Consequently, the registration performance of even the best known methods deteriorates rapidly when the overlap between the two point clouds falls below 30%, see Fig. 3.2. Human operators, in contrast, can still register such low overlap point clouds without much effort.

This discrepancy is the starting point of the present work. To study its reasons, we have constructed a low-overlap dataset *3DLoMatch* from scans of the popular *3DMatch* benchmark, and have analysed the individual modules/steps of the registration pipeline (Fig. 3.2). It turns out that the effective receptive field of fully convolutional feature point descriptors Choy et al. (2019b); Bai et al. (2020) is local enough and the descriptors are hardly corrupted by non-overlapping parts of the scans. Rather than coming up with yet another way to learn better

*First two authors contributed equally to this work.

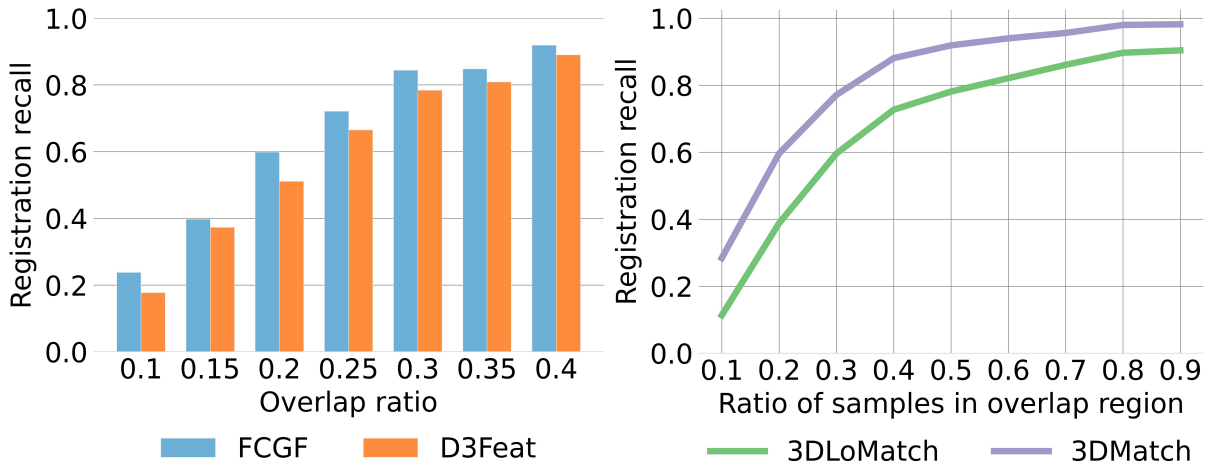


Figure 3.2.: Registration with SoTA methods deteriorates rapidly for pairs with <30% overlap (*left*). By increasing the fraction of points sampled in the overlap region, many failures can be avoided as shown here for FCGF Choy et al. (2019b) (*right*).

descriptors, the key to registering low overlap point clouds is *learning where to sample feature points*. A large performance boost can be achieved if the feature points are predominantly sampled from the overlapping portions of the scans (Fig. 3.2, right).

We follow this path and introduce PREDATOR, a neural architecture for pairwise 3D point cloud registration that learns to detect the overlap region between two unregistered scans, and to focus on that region when sampling feature points. The main contributions of our work are:

- an analysis why existing registration pipelines break down in the low-overlap regime
- a novel *overlap attention* block that allows for early information exchange between the two point clouds and focuses the subsequent steps on the overlap region
- a scheme to refine the feature point descriptors, by conditioning them also on the respective other point cloud
- a novel loss function to train *matchability* scores, which help to sample better and more repeatable interest points

Moreover, we make available the *3DLoMatch* dataset, containing the previously ignored scan pairs of *3DMatch* that have low (10-30%) overlap. In our experiments, PREDATOR greatly outperforms existing methods in the low-overlap regime, increasing registration recall by >15 percent points. It also sets a new state of the art on the *3DMatch* benchmark, reaching a registration recall of >90%.

3.2. Related work

We start this related-work section by reviewing the individual components of the traditional point cloud registration pipelines, before proceeding to newer, end-to-end point-cloud registration algorithms. Finally, we briefly cover recent advances in using contextual information to guide and robustify feature extraction and matching.

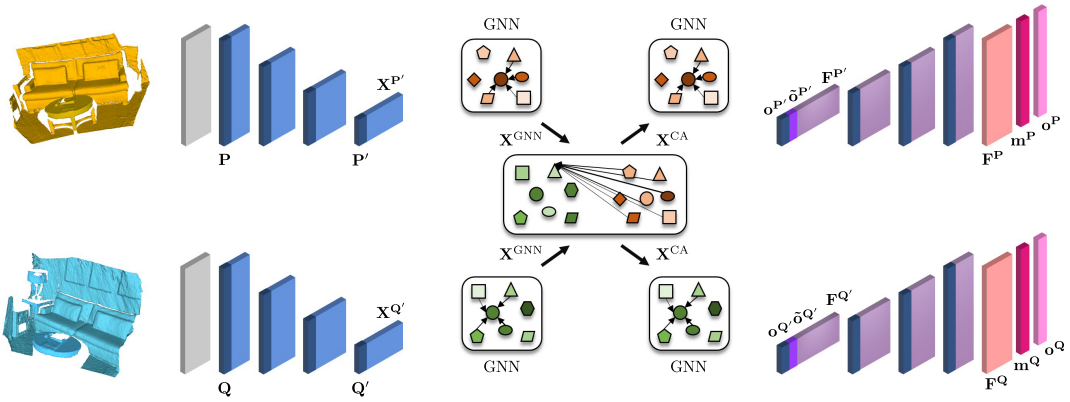


Figure 3.3.: Network architecture of PREDATOR. Voxel-gridded point clouds $\mathbf{P}$ and $\mathbf{Q}$ are fed to the encoder, which extracts the superpoints $\mathbf{P}'$ and $\mathbf{Q}'$ and their latent features $\mathbf{X}^{\mathbf{P}'}$, $\mathbf{X}^{\mathbf{Q}'}$. The overlap-attention module updates the features with co-contextual information in a series of self- (GNN) and cross-attention (CA) blocks, and projects them to overlap $\mathbf{o}^{\mathbf{P}'}$, $\mathbf{o}^{\mathbf{Q}'}$ and cross-overlap $\tilde{\mathbf{o}}^{\mathbf{P}'}$, $\tilde{\mathbf{o}}^{\mathbf{Q}'}$ scores. Finally, the decoder transforms the conditioned features and overlap scores to per-point feature descriptors $\mathbf{F}^{\mathbf{P}}$, $\mathbf{F}^{\mathbf{Q}}$, overlap scores $\mathbf{o}^{\mathbf{P}}$, $\mathbf{o}^{\mathbf{Q}}$, and matchability scores $\mathbf{m}^{\mathbf{P}}$, $\mathbf{m}^{\mathbf{Q}}$.

Local 3D feature descriptors. Early local descriptors for point clouds Johnson and Hebert (1999); Rusu et al. (2008, 2009); Tombari et al. (2010b,a) aimed to characterise the local geometry by using hand-crafted features. While often lacking robustness against clutter and occlusions, they have long been a default choice for downstream tasks because they naturally generalise across datasets Guo et al. (2014). In the last years, learned 3D feature descriptors have taken over and now routinely outperform their hand-crafted counterparts.

The pioneering 3DMatch method Zeng et al. (2017) is based on a Siamese 3D CNN that extracts local feature descriptors from a signed distance function embedding. Others Khoury et al. (2017); Gojcic et al. (2018) first extract hand-crafted features, then map them to a compact representation using multi-layer perceptrons. PPFNet Deng et al. (2018b), and its self-supervised version PPF-FoldNet Deng et al. (2018a), combine point pair features with a PointNet Qi et al. (2017) architecture to extract descriptors that are aware of the global context. To alleviate artefacts caused by noise and voxelisation, Gojcic et al. (2019) proposed to use a smoothed density voxel grid as input to a 3D CNN. These early works achieved strong performance, but still operate on individual local patches, which greatly increases the computational cost and limits the receptive field to a predefined size.

Fully convolutional architectures Long et al. (2015) that enable dense feature computation over the whole input in a single forward pass DeTone et al. (2018); Dusmanu et al. (2019); Revaud et al. (2019) have been adopted to design faster 3D feature descriptors. Building on sparse convolutions Choy et al. (2019a), FCGF Choy et al. (2019b) achieves a performance similar to the best patch-based descriptors Gojcic et al. (2019), while being orders of magnitude faster. D3Feat Bai et al. (2020) complements a fully convolutional feature descriptor with an salient point detector.

Interest point sampling. The classic principle to sample salient rather than random points has also found its way into learned 2D DeTone et al. (2018); Dusmanu et al. (2019); Revaud et al.

(2019); Wiles et al. (2020) and 3D Yew and Lee (2018); Bai et al. (2020); Lu et al. (2020) local feature extraction. All these methods implicitly assume that the saliency of a point fully determines its utility for downstream tasks. Here, we take a step back and argue that, while saliency is desirable for an interest point, it is not sufficient on its own. Indeed, in order to contribute to registration a point should not only be salient, but must also lie in the region where the two point clouds overlap—an essential property that, surprisingly, has largely been neglected thus far.

Deep point-cloud registration. Instead of combining learned feature descriptors with some off-the-shelf robust optimization at inference time, a parallel stream of work aims to embed the differentiable pose estimation into the learning pipeline. PointNetLK Aoki et al. (2019) combines a PointNet-based global feature descriptor Qi et al. (2017) with a Lucas/Kanade-like optimization algorithm Lucas and Kanade (1981) and estimates the relative transformation in an iterative fashion. DCP Wang and Solomon (2019a) use a DGCNN network Wang et al. (2019b) to extract local features and computes soft correspondences before using the Kabsch algorithm to estimate the transformation parameters. To relax the need for strict one-to-one correspondence, DCP was later extended to PRNet Wang and Solomon (2019b), which includes a *keypoint* detection step and allows for partial correspondence. Instead of simply using soft correspondences, Yew and Lee (2020) update the similarity matrix with a differentiable Sinkhorn layer Sinkhorn (1964). Similar to other methods, the weighted Kabsch algorithmArun et al. (1987) is used to estimate the transformation parameters. Finally, Gojcic et al. (2020a); Choy et al. (2020); Pais et al. (2020) complement a learned feature descriptor with an outlier filtering network, which infers the correspondence weights for later use in the weighted Kabsch algorithm.

Contextual information. In the traditional pipeline, feature extraction is done independently per point cloud. Information is only communicated when computing pairwise similarities, although aggregating contextual information at an earlier stage could provide additional cues to robustify the descriptors and guide the matching step.

In 2D feature learning, D2D-Net Wiles et al. (2020) use an attention mechanism in the bottleneck of an encoder-decoder scheme to aggregate the contextual information, which is later used to condition the output of the decoder on the second image. SuperGlue Sarlin et al. (2020) infuses the contextual information into the learned descriptors with a whole series of self- and cross-attention layers, built upon the message-passing GNN Kipf and Welling (2017). Early information mixing was previously also explored in the field of deep point cloud registration, where Wang and Solomon (2019a,b) use a transformer module to extract task-specific 3D features that are reinforced with contextual information.

3.3. Method

PREDATOR is a two-stream encoder-decoder network. Our default implementation uses residual blocks with KPConv-style point convolutions Thomas et al. (2019), but the architecture is agnostic w.r.t. the backbone and can also be implemented with other formulations of 3D convolutions, such as for instance sparse voxel convolutions Choy et al. (2019a) (*c.f*. Appendix).

As illustrated in Fig. 3.3, the architecture of PREDATOR can be decomposed into three main modules:

1. encoding of the two point clouds into smaller sets of superpoints and associated latent feature encodings, with shared weights (Sec. 3.3.2);
2. the overlap attention module (in the bottleneck) that extracts co-contextual information between the feature encodings of the two point clouds, and assigns each superpoint two overlap scores that quantify how likely the superpoint itself and its soft-correspondence are located in the overlap between the two inputs (Sec. 3.3.3);
3. decoding of the mutually conditioned bottleneck representations to point-wise descriptors as well as refined per-point overlap and matchability scores (Sec. 3.3.4).

Before diving into each component we lay out the basic problem setting and notation in Sec. 3.3.1.

3.3.1. Problem setting

Consider two point clouds $\mathbf{P} = \{\mathbf{p}_i \in \mathbb{R}^3 | i = 1..N\}$, and $\mathbf{Q} = \{\mathbf{q}_i \in \mathbb{R}^3 | i = 1..M\}$. Our goal is to recover a rigid transformation $\mathbf{T}_{\mathbf{P}}^{\mathbf{Q}}$ with parameters $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$ that aligns $\mathbf{P}$ to $\mathbf{Q}$. By a slight abuse of notation we use the same symbols for sets of points and for their corresponding matrices $\mathbf{P} \in \mathbb{R}^{N \times 3}$ and $\mathbf{Q} \in \mathbb{R}^{M \times 3}$.

Obviously $\mathbf{T}_{\mathbf{P}}^{\mathbf{Q}}$ can only ever be determined from the data if $\mathbf{P}$ and $\mathbf{Q}$ have sufficient overlap, meaning that after applying the ground truth transformation $\overline{\mathbf{T}}_{\mathbf{P}}^{\mathbf{Q}}$ the overlap ratio

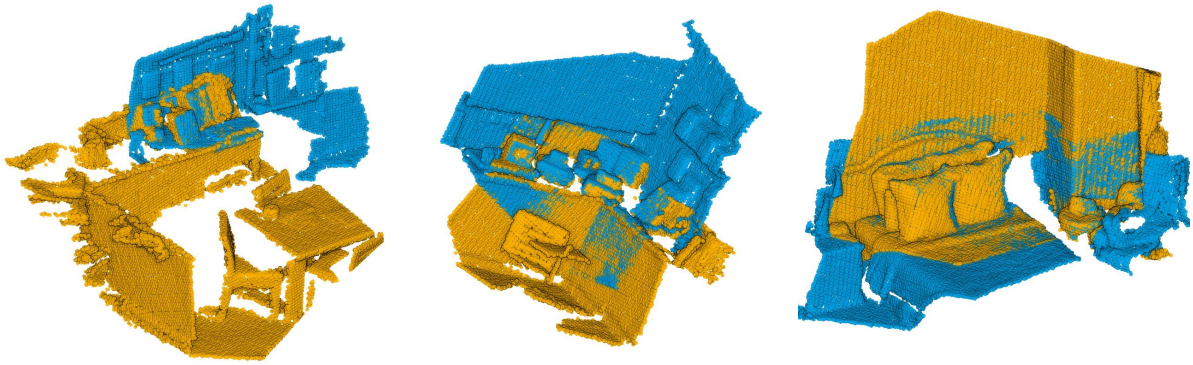
$$\frac{1}{N} \left| \left\{ \| (\overline{\mathbf{T}}_{\mathbf{P}}^{\mathbf{Q}}(\mathbf{p}_i) - \mathsf{NN}(\overline{\mathbf{T}}_{\mathbf{P}}^{\mathbf{Q}}(\mathbf{p}_i), \mathbf{Q}) \|_2 \leq v \right\} \right| > \tau \ , \tag{3.1}$$

where NN denotes the nearest-neighbour operator w.r.t. its second argument, $\| \cdot \|_2$ is the Euclidean norm, $|\cdot|$ is the set cardinality, and v is a tolerance that depends on the point density.[2] Contrary to previous work Zeng et al. (2017); Khoury et al. (2017), where the threshold to even attempt the alignment is typically $\tau > 0.3$, we are interested in low-overlap point clouds with $\tau > 0.1$. Fragments with different overlap ratios are shown in Fig. 3.4.

3.3.2. Encoder

We follow Thomas et al. (2019) and first down-sample raw point clouds with a voxel-grid filter of size V, such that $\mathbf{P}$ and $\mathbf{Q}$ have reasonably uniform point density. In the shared encoder, a series of ResNet-like blocks and strided convolutions aggregate the raw points into *superpoints* $\mathbf{P}' \in \mathbb{R}^{N' \times 3}$ and $\mathbf{Q}' \in \mathbb{R}^{M' \times 3}$ with associated features $\mathbf{X}^{\mathbf{P}'} \in \mathbb{R}^{N' \times b}$ and $\mathbf{X}^{\mathbf{Q}'} \in \mathbb{R}^{M' \times b}$. Note that superpoints correspond to a fixed receptive field, so their number depends on the spatial extent of the input point cloud and may be different for the two inputs.

[2] For efficiency, v is in practice determined after voxel-grid down-sampling of the two point clouds.



a) overlap ratio = 0.1 b) overlap ratio = 0.3 c) overlap ratio = 0.5

Figure 3.4.: Fragments with different overlap ratios. Overlap is computed relative to the source fragment (orange).

3.3.3. Overlap attention module

So far, the features $\mathbf{X}^{\mathbf{P}'}, \mathbf{X}^{\mathbf{Q}'}$ in the bottleneck encode the geometry and context of the two point clouds. But $\mathbf{X}^{\mathbf{P}'}$ has no knowledge of point cloud $\mathbf{Q}$ and vice versa. In order to reason about their respective overlap regions, some cross-talk is necessary. We argue that it makes sense to add that cross-talk at the level of superpoints in the bottleneck, just like a human operator will first get a rough overview of the overall shape to determine likely overlap regions, and only after that identifies precise feature points in those regions.

Graph convolutional neural network. Before connecting the two feature encodings, we first further aggregate and strengthen their contextual relations individually with a graph neural network (GNN) Wang et al. (2019b). In the following, we describe the GNN for point cloud $\mathbf{P}'$. The GNN for $\mathbf{Q}'$ is the same. First, the superpoints in $\mathbf{P}'$ are linked into a graph in Euclidean space with the k-NN method. Let $\mathbf{x}_i \in \mathbb{R}^b$ denote the feature encoding of superpoint $\mathbf{p}'_i$, and $(i,j) \in \mathcal{E}$ the graph edge between superpoints $\mathbf{p}'_i$ and $\mathbf{p}'_j$. The encoder features are then iteratively updated as

$$ {}^{(k+1)}\mathbf{x}_i = \max_{(i,j)\in\mathcal{E}} h_\theta\big(\mathrm{cat}[{}^{(k)}\mathbf{x}_i, {}^{(k)}\mathbf{x}_j - {}^{(k)}\mathbf{x}_i]\big) \ , \tag{3.2} $$

where $h_\theta(\cdot)$ denotes a linear layer followed by instance normalization Ulyanov et al. (2016) and a LeakyReLU activation Maas et al. (2013), $\max(\cdot)$ denotes element-/channel-wise max-pooling, and $\mathrm{cat}[\cdot,\cdot]$ means concatenation. This update is performed twice with separate (not shared) parameters θ, and the final GNN features $\mathbf{x}_i^{\mathrm{GNN}} \in \mathbb{R}^{d_b}$ are obtained as

$$ \mathbf{x}_i^{\mathrm{GNN}} = h_\theta(\mathrm{cat}[{}^{(0)}\mathbf{x}_i, {}^{(1)}\mathbf{x}_i, {}^{(2)}\mathbf{x}_i]) \ . \tag{3.3} $$

Cross-attention block. Knowledge about potential overlap regions can only be gained by mixing information about both point clouds. To this end we adopt a cross-attention block Sarlin et al. (2020) based on the message passing formulation Gilmer et al. (2017). First, each superpoint in $\mathbf{P}'$ is connected to all superpoints in $\mathbf{Q}'$ to form a bipartite graph. Inspired by the Transformer architecture Vaswani et al. (2017), vector-valued queries $\mathbf{s}_i \in \mathbb{R}^b$ are used to

retrieve the values $\mathbf{v}_j \in \mathbb{R}^b$ of other superpoints based on their keys $\mathbf{k}_j \in \mathbb{R}^b$, where

$$\mathbf{k}_j = \mathbf{W}_k \mathbf{x}_j^{\text{GNN}} \quad \mathbf{v}_j = \mathbf{W}_v \mathbf{x}_j^{\text{GNN}} \quad \mathbf{s}_i = \mathbf{W}_s \mathbf{x}_i^{\text{GNN}} \tag{3.4}$$

and $\mathbf{W}_k$, $\mathbf{W}_v$, and $\mathbf{W}_s$ are learnable weight matrices. The messages are computed as weighted averages of the values,

$$\mathbf{m}_{i\leftarrow} = \sum_{j:(i,j)\in\mathcal{E}} a_{ij}\mathbf{v}_j \ , \tag{3.5}$$

with attention weights $a_{ij} = \text{softmax}(\mathbf{s}_i^T \mathbf{k}_j / \sqrt{b})$ Sarlin et al. (2020). I.e., to update a superpoint $\mathbf{p}'_i$ one combines that point's query with the keys and values of all superpoints $\mathbf{q}'_j$. In line with the literature, in practice we use a multi-attention layer with four parallel attention heads Vaswani et al. (2017). The co-contextual features are computed as

$$\mathbf{x}_i^{\text{CA}} = \mathbf{x}_i^{\text{GNN}} + \text{MLP}(\text{cat}[\mathbf{s}_i, \mathbf{m}_{i\leftarrow}]) \ , \tag{3.6}$$

with $\text{MLP}(\cdot)$ denoting a three-layer fully connected network with instance normalization Ulyanov et al. (2016) and ReLU Nair and Hinton (2010) activations after the first two layers. The same cross-attention block is also applied in reverse direction, so that information flows in both directions, $\mathbf{P}' \rightarrow \mathbf{Q}'$ and $\mathbf{Q}' \rightarrow \mathbf{P}'$.

Overlap scores of the bottleneck points. The above update with co-contextual information is done for each superpoint in isolation, without considering the local context within each point cloud. We therefore, explicitly update the local context after the cross-attention block using another GNN that has the same architecture and underlying graph (within-point cloud links) as above, but separate parameters θ. This yields the final latent feature encodings $\mathbf{F}^{\mathbf{P}'} \in \mathbb{R}^{N' \times b}$ and $\mathbf{F}^{\mathbf{Q}'} \in \mathbb{R}^{M' \times b}$, which are now conditioned on the features of the respective other point cloud. Those features are linearly projected to overlap scores $\mathbf{o}^{\mathbf{P}'} \in \mathbb{R}^{N'}$ and $\mathbf{o}^{\mathbf{Q}'} \in \mathbb{R}^{M'}$, which can be interpreted as probabilities that a certain superpoint lies in the overlap region. Additionally, one can compute *soft correspondences* between superpoints and from the correspondence weights predict the *cross-overlap score* of a superpoint $\mathbf{p}'_i$, i.e., the probability that its correspondence in $\mathbf{Q}'$ lies in the overlap region:

$$\tilde{o}_i^{\mathbf{P}'} := \mathbf{w}_i^T \mathbf{o}^{\mathbf{Q}'}, \quad w_{ij} := \text{softmax}\big(\frac{1}{t}\langle \mathbf{f}_i^{\mathbf{P}'}, \mathbf{f}_j^{\mathbf{Q}'} \rangle\big) \ , \tag{3.7}$$

where $\langle \cdot, \cdot \rangle$ is the inner product, and t is the temperature parameter that controls the soft assignment. In the limit $t \rightarrow 0$, Eq. (3.7) converges to hard nearest-neighbour assignment.

3.3.4. Decoder

Our decoder starts from conditioned features $\mathbf{F}^{\mathbf{P}'}$, concatenates them with the overlap scores $\mathbf{o}^{\mathbf{P}'}$, $\tilde{\mathbf{o}}^{\mathbf{P}'}$, and outputs per-point feature descriptors $\mathbf{F}^{\mathbf{P}} \in \mathbb{R}^{N \times 32}$ and refined per-point overlap and matchability scores $\mathbf{o}^{\mathbf{P}}, \mathbf{m}^{\mathbf{P}} \in \mathbb{R}^N$. The matchability can be seen as a "conditional saliency" that quantifies how likely a point is to be matched correctly, given the points (resp. features) in the other point cloud $\mathbf{Q}$.

The decoder architecture combines NN-upsampling with linear layers, and includes skip

connections from the corresponding encoder layers. We deliberately keep the overlap score and the matchability separate to disentangle the reasons why a point is a good/bad candidate for matching: in principle a point can be unambiguously matchable but lie outside the overlap region, or it can lie in the overlap but have an ambiguous descriptor. Empirically, we find that the network learns to predict high matchability mostly for points in the overlap; probably reflecting the fact that the ground truth correspondences used for training, naturally, always lie in the overlap. For further details about the architecture, please refer to Appendix and the source code.

3.3.5. Loss function and training

PREDATOR is trained end-to-end, using three losses w.r.t. ground truth correspondences as supervision.

Circle loss. To supervise the point-wise feature descriptors we follow[3] Bai et al. (2020) and use the circle loss Sun et al. (2020b), a variant of the more common triplet loss. Consider again a pair of overlapping point clouds $\mathbf{P}$ and $\mathbf{Q}$, this time aligned with the ground truth transformation. We start by extracting the points $\mathbf{p}_i \in \mathbf{P}_p \subset \mathbf{P}$ that have at least one (possibly multiple) correspondence in $\mathbf{Q}$, where the set of correspondences $\mathcal{E}_p(\mathbf{p}_i)$ is defined as points in $\mathbf{Q}$ that lie within a radius r_p around $\mathbf{p}_i$. Similarly, all points of $\mathbf{Q}$ outside a (larger) radius r_s form the set of negatives $\mathcal{E}_n(\mathbf{p}_i)$. The circle loss is then computed from n_p points sampled randomly from $\mathbf{P}_p$:

$$\mathcal{L}_c^{\mathbf{P}} = \frac{1}{n_p} \sum_{i=1}^{n_p} \log \Big[1 + \sum_{j \in \mathcal{E}_p} e^{\beta_p^j (d_i^j - \Delta_p)} \cdot \sum_{k \in \mathcal{E}_n} e^{\beta_n^k (\Delta_n - d_i^k)} \Big], \tag{3.8}$$

where $d_i^j = ||\mathbf{f}_{\mathbf{p}_i} - \mathbf{f}_{\mathbf{q}_j}||_2$ denotes distance in feature space, and Δ_n, Δ_p are negative and positive margins, respectively. The weights $\beta_p^j = \gamma(d_i^j - \Delta_p)$ and $\beta_n^k = \gamma(\Delta_n - d_i^k)$ are determined individually for each positive and negative example, using the empirical margins $\Delta_p := 0.1$ and $\Delta_n := 1.4$ with hyper-parameter γ. The reverse loss $\mathcal{L}_c^{\mathbf{Q}}$ is computed in the same way, for a total circle loss $\mathcal{L}_c = \frac{1}{2}(\mathcal{L}_c^{\mathbf{P}} + \mathcal{L}_c^{\mathbf{Q}})$.

Overlap loss. The estimation of the overlap probability is cast as binary classification and supervised using the overlap loss $\mathcal{L}_o = \frac{1}{2}(\mathcal{L}_o^{\mathbf{P}} + \mathcal{L}_o^{\mathbf{Q}})$, where

$$\mathcal{L}_o^{\mathbf{P}} = \frac{1}{|\mathbf{P}|} \sum_{i=1}^{|\mathbf{P}|} \bar{o}_{\mathbf{p}_i} \log(o_{\mathbf{p}_i}) + (1 - \bar{o}_{\mathbf{p}_i}) \log(1 - o_{\mathbf{p}_i}). \tag{3.9}$$

The ground truth label $\bar{o}_{\mathbf{p}_i}$ of point $\mathbf{p}_i$ is defined as

$$\bar{o}_{\mathbf{p}_i} = \begin{cases} 1, & ||\overline{\mathbf{T}}_{\mathbf{P}}^{\mathbf{Q}}(\mathbf{p}_i) - \mathsf{NN}(\overline{\mathbf{T}}_{\mathbf{P}}^{\mathbf{Q}}(\mathbf{p}_i), \mathbf{Q})||_2 < r_o \\ 0, & \text{otherwise} \end{cases}, \tag{3.10}$$

[3] Added to the repository after publication, not mentioned in the paper.

with overlap threshold r_o. The reverse loss $\mathcal{L}_o^{\mathbf{Q}}$ is computed in the same way. The contributions from positive and negative examples are balanced with weights inversely proportional to their relative frequencies.

Matchability loss. Supervising the matchability scores is more difficult, as it is not clear in advance which are the right points to take into account during correspondence search. We follow a simple intuition: good keypoints are those that can be matched successfully at a given point during training, with the current feature descriptors. Hence, we cast the prediction as binary classification and generate the ground truth labels on the fly. Again, we sum the two symmetric losses, $\mathcal{L}_m = \frac{1}{2}(\mathcal{L}_m^{\mathbf{P}} + \mathcal{L}_m^{\mathbf{Q}})$, with

$$\mathcal{L}_m^{\mathbf{P}} = \frac{1}{|\mathbf{P}|} \sum_{i=1}^{|\mathbf{P}|} \overline{m}_{\mathbf{p}_i} \log(m_{\mathbf{p}_i}) + (1 - \overline{m}_{\mathbf{p}_i}) \log(1 - m_{\mathbf{p}_i}), \tag{3.11}$$

where ground truth labels $\overline{m}_{\mathbf{p}_i}$ are computed on the fly via nearest neighbour search $\mathsf{NN}_{\mathbf{F}}(\cdot, \cdot)$ in feature space:

$$\overline{m}_{\mathbf{p}_i} = \begin{cases} 1, & ||\overline{\mathbf{T}}_{\mathbf{P}}^{\mathbf{Q}}(\mathbf{p}_i) - \mathsf{NN}_{\mathbf{F}}(\mathbf{p}_i, \mathbf{Q})||_2 < r_m \\ 0, & \text{otherwise.} \end{cases} \tag{3.12}$$

Implementation and training. PREDATOR is implemented in pytorch and can be trained on a single RTX 3090 GPU. At the start of the training we supervise PREDATOR only with the circle and overlap losses, the matchability loss is added only after few epochs, when the point-wise features are already meaningful (i.e., >30% of interest points can be matched correctly). The three loss terms are weighted equally. For more details, please refer to Appendix.

3.4. Experiments

We evaluate PREDATOR and justify our design choices on real point clouds, using *3DMatch* Zeng et al. (2017) and *3DLoMatch* (§ 3.4.1). Additionally, we compare PREDATOR to direct registration methods on the synthetic, object-centric *ModelNet40* Wu et al. (2015) (§ 3.4.2) and evaluate it on large outdoor scenes using odometryKITTI Geiger et al. (2012) (§ 3.4.3). More details about the datasets and evaluation metrics are available in the Appndix. Qualitative results are shown in Fig. 3.5.

3.4.1. 3DMatch

Dataset. Zeng et al. (2017) is a collection of 62 scenes, from which we use 46 scenes for training, 8 scenes for validation and 8 for testing. Official *3DMatch* dataset considers only scan pairs with >30% overlap. Here, we add its counterpart in which we consider only scan pairs with overlaps between 10 and 30% and call this collection *3DLoMatch*[4].

[4]Due to a bug in the official implementation of the overlap computation for *3DMatch*, a few (<7%) scan pairs are included in both datasets.

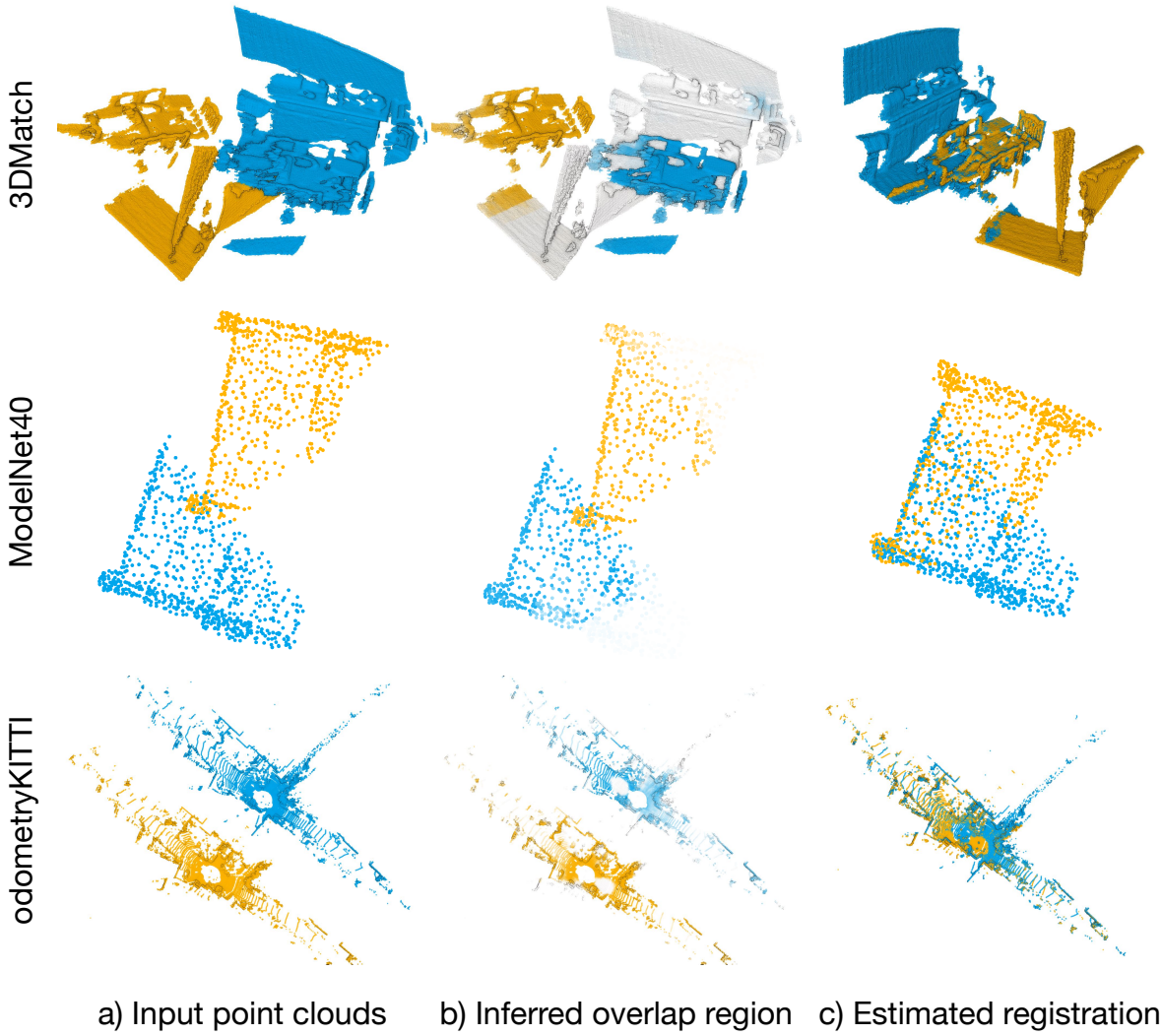


Figure 3.5.: Example results of PREDATOR that succeeds in attending to the overlap region to enable robust registration.

Metrics. Our main metric, corresponding to the actual aim of point cloud registration, is *Registration Recall (RR)*, i.e., the fraction of scan pairs for which the correct transformation parameters are found with RANSAC. Following the literature Zeng et al. (2017); Gojcic et al. (2018); Choy et al. (2019b), we also report *Feature Match Recall (FMR)*, defined as the fraction of pairs that have $>5\%$ "inlier" matches with <10 cm residual under the ground truth transformation (without checking if the transformation can be recovered from those matches), and *Inlier Ratio (IR)*, the fraction of correct correspondences among the putative matches. Additionally, we use empirical cumulative distribution functions (ECDF) to evaluate the relative overlap ratio. At a specific overlap value, the $(1-\text{ECDF})$ curve shows the fraction of fragment pairs that have relative overlap greater or equal to that value.

Relative overlap ratio. We first evaluate if PREDATOR achieves its goal to focus on the overlap. We discard points with a predicted overlap score $\mathbf{o}_i < 0.5$, compute the overlap ratio, and compare it to the one of the original scans. Fig. 3.6 shows that more than half (71%) of the low-overlap pairs are pushed over the 30% threshold that prior works considered the lower

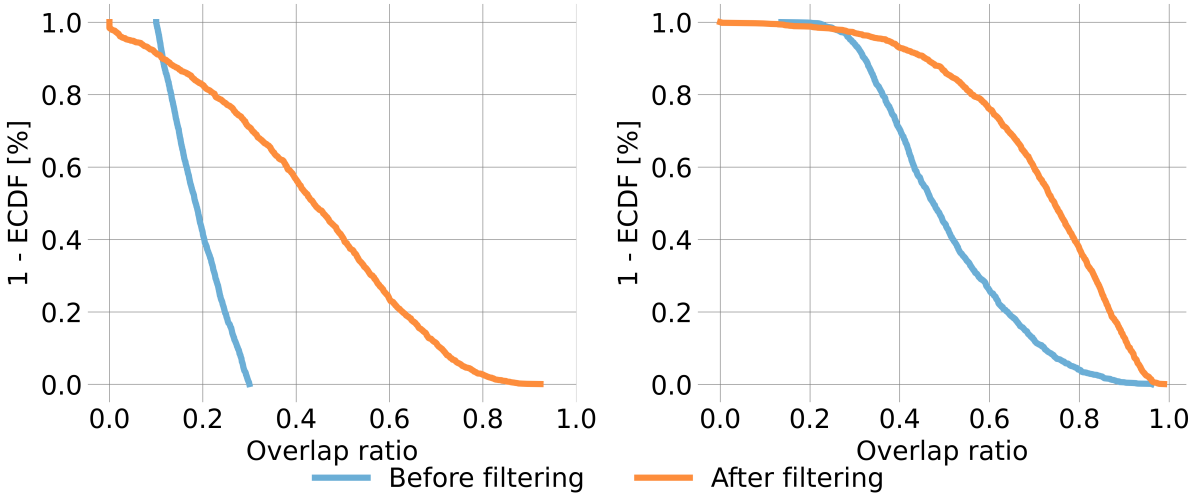


Figure 3.6.: Distribution of the relative overlap ratio before and after filtering the points with the inferred overlap scores, *3DLoMatch* (left) and *3DMatch* (right).

limit for registration. On average, discarding points with low overlap scores almost doubles the overlap in *3DLoMatch* (133% increase). Notably, it also increases the overlap in standard *3DMatch* by, on average, >50%.

Interest point sampling. PREDATOR significantly increases the effective overlap, but does that improve registration performance? To test this we use the product of the overlap scores $\mathbf{o}$ and matchability scores $\mathbf{m}$ to bias interest point sampling. We compare two variants: *top-k (om)*, where we pick the top-k points according to the multiplied scores; and *prob. (om)*, where we instead sample points with probability proportional to the multiplied scores.

For a more comprehensive assessment we follow Bai et al. (2020) and report performance with different numbers of sampled interest points. Tab. 3.1 shows that any of the informed sampling strategies greatly increases the *inlier ratio*, and as a consequence also the *registration recall*. The gains are larger when fewer points are sampled. In the low-overlap regime the inlier ratios more than triple for up to 1000 points. We observe that, as expected, high inlier ratio does not necessarily imply high registration recall: our scores are apparently well calibrated, so that *top-k (om)* indeed finds most inliers, but these are often clustered and too close to each other to reliably estimate the transformation parameters (Fig. 3.7). We thus use the more robust *prob. (om)* sampling, which yields the best *registration recall*. It may be possible to achieve even higher registration recall by combining *top-k (om)* sampling with non-maxima suppression. We leave this for future work.

Comparison to feature-based methods. We compare PREDATOR to recent feature-based registration methods: 3DSN Gojcic et al. (2018), FCGF Choy et al. (2019b) and D3Feat Bai et al. (2020), see Tab. 3.2. Even though PREDATOR can not solve all the cases (*c.f.* Fig. 3.8), it greatly outperforms existing methods on the low-overlap *3DLoMatch* dataset, improving registration recall by 15.5-19.7 percent points (pp) over the closest competitor—variously FCGF or 3DFeat. Moreover, it also consistently reaches the highest registration recall on standard *3DMatch*, showing that its attention to the overlap pays off even for scans with moderately

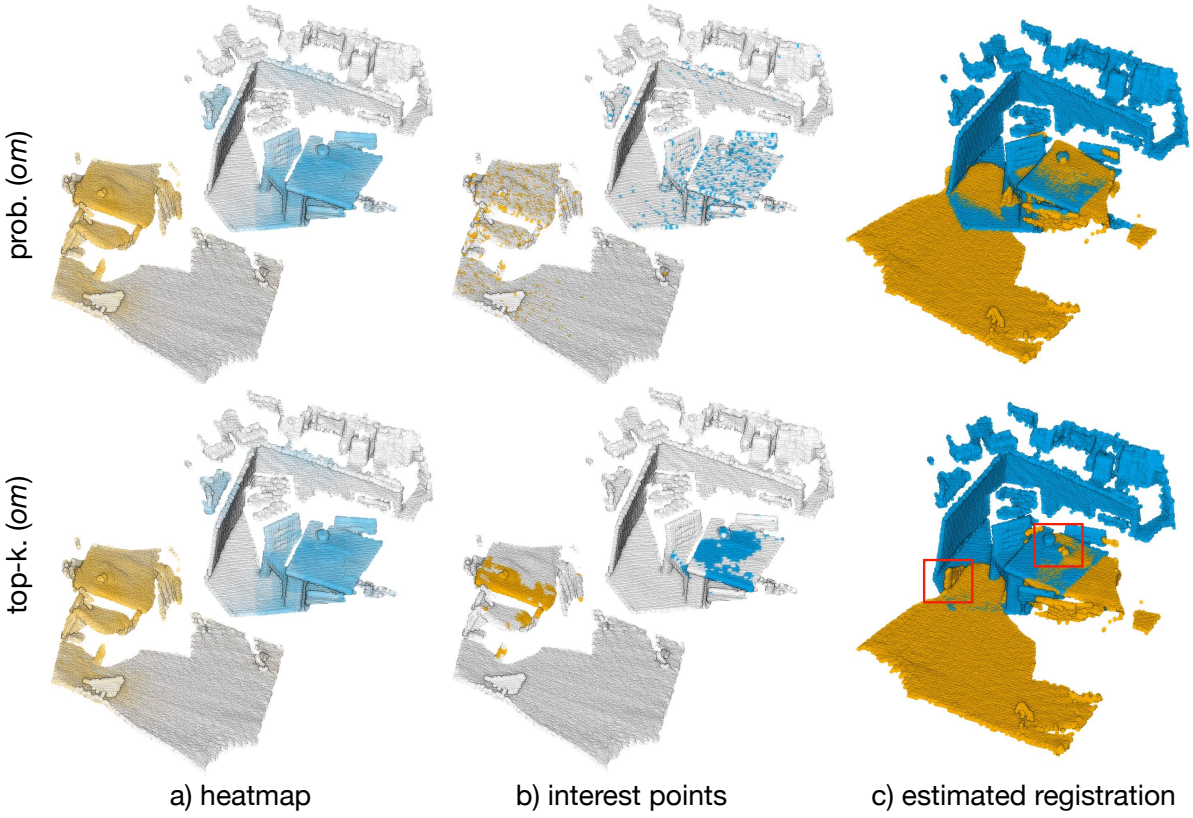


Figure 3.7.: *Top-k (om)* sampling yields clustered interest points, whereas the points obtained with *prob. (om)* sampling are more scattered and thus enable a more robust estimation of the transformation parameters.

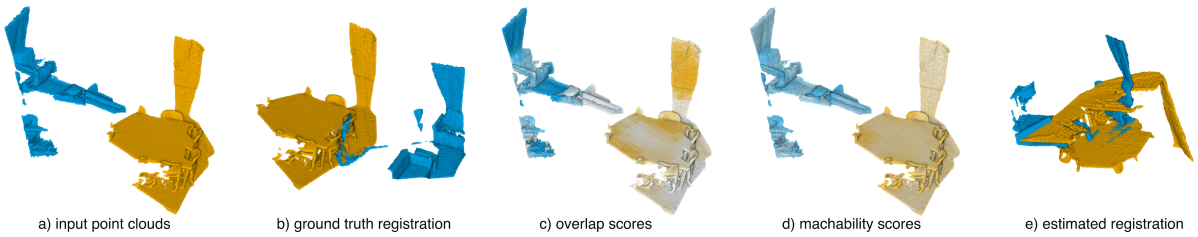


Figure 3.8.: An extreme case where the overlap is insufficient for registration even with the proposed attention mechanism.

large overlap. In line with our motivation, what matters is not so much the choice of descriptors, but finding interest points that lie in the overlap region – especially if that region is small.

Comparison to direct registration methods. We also tried to compare PREDATOR to recent methods for direct registration of partial point clouds. Unfortunately, for both PRNet Wang and Solomon (2019b) and RPM-Net Yew and Lee (2020), training on *3DMatch* failed to converge to reasonable results, as already observed in Choy et al. (2020). It appears that their feature extraction is specifically tuned to synthetic, object-centric point clouds. Thus, in a further attempt we replaced the feature extractor of RPM-Net with FCGF. This brought the registration recall on *3DMatch* to 54.9%, still far from the 85.1% that FCGF features achieve with RANSAC. We conclude that direct pairwise registration is at this point only suitable for geometrically simple

	3DMatch					*3DLoMatch*				
# Samples (k)	5000	2500	1000	500	250	5000	2500	1000	500	250
	Inlier ratio (%)									
rand	51.6	49.5	44.5	38.9	32.1	20.4	19.2	16.8	14.3	11.5
top-k (om)	**68.4**	**73.8**	**77.6**	**78.6**	**78.7**	**33.7**	**39.9**	**44.9**	**47.0**	**47.7**
prob. (om)	58.0	58.4	57.1	54.1	49.3	26.7	28.1	28.3	27.5	25.8
	Registration Recall (%)									
rand	86.0	84.8	84.7	81.7	75.3	43.3	45.3	40.4	35.9	28.0
top-k (om)	88.9	87.4	82.0	75.6	64.0	58.5	57.8	53.1	44.9	35.9
prob. (om)	**89.0**	**89.9**	**90.6**	**88.5**	**86.6**	**59.8**	**61.2**	**62.4**	**60.8**	**58.1**

Table 3.1.: Performance of PREDATOR with different interest point sampling strategies; *om* denotes the product of overlap score and matchability score.

	3DMatch					*3DLoMatch*				
# Samples	5000	2500	1000	500	250	5000	2500	1000	500	250
	Registration Recall (%)									
3DSN Gojcic et al. (2019)	78.4	76.2	71.4	67.6	50.8	33.0	29.0	23.3	17.0	11.0
FCGF Choy et al. (2019b)	85.1	84.7	83.3	81.6	71.4	40.1	41.7	38.2	35.4	26.8
D3Feat Bai et al. (2020)	81.6	84.5	83.4	82.4	77.9	37.2	42.7	46.9	43.8	39.1
PREDATOR	**89.0**	**89.9**	**90.6**	**88.5**	**86.6**	**59.8**	**61.2**	**62.4**	**60.8**	**58.1**

Table 3.2.: Results on the *3DMatch* and *3DLoMatch* datasets.

overlap attention			*3DMatch*			*3DLoMatch*		
ov.	$\times$*ov.*	*cond.*	FMR	IR	RR	FMR	IR	RR
			96.4	39.6	82.6	72.2	14.5	38.9
✓			94.6	38.3	84.1	67.1	14.3	42.8
✓	✓		96.4	50.8	87.7	73.8	20.9	56.5
✓		✓	95.7	52.1	88.0	72.5	21.2	57.5
✓	✓	✓	**96.7**	**58.0**	**89.0**	**78.6**	**26.7**	**59.8**

Table 3.3.: Ablation of the network architecture. *ov.* denotes upsampling the overlap scores; *cond.* denotes conditioning the bottleneck features on the respective other point cloud; $\times$*ov.* denotes upsampling the cross overlap scores.

objects in controlled settings like *ModelNet40*.

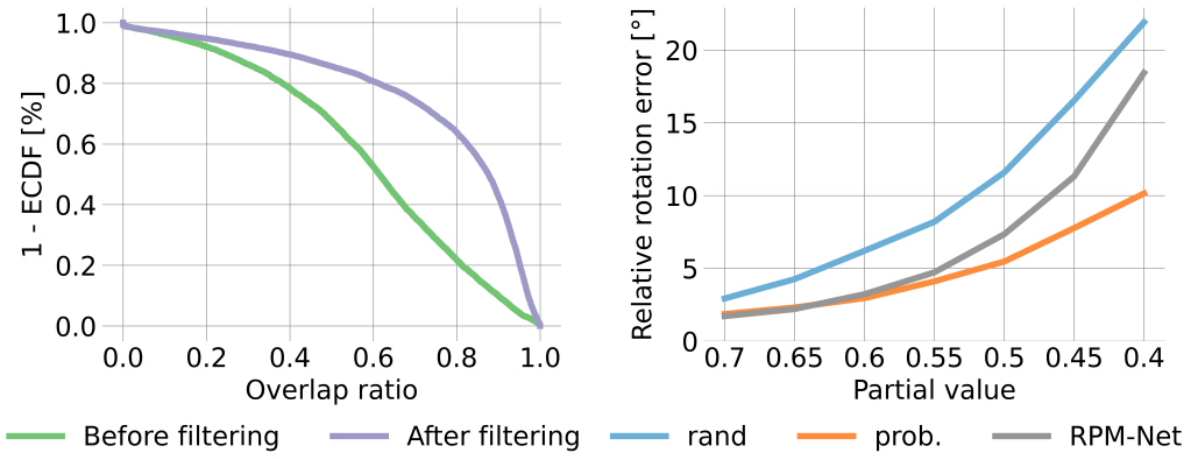


Figure 3.9.: Improved relative overlap ratio after filtering the points with the inferred overlap scores on 8862 *ModelNet* partial scans(*left*). Owing to the improved overlap ratio, PREDATOR is robust to the changes of partial value p_v, while the performance of RPM-Net drops rapidly (*right*). *rand* and *prob.* denote the random and *prob. (om)* biased sampling of 450 interest points, respectively.

Ablations study. We ablate our overlap attention module in Tab. 3.3. We first compare PREDATOR with a baseline model, in which we completely remove the proposed overlap attention module. That baseline, combined with random sampling, achieves the 2$^{\text{nd}}$-highest FMR on both benchmarks, but only reaches 82.6%, respectively 38.9% RR. By adding the overlap scores, RR increases by 1.5, respectively 3.9 pp on *3DMatch* and *3DLoMatch*. Additionally upsampling conditioned feature scores or cross overlap scores further improves performance, especially on *3DLoMatch*. All three parts combined lead to the best overall performance. For further ablation studies, see Appendix.

3.4.2. ModelNet40

Dataset. Wu et al. (2015) contains 12,311 CAD models of man-made objects from 40 different categories. We follow Yew and Lee (2020) to use 5,112 samples for training, 1,202 samples for validation, and 1,266 samples for testing. Partial scans are generated following Yew and Lee (2020). In addition to *ModelNet* which has 73.5% pairwise overlap on average, we generate *ModelLoNet* with lower (53.6%) average overlap. For more details see Appendix.

Metrics. We follow Yew and Lee (2020) and measure the performance using the *Relative Rotation Error (RRE)* (geodesic distance between estimated and GT rotation matrices), the *Relative Translation Error (RTE)* (Euclidean distance between the estimated and GT translations), and the *Chamfer distance (CD)* between the two registered scans.

Relative overlap ratio. We again evaluate if PREDATOR focuses on the overlap region. We extract 8,862 test pairs by varying the completeness of the input point clouds from 70 to 40%. Fig. 3.9 shows that PREDATOR substantially increases the relative overlap and reduces the number of pairs with overlap <70% by more than 40 pp.

	ModelNet			*ModelLoNet*		
Methods	RRE	RTE	CD	RRE	RTE	CD
DCP-v2 Wang and Solomon (2019a)	11.975	0.171	0.0117	16.501	0.300	0.0268
RPM-Net Yew and Lee (2020)	**1.712**	**0.018**	**0.00085**	7.342	**0.124**	**0.0050**
PREDATOR (*rand*)	2.407	0.028	0.00120	10.985	0.175	0.0097
PREDATOR (*prob. (om)*)	1.739	0.019	0.00089	**5.235**	0.132	0.0083

Table 3.4.: Evaluation results on *ModelNet* and *ModelLoNet*. 450 points are sampled for RANSAC with *rand* / *prob.*.

Method	*RTE [cm]* ↓	*RRE [°]* ↓	RR ↑
3DFeat-Net Yew and Lee (2018)	25.9	0.57	96.0
FCGF Choy et al. (2019b)	9.5	0.30	96.6
D3Feat* Bai et al. (2020)	7.2	0.30	**99.8**
PREDATOR (*rand*)	8.8	0.34	**99.8**
PREDATOR (*prob. (om)*)	**6.8**	**0.27**	**99.8**

Table 3.5.: Evaluation of PREDATOR on *odometryKITTI*, following the evaluation protocol employed by D3Feat Bai et al. (2020).

Comparison to direct registration methods. To be able to compare PREDATOR to RPM-Net Yew and Lee (2020) and DCP Wang and Solomon (2019a), we resort to the synthetic, object-centric dataset they were designed for. We failed to train PRNet Wang and Solomon (2019b) due to random crashes of the original code (also observed in Choy et al. (2020)).

Remarkably, PREDATOR can compete with methods specifically tuned for *ModelNet*, and in the low-overlap regime outperforms them in terms of *RRE*, see Tab. 3.4. Moreover, we observe a large boost by sampling points with overlap attention (*prob. (om)*) rather than randomly (*rand*). Fig. 3.9 (right) further underlines the importance of sampling in the overlap: PREDATOR is a lot more robust in the low overlap regime (≈8° lower RRE at completeness 0.4).

3.4.3. odometryKITTI

Dataset. Geiger et al. (2012) contains 11 sequences of LiDAR-scanned outdoor driving scenarios. We follow Choy et al. (2019b) and use sequences 0-5 for training, 6-7 for validation, and 8-10 for testing. In line with Choy et al. (2019b); Bai et al. (2020) we further refine the provided ground truth poses using ICP Besl and McKay (1992) and only use point cloud pairs that are at most 10 m away from each other for evaluation.

Comparision to the SoTAs. We compare PREDATOR to 3DFeat-Net Yew and Lee (2018), FCGF Choy et al. (2019b) and D3Feat* Bai et al. (2020)[5] As shown in Tab. 3.5, PREDATOR

[5]We find that the released D3Feat code fails to reproduce the results in the paper, possible due to hyper-parameter

performs on-par with the SoTA. The results also corroborate the impact of our overlap attention which again outperforms the random sampling baseline.

Computational complexity. With $O(n^2)$ complexity the cross-attention module represents the memory bottleneck of PREDATOR. Furthermore, n cannot be selected freely but results from the interplay of (i) the resolution of the initial voxel grid, (ii) the network architecture (number of strided convolution layers), and (iii) the spatial extent of the scene. Nevertheless, by executing the cross-attention at the *superpoint* level, with greatly reduced n, we are able to apply PREDATOR to large outdoor scans like *odometryKITTI* on a single GPU. For even larger scenes, a simple engineering trick could be to split them into parts, as often done for semantic segmentation.

3.5. Conclusion

We have introduced PREDATOR, a deep model designed for pairwise registration of low-overlap point clouds. The core of the model is an overlap attention module that enables early information exchange between the point clouds' latent encodings, in order to infer which of their points are likely to lie in their overlap region.

There are a number of directions in which PREDATOR could be extended. At present it is tightly coupled to fully convolutional point cloud encoders, and relies on having a reasonable number of *superpoints* in the bottleneck. This could be a limitation in scenarios where the point density is very uneven. It would also be interesting to explore how our overlap-attention module can be integrated into direct point cloud registration methods and other neural architectures that have to handle two inputs with low overlap, e.g. in image matching Sarlin et al. (2020). Finally, registration in the low-overlap regime is challenging and PREDATOR cannot solve all the cases. A user study could provide a better understanding of how PREDATOR compares to human operators.

Acknowledgements. This work was sponsored by the NVIDIA GPU grant.

changes.

3.6. Appendix

In this supplementary material, we first provide rigorous definitions of evaluation metrics (Sec. 3.6.1), then describe the data pre-processing step (Sec. 3.6.2), network architectures (Sec. 3.6.4) and training on individual datasets (Sec. 3.6.3) in more detail. We further provide additional results (Sec. 3.6.5), ablation studies (Sec. 3.6.6) as well as a runtime analysis (Sec. 3.6.7). Finally, we show more visualisations on *3DLoMatch* and *ModelLoNet* benchmarks (Sec. 3.6.8).

3.6.1. Evaluation metrics

The evaluation metrics, which we use to assess model performance in Sec. 4 of the main paper and Sec. 3.6.5 of this supplementary material, are formally defined as follows:

Inlier ratio looks at the set of putative correspondences $(\mathbf{p}, \mathbf{q}) \in \mathcal{K}_{ij}$ found by reciprocal matchingin feature space, and measures what fraction of them is "correct", in the sense that they lie within a threshold $\tau_1 = 10\,\text{cm}$ after registering the two scans with the ground truth transformation $\overline{T}_{\mathbf{P}}^{\mathbf{Q}}$:

$$\text{IR} = \frac{1}{|\mathcal{K}_{ij}|} \sum_{(\mathbf{p},\mathbf{q}) \in \mathcal{K}_{ij}} \left[||\overline{\mathbf{T}}_{\mathbf{P}}^{\mathbf{Q}}(\mathbf{p}) - \mathbf{q}||_2 < \tau_1 \right] , \tag{3.13}$$

with $[\cdot]$ the Iverson bracket.

Feature Match recall (FMR) Deng et al. (2018b) measures the fraction of point cloud pairs for which, based on the number of inlier correspondences, it is *likely* that accurate transformation parameters can be recovered with a robust estimator such as RANSAC. Note that FMR only checks whether the inlier ratio is above a threshold $\tau_2 = 0.05$. It does not test if the transformation can actually be determined from those correspondences, which in practice is not always the case, since their geometric configuration may be (nearly) degenerate, e.g., they might lie very close together or along a straight edge. A single pair of point clouds counts as suitable for registration if

$$IR > \tau_2 \tag{3.14}$$

Registration recall Choi et al. (2015) is the most reliable metric, as it measures end-to-end performance on the actual task of point cloud registration. Specifically, it looks at the set of ground truth correspondences $\mathcal{H}_{ij}^*$ after applying the estimated transformation $T_{\mathbf{P}}^{\mathbf{Q}}$, computes their root mean square error,

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{H}_{ij}^*|} \sum_{(\mathbf{p},\mathbf{q}) \in \mathcal{H}_{ij}^*} ||\mathbf{T}_{\mathbf{P}}^{\mathbf{Q}}(\mathbf{p}) - \mathbf{q}||_2^2} , \tag{3.15}$$

and checks for what fraction of all point pairs $\text{RMSE} < 0.2$. In keeping with the original evaluation script of *3DMatch*, immediately adjacent point clouds are excluded, since they have very high overlap by construction.

Chamfer distance measures the quality of registration on synthetic data. We follow Yew

and Lee (2020) and use the *modified* Chamfer distance metric:

$$\begin{aligned}\tilde{CD}(\mathbf{P},\mathbf{Q}) =& \frac{1}{|\mathbf{P}|}\sum_{\mathbf{p}\in\mathbf{P}}\min_{\mathbf{q}\in\mathbf{Q}_{\text{raw}}}\|\mathbf{T}_{\mathbf{P}}^{\mathbf{Q}}(\mathbf{p})-\mathbf{q}\|_2^2 + \\ & \frac{1}{|\mathbf{Q}|}\sum_{\mathbf{q}\in\mathbf{Q}}\min_{\mathbf{p}\in\mathbf{P}_{\text{raw}}}\|\mathbf{q}-\mathbf{T}_{\mathbf{P}}^{\mathbf{Q}}(\mathbf{p})\|_2^2\end{aligned} \tag{3.16}$$

where $\mathbf{P}_{\text{raw}} \in \mathbb{R}^{2048\times 3}$ and $\mathbf{Q}_{\text{raw}} \in \mathbb{R}^{2048\times 3}$ are *raw* source and target point clouds, $\mathbf{P} \in \mathbb{R}^{717\times 3}$ and $\mathbf{Q} \in \mathbb{R}^{717\times 3}$ are *input* source and target point clouds.

Relative translation and rotation errors (RTE/RRE) measures the deviations from the ground truth pose as:

$$\begin{aligned}\text{RTE} &= \|\mathbf{t}-\bar{\mathbf{t}}\|_2 \\ \text{RRE} &= \arccos\big(\frac{\text{trace}(\mathbf{R}^T\overline{\mathbf{R}})-1}{2}\big)\end{aligned} \tag{3.17}$$

where $\mathbf{R}$ and $\mathbf{t}$ denote the estimated rotation matrix and translation vector, respectively.

Empirical Cumulative Distribution Function (ECDF) measures the distribution of a set of values:

$$\text{ECDF}(x) = \frac{\big|\{o_i < x\}\big|}{\big|O\big|} \tag{3.18}$$

where O is a set of values(ovelap ratios in our case) and $x \in [\min\{O\}, \max\{O\}]$.

3.6.2. Dataset preprocessing

3DMatch. Zeng et al. (2017) is a collection of 62 scenes, combining earlier data from Analysis-by-Synthesis Valentin et al. (2016), 7Scenes Shotton et al. (2013), SUN3D Xiao et al. (2013), RGB-D Scenes v.2 Lai et al. (2014), and Halber *et al*. Halber and Funkhouser (2016). The official benchmark splits the data into 54 scenes for training and 8 for testing. Individual scenes are not only captured in different indoor spaces (e.g., bedrooms, offices, living rooms, restrooms) but also with different depth sensors (e.g., Microsoft Kinect, Structure Sensor, Asus Xtion Pro Live, and Intel RealSense). *3DMatch* provides great diversity and allows our model to generalize across different indoor spaces. Individual scenes of *3DMatch* are split into point cloud fragments, which are generated by fusing 50 consecutive depth frames using TSDF volumetric fusion Curless and Levoy (1996). As a preprocessing step, we apply voxel-grid downsampling to all point clouds, and if multiple points fall into the same voxel, we randomly pick one.

ModelNet40. For each CAD model of *ModelNet40*, 2048 points are first generated by uniform sampling and scaled to fit into a unit sphere. Then we follow Yew and Lee (2020) to produce partial scans: for source partial point cloud, we uniformly sample a plane through the origin that splits the unit sphere into two half-spaces, shift that plane along its normal until $\lfloor 2048 \cdot p_v \rfloor$ points are on one side, and discard the points on the other side; the target point cloud is generated in the same manner; then the two resulting, partial point clouds are randomly rotated, translated and jittered with Gaussian noise. For the rotation, we sample a random axis and a random angle $<45°$. The translation is sampled in the range $[-0.5, 0.5]$. Gaussian noise is

	n_p	γ	V	r_p	r_s	r_o	r_m
3DMatch	256	24	0.025	0.0375	0.1	0.0375	0.05
ModelNet	384	64	0.06	0.018	0.06	0.04	0.04
odometryKITTI	512	48	0.3	0.21	0.75	0.45	0.3

Table 3.6.: Hyper-parameters configurations for different datasets.

applied per coordinate with $\sigma = 0.05$. Finally, 717 points are randomly sampled from the $\lfloor 2048 \cdot p_v \rfloor$ points.

odometryKITTI. The dataset was captured using a Velodyne HDL-64 3D laser scanner by driving around the mid-size city of Karlsruhe, in rural areas and on highways. The ground truth poses are provided by GPS/IMU system. We follow Bai et al. (2020) to use ICP to reduce the noise in the ground truth poses.

3.6.3. Implementation and training

For 3DMatch/Modelnet/KITTI, we train PREDATOR using Stochastic Gradient Descent for 30/ 200/ 150 epochs, with initial learning rate 0.005/ 0.01/ 0.05, momentum 0.98, and weight decay 10^{-6}. The learning rate is exponentially decayed by 0.05 after each epoch. Due to memory constraints we use batch size 1 in all experiments. The dataset-dependent hyper-parameters which include number of negative pairs in circle loss n_p, temperature factor γ, voxel size V, search radius for positive pair r_p, safe radius r_s, overlap and matchability radius r_o and r_m are given in Tab. 3.6. On odometryKITTI dataset, we take the curriculum learning Bengio et al. (2009) strategy to gradually learn sharper local descriptors by adjusting n_p. For more details please see our code.

3.6.4. Network architecture

The detailed network architecture of PREDATOR is depicted in Fig. 3.11. Our model is built on the KPConv implementation from the D3Feat repository.[6] We complement each KPConv layer with instance normalisation Leaky ReLU activations. The l-th strided convolution is applied to a point cloud dowsampled with voxel size $2^l \cdot V$. Upsampling in the decoder is performed by querying the associated feature of the closest point from the previous layer.

With ≈20k points after voxel-grid downsampling, the point clouds in *3DMatch* are much denser than those of *ModelNet40* with only 717 points. Moreover, they also have larger spatial extent with bounding boxes up to $3 \times 3 \times 3$ m^3, while *ModelNet40* point clouds are normalised to fit into a unit sphere. To account for these large differences, we slightly adapt the encoder and decoder per dataset, but keep the same overlap attention model. Differences in network hyper-parameters are shown in Tab. 3.7.

[6] `https://github.com/XuyangBai/D3Feat.pytorch`

	# strided convolutions	convolution radius	first conv. feature dim.	final feature dim.
3DMatch	3	2.5	64	32
ModelNet	2	2.75	256	96
odometryKITTI	3	4.25	128	32

Table 3.7.: Different network configurations for *3DMatch*, *ModelNet* and *odometryKITTI* datasets.

	3DMatch										*3DLoMatch*									
	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Avg.	STD	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Avg.	STD
								# Sample												
	449	106	159	182	78	26	234	45	160	128	**524**	283	222	210	138	42	237	70	191	154
								Registration Recall (%) ↑												
3DSN Gojcic et al. (2019)	90.6	90.6	65.4	89.6	82.1	80.8	68.4	60.0	78.4	11.5	51.4	25.9	44.1	41.1	30.7	36.6	14.0	20.3	33.0	11.8
FCGF Choy et al. (2019b)	**98.0**	94.3	68.6	**96.7**	91.0	**84.6**	76.1	71.1	85.1	11.0	60.8	42.2	53.6	53.1	38.0	26.8	16.1	30.4	40.1	14.3
D3Feat Bai et al. (2020)	96.0	86.8	67.3	90.7	88.5	80.8	78.2	64.4	81.6	10.5	49.7	37.2	47.3	47.8	36.5	31.7	15.7	31.9	37.2	**10.6**
Ours	97.6	**97.2**	**74.8**	**98.9**	**96.2**	**88.5**	**85.9**	**73.3**	**89.0**	**9.6**	**71.5**	**58.2**	**60.8**	**77.5**	**64.2**	**61.0**	**45.8**	**39.1**	**59.8**	11.7
								Relative Rotation Error (°) ↓												
3DSN Gojcic et al. (2019)	1.926	1.843	2.324	2.041	1.952	2.908	2.296	2.301	2.199	0.321	3.020	3.898	3.427	3.196	3.217	3.328	4.325	3.814	3.528	0.414
FCGF Choy et al. (2019b)	**1.767**	1.849	**2.210**	**1.867**	1.667	2.417	**2.024**	**1.792**	**1.949**	0.236	**2.904**	3.229	3.277	2.768	**2.801**	**2.822**	3.372	4.006	3.147	0.394
D3Feat Bai et al. (2020)	2.016	2.029	2.425	1.990	1.967	**2.400**	2.346	2.115	2.161	**0.183**	3.226	3.492	3.373	3.330	3.165	2.972	3.708	3.619	3.361	**0.227**
Ours	1.861	**1.806**	2.473	2.045	**1.600**	2.458	2.067	1.926	2.029	0.286	3.079	**2.637**	**3.220**	**2.694**	2.907	3.390	**3.046**	**3.412**	**3.048**	0.273
								Relative Translation Error (m) ↓												
3DSN Gojcic et al. (2019)	0.059	0.070	0.079	0.065	0.074	0.062	0.093	0.065	0.071	**0.010**	0.082	0.098	0.096	0.101	**0.080**	0.089	0.158	**0.120**	0.103	0.024
FCGF Choy et al. (2019b)	0.053	0.056	0.071	**0.062**	0.061	0.055	0.082	0.090	0.066	0.013	0.084	0.097	**0.076**	0.101	0.084	0.077	0.144	0.140	0.100	0.025
D3Feat Bai et al. (2020)	0.055	0.065	0.080	0.064	0.078	**0.049**	0.083	0.064	0.067	0.011	0.088	0.101	0.086	0.099	0.092	**0.075**	0.146	0.135	0.103	0.023
Ours	**0.048**	**0.055**	**0.070**	0.073	**0.060**	0.065	**0.080**	**0.063**	**0.064**	0.010	**0.081**	**0.080**	0.084	**0.099**	0.096	0.077	**0.101**	0.130	**0.093**	**0.016**

Table 3.8.: Detailed results on the *3DMatch* and *3DLoMatch* datasets.

3.6.5. Additional results

Detailed registration results. We report detailed per-scene *Registration Recall (RR)*, *Relative Rotation Error (RRE)* and *Relative Translation Error (RTE)* in Tab. 3.8. RRE and RTE are only averaged over successfully registered pairs for each scene, such that the numbers are mot dominated by gross errors from complete registration failures. We get the highest RR and lowest or second lowest RTE and RRE for almost all scenes, this further shows that our overlap attention module together with probabilistic sampling supports not only robust, but also accurate registration.

Feature match recall. Finally, Fig. 3.10 shows that our descriptors are robust and perform well over a wide range of thresholds for the allowable inlier distance and the minimum inlier ratio. Notably, PREDATOR consistently outperforms D3Feat that uses a similar KPConv backbone.

3.6.6. Additional ablation studies

Ablations of matchability score. We find that probabilistic sampling guided by the product of the overlap and matchability scores attains the highest RR. Here we further analyse the impact of each individual component. We first construct a baseline which applies random sampling (*rand*) over conditioned features, then we sample points with probability proportional

matchability	overlap	*3DMatch* FMR	*3DMatch* IR	*3DMatch* RR	*3DLoMatch* FMR	*3DLoMatch* IR	*3DLoMatch* RR
		96.2	51.6	86.0	74.9	20.4	43.3
✓		96.1	54.0	**89.2**	75.5	21.9	52.2
	✓	96.2	56.7	89.1	78.3	26.1	57.4
✓	✓	**96.7**	**58.0**	89.0	**78.6**	**26.7**	**59.8**

Table 3.9.: Different combinations of scores used for probabilistic sampling.

# Samples	*3DMatch* 5000	2500	1000	500	250	*3DLoMatch* 5000	2500	1000	500	250
	Registration Recall (%)									
FCGF Choy et al. (2019b)	85.1	84.7	83.3	81.6	71.4	40.1	41.7	38.2	35.4	26.8
FCGF+OA	**89.1**	**88.9**	**88.7**	**87.5**	**85.4**	**57.8**	**58.3**	**59.8**	**58.7**	**55.9**

Table 3.10.: Ablation of the proposed overlap attention module with sparse convolution backbone. FCGF + OA denotes adding proposed overlap attention module to FCGF model.

to overlap scores (*prob. (o)*), to matchability scores (*prob. (m)*), and to the combination of the two scores (*prob. (om)*). As shown in Tab. 3.9, *rand* fares clearly worse, in all metrics. Compared to *prob. (om)*, either *prob. (o)* or *prob. (m)* can achieve comparable results on *3DMatch*; the performance gap becomes big on the more challenging *3DLoMatch* dataset, where our *prob. (om)* is around 4 pp better in terms of RR.

Ablations of overlap attention module with FCGF. To demonstrate the flexibility of our model, we additionally add proposed overlap attention module to FCGF model. We train it on *3DMatch* dataset with our proposed loss for 100 epochs, the results are shown in Tab. 3.10. It shows that FCGF can also greatly benefit from the overlap attention module. Registration recall almost doubles when sampling only 250 points on the challenging *3DLoMatch* benchmark.

3.6.7. Timings

We compare the runtime of PREDATOR with FCGF[7] Choy et al. (2019b) and D3Feat[8] Bai et al. (2020) on *3DMatch*. For all three methods we set voxel size $V = 2.5\,\text{cm}$ and batch size 1. The test is run on a single GeForce GTX 1080 Ti with Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz, 32GB RAM. The most time-consuming step of our model, and also of D3Feat, is the data loader, as we have to pre-compute the neighborhood indices before the forward pass.

[7] All experiments were done with MinkowskiEngine v0.4.2.
[8] We use its PyTorch implementation.

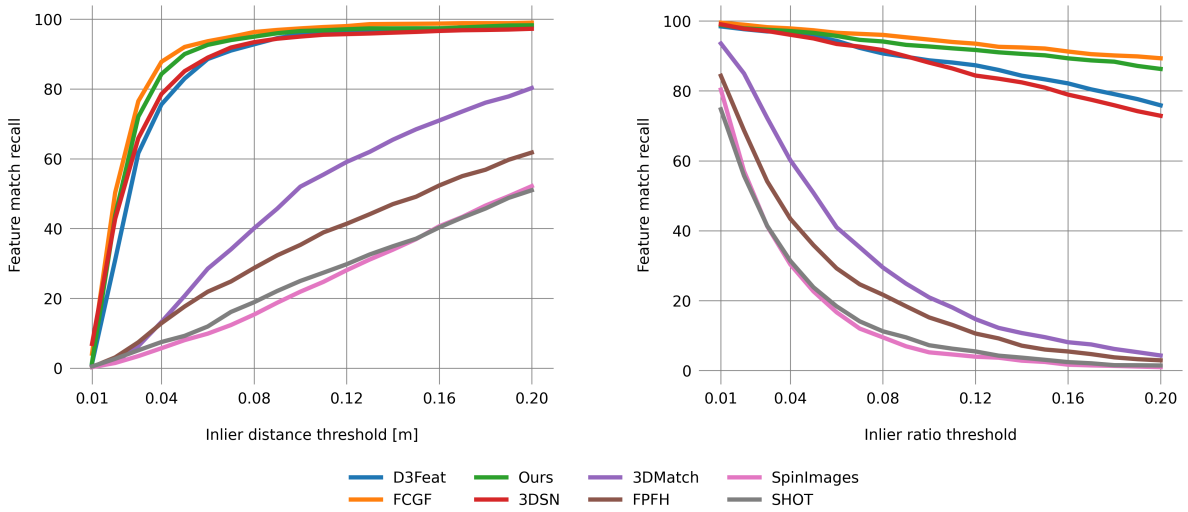


Figure 3.10.: Feature matching recall in relation to inlier distance threshold τ_1 (left) and inlier ratio threshold τ_2 (right)

	data loader	encoder	overlap attention	decoder	overall
FCGF Choy et al. (2019b)	**6**	414	—	25	445
D3Feat Bai et al. (2020)	200	11	—	63	274
Ours	191	**9**	70	**1**	**271**

Table 3.11.: Runtime per fragment pair in milli-seconds, averaged over 1623 test pairs of *3DMatch*.

With its smaller encoder and decoder, but the additional overlap attention module, PREDATOR is still marginally faster than D3Feat. FCGF has a more efficient data loader that relies on sparse convolution and queries neighbors during the forward pass. See Tab. 3.11.

3.6.8. Qualitative visualization

We show more qualitative results in Fig. 3.12 and Fig. 3.13 for *3DLoMatch* and *ModelLoNet* respectively. The input points clouds are rotated and translated here for better visualization of overlap and matchability scores.

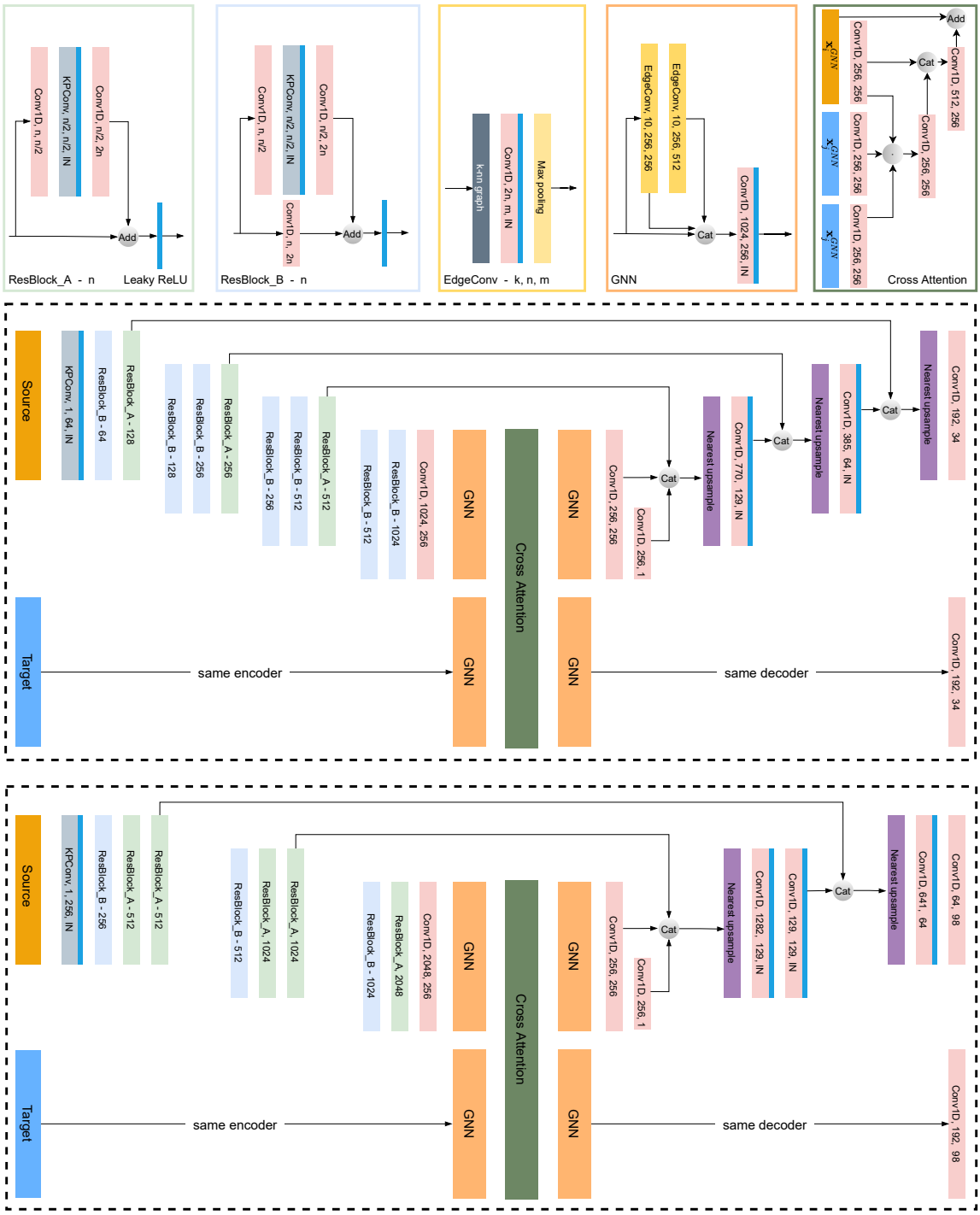


Figure 3.11.: Network architecture of PREDATOR for *3DMatch* (*middle*) and *ModelNet* (*bottom*). In the cross attention module, for each (query $\mathbf{s}_i \in \mathbb{R}^{b\times 1}$, key $\mathbf{k}_i \in \mathbb{R}^{b\times 1}$, value $\mathbf{v}_i \in \mathbb{R}^{b\times 1}$), $\odot$ denotes first reshape them into shape $(4, \frac{b}{4})$(4 heads), then compute scores matrix $\mathbf{S}$ from $\mathbf{s}_i$ and $\mathbf{k}_i$, finally get message update from $\mathbf{v}_i$ and reshape back to $(b, 1)$.

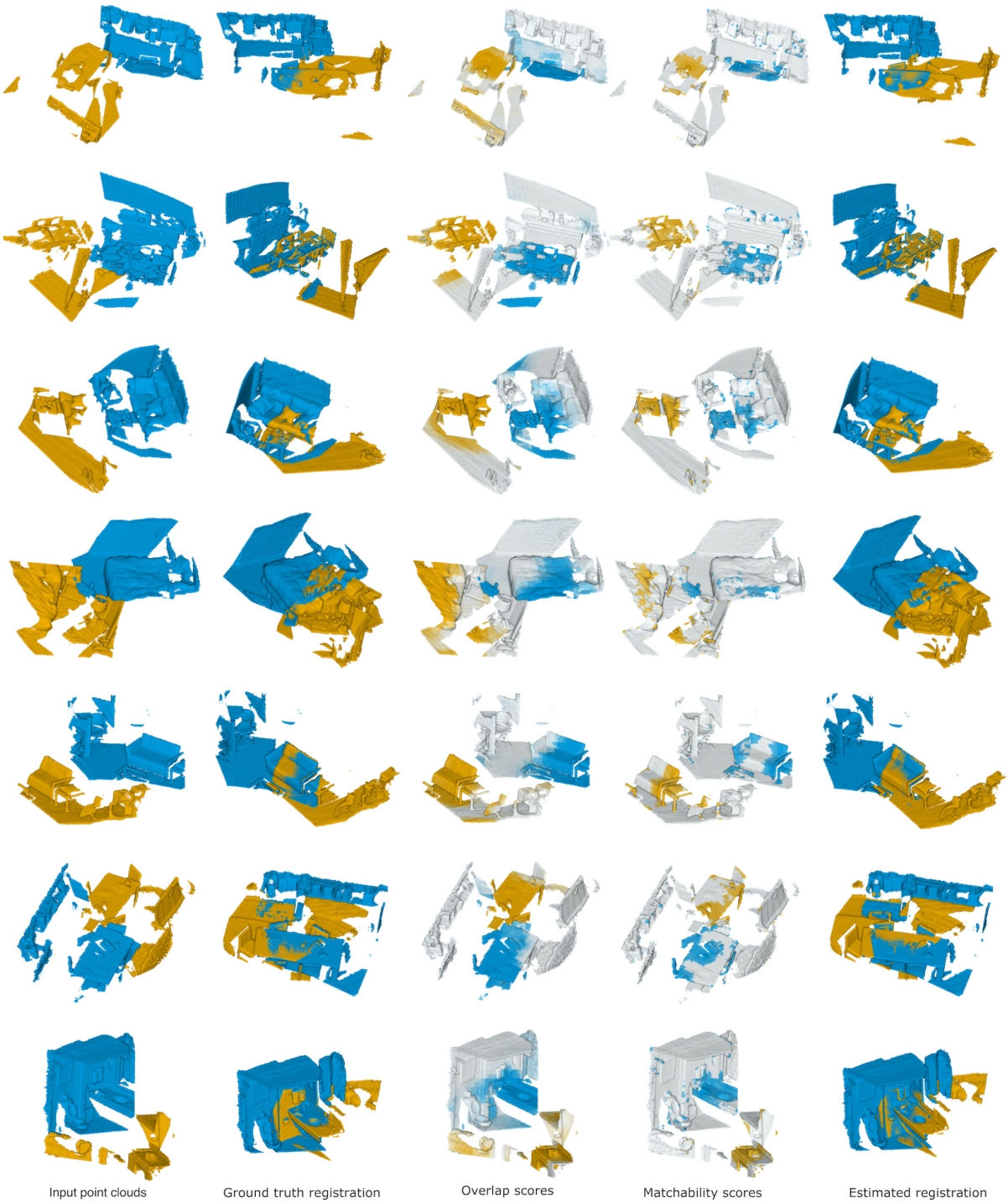


Figure 3.12.: Example results on *3DLoMatch*.

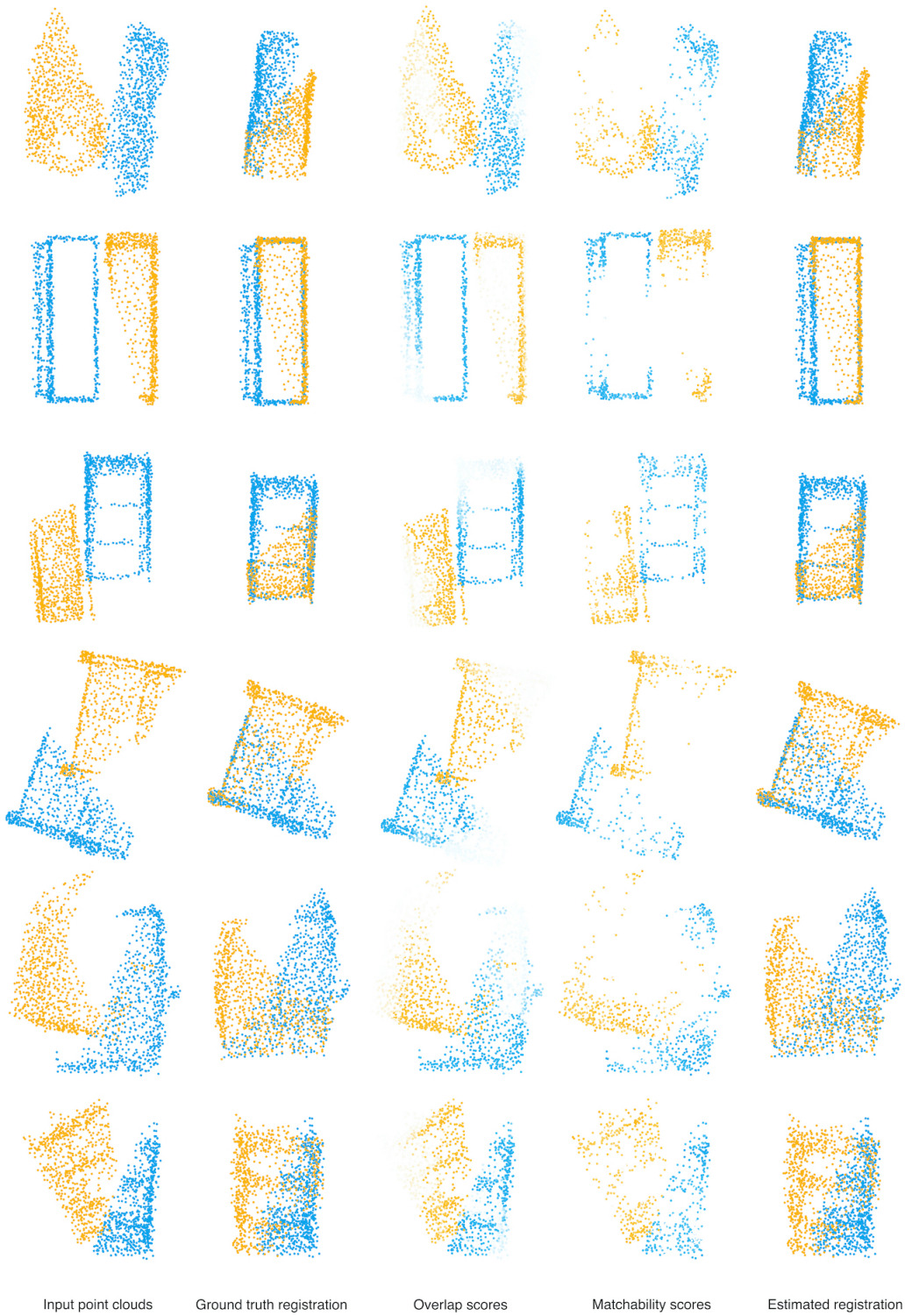


Figure 3.13.: Example results on *ModelLoNet*.

4 Dynamic 3D Scene Analysis by Point Cloud Accumulation

Shengyu Huang, Zan Gojcic, Jiahui Huang, Andreas Wieser, Konrad Schindler
European Conference on Computer Vision, 2021

(Author version; for typeset version please refer to the original conference paper.)

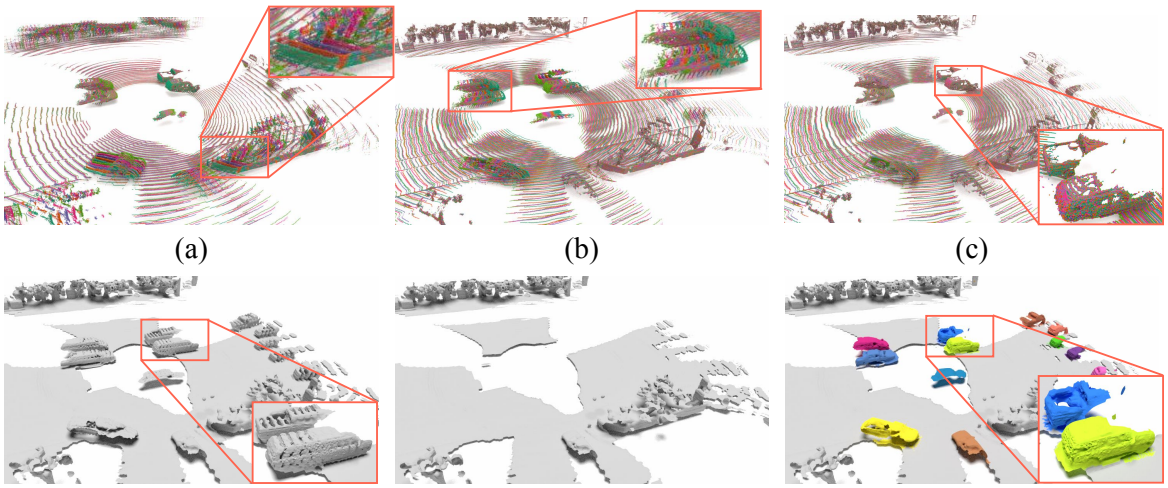


Figure 4.1.: Points in LiDAR frames acquired over time are not aligned due to the motion of the sensor and of other agents in the scene (a, d). Static background points can be aligned using ego-motion, but this smears the dynamic points across their trajectories (b). While motion segmentation only enables removing the moving points from the scene (e), our method properly disentangles individual moving objects from the static part and accumulates both correctly (c, f).

Abstract

Multi-beam LiDAR sensors, as used on autonomous vehicles and mobile robots, acquire sequences of 3D range scans ("frames"). Each frame covers the scene sparsely, due to limited angular scanning resolution and occlusion. The sparsity restricts the performance of downstream processes like semantic segmentation or surface reconstruction. Luckily, when the sensor moves, frames are captured from a sequence of different viewpoints. This provides complementary information and, when accumulated in a common scene coordinate frame, yields a

denser sampling and a more complete coverage of the underlying 3D scene. However, often the scanned scenes contain moving objects. Points on those objects are not correctly aligned by just undoing the scanner's ego-motion. In the present paper, we explore multi-frame point cloud accumulation as a mid-level representation of 3D scan sequences, and develop a method that exploits inductive biases of outdoor street scenes, including their geometric layout and object-level rigidity. Compared to state-of-the-art scene flow estimators, our proposed approach aims to align all 3D points in a common reference frame correctly accumulating the points on the individual objects. Our approach greatly reduces the alignment errors on several benchmark datasets. Moreover, the accumulated point clouds benefit high-level tasks like surface reconstruction. [`project page`]

4.1. Introduction

LiDAR point clouds are a primary data source for robot perception in dynamically changing 3D scenes. They play a crucial role in mobile robotics and autonomous driving. To ensure awareness of a large field of view at any point in time, 3D point measurements are acquired as a sequence of sparse scans that each cover a large field-of-view—typically, the full 360°. In each individual scan *(i)* the point density is low, and *(ii)* some scene parts are occluded. Both issues complicate downstream processing. One way to mitigate the problem is to assume the sensor ego-motion is known and to align multiple consecutive scans into a common scene coordinate frame, thus accumulating them into a denser and more complete point cloud. This simple accumulation strategy already boosts performance for perception tasks like object detection Caesar et al. (2020) and semantic segmentation Behley et al. (2019), but it also highlights important problems. First, to obtain sufficiently accurate sensor poses the ego-motion is typically computed in post-processing to enable sensor fusion and loop closures — meaning that it would actually not be available for online perception. Second, compensating the sensor ego-motion only aligns scan points of the static background, while moving foreground objects are smeared out along their motion trajectories (Fig. 4.1b).

To properly accumulate 3D points across multiple frames, one must disentangle the individual moving objects from the static background and reason about their spatio-temporal properties. Since the grouping of the 3D points into moving objects itself depends on their motion, the task becomes a form of multi-frame 3D scene flow estimation. Traditional scene flow methods Liu et al. (2019a); Wu et al. (2020b); Puy et al. (2020) model dynamics in the form of a free-form velocity field from one frame to the next, only constrained by some form of (piece-wise) smoothing Vogel et al. (2013, 2015); Dewan et al. (2016).

While this is a very flexible and general approach, it also has two disadvantages in the context of autonomous driving scenes: *(i)* it ignores the geometric structure of the scene, which normally consists of a dominant, static background and a small number of discrete objects that, at least locally, move rigidly; *(ii)* it also ignores the temporal structure and only looks at the minimal setup of two frames. Consequently, one risks physically implausible scene flow estimates Gojcic et al. (2021), and does not benefit from the dense temporal sequence of scans.

Starting from these observations, we propose a novel point cloud accumulation scheme tailored to the autonomous driving setting. To that end, we aim to accumulate point clouds over time while abstracting the scene into a collection of rigidly moving agents Behl et al. (2019); Gojcic et al. (2021); Teed and Deng (2021) and reasoning about each agent's motion on the basis of a longer sequence of frames Huang et al. (2021a, 2022a). Along with the accumulated point cloud (Fig. 4.1c), our method provides more holistic scene understanding, including foreground/background segmentation, motion segmentation, and per-object parametric motion compensation. As a result, our method can conveniently serve as a common, low-latency preprocessing step for perception tasks including surface reconstruction (Fig. 4.1f) and semantic segmentation Behley et al. (2019).

We carry out extensive evaluations on two autonomous driving datasets *Waymo* Sun et al. (2020a) and *nuScenes* Caesar et al. (2020), where our method greatly outperforms prior art. For example, on *Waymo* we reduce the average endpoint error from 12.9 cm to 1.8 cm for the static part and from 23.7 cm to 17.3 cm for the dynamic objects. We observe similar performance

gains also on *nuScenes*.

In summary, we present a novel, learnable model for temporal accumulation of 3D point cloud sequences over multiple frames, which disentangles the background from dynamic foreground objects. By decomposing the scene into agents that move rigidly over time, our model is able to learn multi-frame motion and reason about motion in context over longer time sequences. Moreover, our method allows for low-latency processing, as it operates on raw point clouds and requires only their sequence order as further input. It is therefore suitable for use in online scene understanding, for instance as a low-level preprocessor for semantic segmentation or surface reconstruction.

4.2. Related work

Temporal point cloud processing. Modeling a sequence of point clouds usually starts with estimating accurate correspondences between frames, for which scene flow emerged as a popular representation. Originating from Vedula et al. (1999), scene flow estimation builds an intuitive and effective dynamic scene representation by computing a flow vector for each source point. While traditional scene flow methods Wedel et al. (2008); Vogel et al. (2011, 2013, 2015) leverage motion smoothness as regularizer within their optimization frameworks, modern learning-based methods learn the preference for smooth motions directly from large-scale datasets Liu et al. (2019a); Wu et al. (2020b); Puy et al. (2020); Ouyang and Raviv (2021). Moreover, manually designed scene priors proved beneficial for structured scenes, for instance supervoxel-based local rigidity constraints Li et al. (2021a), or object-level shape priors learned in fully supervised Behl et al. (2019) or weakly supervised Gojcic et al. (2021) fashion. Methods like SLIM Baur et al. (2021) take a decoupled approach, where they first run motion segmentation before deriving scene flows for each segment separately. Treating the entire point cloud sequence as 4D data and applying a spatio-temporal backbone Liu et al. (2019b); Choy et al. (2019a); Fan et al. (2020) demonstrates superior performance and efficiency. Subsequent works enhance such backbones by employing long-range modeling techniques such as Transformers Vaswani et al. (2017); Fan et al. (2021); Yang et al. (2021b), or by coupling downstream tasks like semantic segmentation Aygun et al. (2021), object detection Yang et al. (2021a); Qi et al. (2021) and multi-modal fusion Piergiovanni et al. (2021). While our method employs the representation of multi-frame scene flow, we explicitly model individual dynamic objects, which not only provides a high-level scene decomposition, but also yields markedly higher accuracy.

Motion segmentation. Classification of the points into static and dynamic scene parts serves as an essential component in our pipeline. Conventional geometric approaches either rely on ray casting Chen and Yang (2016); Schauer and Nüchter (2018) over dense terrestrial laser scans to build clean static maps, or on visibility Pomerleau et al. (2014); Kim and Kim (2020) to determine the dynamics of the query point by checking its occlusion state in a dense map. Removert Kim and Kim (2020) iteratively recovers falsely removed static points from multi-scale range images. Most recently, learning-based methods formulate and solve the segmentation task in a data-driven way: Chen *et al*. Chen et al. (2021) propose a deep model over multi-

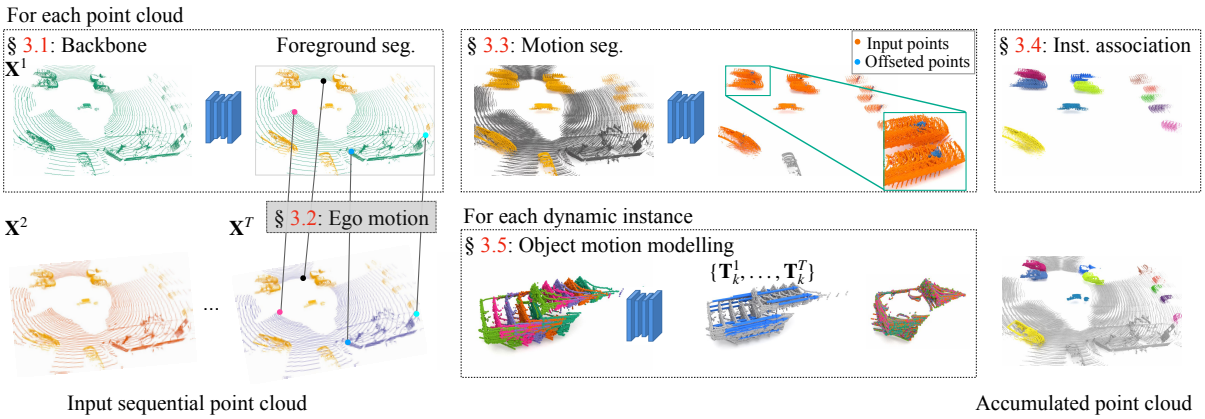


Figure 4.2.: **Overview**. Our method takes in a point cloud sequence of T frames and starts by extracting foreground points (marked yellow) for each frame. To obtain ego motion, $T-1$ pairwise registrations are performed. Next, points belonging to dynamic foreground object are extracted using our motion segmentation module (marked orange). To boost subsequent spatio-temporal instance association, we additionally predict per-point offset vectors. After instance association, we finally compute the rigid motion separately for each segmented dynamic object.

ple range image residuals and show SoTA results on a newly-established motion segmentation benchmark Behley et al. (2019). Any Motion Detector Filatov et al. (2020) first extracts per-frame features from bird's-eye-view projections and then aggregates temporal information from ego-motion compensated per-frame features (in their case with a with convolutional RNN). Our work is similar in spirit, but additionally leverages information from the foreground segmentation task and object clustering in an end-to-end framework.

Dynamic object reconstruction. Given sequential observations of a rigid object, dynamic object reconstruction aims to recover the 3D geometric shape as well as its rigid pose over time. Such a task can be handled either by directly hallucinating the full shape or by registering and accumulating partial observations. Approaches of the former type usually squash partial observations into a global feature vector Yuan et al. (2018); Giancola et al. (2019); Li et al. (2019) and ignore the local geometric structure. Gu et al. (2020) go one step further by disentangling shape and pose with a novel supervised loss. However, there is still no guarantee for the fidelity of the completed shape. We instead rely on registration and accumulation. Related works include AlignNet-3D Groß et al. (2019) that directly regresses the relative transformation matrix from concatenated global features of the two point clouds. NOCS Wang et al. (2019a) proposes a category-aware canonical representation that can be used to estimate instance pose w.r.t. its canonical pose. Caspr Rempe et al. (2020) implicitly accumulates the shapes by mapping a sequence of partial observations to a continuous latent space. Huang et al. (2021a) and Huang et al. (2022a) respectively propose multi-way registration methods that accumulate multi-body and non-rigid dynamic point clouds, but do not scale well to large scenes.

4.3. Method

The network architecture of our multitask model is schematically depicted in Fig. 4.2. To accumulate the points over time, we make use of the inductive bias that scenes can be decomposed into agents that move as rigid bodies Gojcic et al. (2021). We start by extracting the latent base features of each individual frame (§ 4.3.1), which we then use as input to the task-specific heads. To estimate the ego-motion, we employ a differentiable registration module (§ 4.3.2). Instead of using the ego-motion only to align the static scene parts, we also use it to spatially align the base features, which are reused in the later stages. To explain the motion of the dynamic foreground, we utilize the aligned base features and perform motion segmentation (§ 4.3.3) as well as spatio-temporal association of dynamic foreground objects (§ 4.3.4). Finally, we decode the rigid body motion of each foreground object from its spatio-temporal features (§ 4.3.5). We train the entire model end-to-end with a loss $\mathcal{L}$ composed of five terms:

$$\mathcal{L} = \mathcal{L}_{\text{ego}} + \mathcal{L}_{\text{FG}} + \mathcal{L}_{\text{motion}} + \lambda_{\text{offset}}\mathcal{L}_{\text{offset}} + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}} \,. \tag{4.1}$$

$$\mathcal{L} = \mathcal{L}_{\text{ego}} + \mathcal{L}_{\text{FG}} + \mathcal{L}_{\text{motion}} + \lambda_{\text{offset}}\mathcal{L}_{\text{offset}} + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}} \,. \tag{4.2}$$

In the following, we provide a high-level description of each module. Detailed network architectures, including parameters and loss formulations, are given in the supplementary material (unless already elaborated).

Problem setting. Consider an *ordered* point cloud sequence $\mathcal{X} = \{\mathbf{X}^t\}_{t=1}^T$ consisting of T frames $\mathbf{X}^t = [\mathbf{x}_1^t, ..., \mathbf{x}_i^t, ..., \mathbf{x}_{n_t}^t] \in \mathbb{R}^{3\times n_t}$ of variable size, captured by a *single moving observer* at constant time intervals Δt. We denote the first frame $\mathbf{X}^1$ the *target* frame, while the remaining frames $\{\mathbf{X}^t \mid t > 1\}$ are termed *source* frames. Our goal is then to estimate the flow vectors $\{\mathbf{V}^t \in \mathbb{R}^{3\times n_t} | t > 1\}$ that align each of the source frames to the target frame, and hence *accumulate* the point clouds. Instead of predicting unconstrained pointwise flow vectors, we make use of the inductive bias that each frame can be decomposed into a *static* part $\mathbf{X}_{\text{static}}^t$ and K_t rigidly-moving *dynamic* parts $\mathcal{X}_{\text{dynamic}}^t = \{\mathbf{X}_k^t\}_{k=1}^{K_t}$ Gojcic et al. (2021). For an individual frame, the scene flow vectors $\mathbf{V}_{\text{static}}^t$ of the static part and $\mathbf{V}_k^t$ of the k-th dynamic object can be explained by the rigid ego-motion $\mathbf{T}_{\text{ego}}^t \in \text{SE}(3)$ and the motion of the dynamic object relative to the static background $\mathbf{T}_k^t \in \text{SE}(3)$, respectively as:

$$\mathbf{V}_{\text{static}}^t = \mathbf{T}_{\text{ego}}^t \circ \mathbf{X}_{\text{static}}^t - \mathbf{X}_{\text{static}}^t, \qquad \mathbf{V}_k^t = \mathbf{T}_k^t \mathbf{T}_{\text{ego}}^t \circ \mathbf{X}_k^t - \mathbf{X}_k^t, \tag{4.3}$$

where $\mathbf{T} \circ \mathbf{X}$ (or $\mathbf{T} \circ \mathbf{x}$) denotes applying the transformation to the point set $\mathbf{X}$ (or point $\mathbf{x}$).

4.3.1. Backbone network

Similar to Baur et al. (2021); Jund et al. (2021); Wu et al. (2020a), our backbone network converts the 3D point cloud of a single frame into a bird's eye view (BEV) latent feature image. Specifically, we lift the point coordinates to a higher-dimensional latent space using a pointwise MLP and then scatter them into a $H \times W$ feature grid aligned with the gravity axis. The features per grid cell ("pillar") are aggregated with max-pooling, then fed through a 2D

UNet Long et al. (2015) to enlarge their receptive field and strengthen the local context. The output of the backbone network is a 2D latent *base* feature map $\mathbf{F}^t_{\text{base}}$ for each of the T frames.

4.3.2. Ego-motion estimation

We estimate the ego-motion $\mathbf{T}^t_{\text{ego}}$ using a correspondence-based registration module separately for each source frame. Points belonging to dynamic objects can bias the estimate of ego-motion, especially when using a correspondence-based approach, and should therefore be discarded. However, at an early stage of the pipeline it is challenging to reason about scene dynamics, so we rather follow the conservative approach and classify points into background and foreground, where foreground contains all the *movable* objects (*e.g.*, cars and pedestrians), irrespective of their actual dynamics Gojcic et al. (2021). The predicted foreground mask is later used to guide motion segmentation in § 4.3.3.

We start by extracting ego-motion features $\mathbf{F}^t_{\text{ego}}$ and foreground scores $\mathbf{s}^t_{\text{FG}}$ from each $\mathbf{F}^t_{\text{base}}$ using two dedicated heads, each consisting of two convolutional layers separated by a ReLU activation and batch normalization. We then randomly sample N_{ego} background pillars whose $\mathbf{s}^t_{\text{FG}} < \tau$ and compute the pillar centroid coordinates $\mathbf{P}^t = \{\mathbf{p}^t_l\}$. The ego motion $\mathbf{T}^t_{\text{ego}}$ is estimated as:

$$\mathbf{T}^t_{\text{ego}} = \text{argmin}_{\mathbf{T}} \sum_{l=1}^{N_{\text{ego}}} w^t_l \left\| \mathbf{T} \circ \mathbf{p}^t_l - \phi(\mathbf{p}^t_l, \mathbf{P}^1) \right\|^2 . \tag{4.4}$$

Here, $\phi(\mathbf{p}^t_l, \mathbf{P}^1)$ finds the *soft correspondence* of $\mathbf{p}^t_l$ in $\mathbf{P}^1$, and w^t_l is the weight of the correspondence pair $\left(\mathbf{p}^t_l, \phi(\mathbf{p}^t_l, \mathbf{P}^1)\right)$. Both $\phi(\mathbf{p}^t_l, \mathbf{P}^1)$ and w^t_l are estimated using an entropy-regularized Sinkhorn algorithm from $\mathbf{F}^t_{\text{ego}}$ with slack row/column padded Cuturi (2013); Yew and Lee (2020) and the optimal $\mathbf{T}^t_{\text{ego}}$ is computed in closed form via the differentiable Kabsch algorithm Kabsch (1976).

We supervise the ego-motion with an L_1 loss over the transformed pillars $\mathcal{L}_{\text{trans}} = \frac{1}{|\mathbf{P}^t|} \sum_{l=1}^{|\mathbf{P}^t|} \|\mathbf{T} \circ \mathbf{p}^t_l - \overline{\mathbf{T}} \circ \mathbf{p}^t_l\|_1$ and an inlier loss $\mathcal{L}_{\text{inlier}}$ Yew and Lee (2020) that regularizes the Sinkhorn algorithm, $\mathcal{L}^{\bullet}_{\text{ego}} = \mathcal{L}_{\text{trans}} + \mathcal{L}_{\text{inlier}}$. The foreground score $\mathbf{s}^t_{\text{FG}}$ is supervised using a combination of weighted binary cross-entropy (BCE) loss $\mathcal{L}_{\text{bce}}$ and Lovasz-Softmax loss $\mathcal{L}_{\text{ls}}$ Berman et al. (2018): $\mathcal{L}^{\bullet}_{\text{FG}} = \mathcal{L}_{\text{bce}}(\mathbf{s}^t_{\text{FG}}, \bar{\mathbf{s}}^t_{\text{FG}}) + \mathcal{L}_{\text{ls}}(\mathbf{s}^t_{\text{FG}}, \bar{\mathbf{s}}^t_{\text{FG}})$, with $\bar{\mathbf{s}}^t_{\text{FG}}$ the binary ground truth. The weights in $\mathcal{L}_{\text{bce}}$ are inversely proportional to the square root of elements in each class.

4.3.3. Motion segmentation

To separate the *moving* objects from the *static* ones we perform motion segmentation, reusing the per-frame base features $\{\mathbf{F}^t_{\text{base}}\}$. Specifically, we apply a differentiable feature warping scheme Sun et al. (2018) that warps each $\mathbf{F}^t_{\text{base}}$ using the predicted ego-motion $\mathbf{T}^t_{\text{ego}}$, and obtain a spatio-temporal 3D feature tensor of size $C \times T \times H \times W$ by stacking the warped feature maps along the channel dimension. This feature tensor is then fed through a series of 3D convolutional layers, followed by max-pooling across the temporal dimension T. Finally, we apply a small 2D UNet to obtain the 2D motion feature map $\mathbf{F}_{\text{motion}}$.

To mitigate discretization error, we bilinearly interpolate grid motion features to all foreground points in each frame.[1] The point-level motion feature for $\mathbf{x}_i^t$ is computed as:

$$\mathbf{f}_{\text{motion},i}^t = \text{MLP}\big(\text{cat}[\psi(\mathbf{x}_i^t, \mathbf{F}_{\text{motion}}), \text{MLP}(\mathbf{x}_i^t)]\big), \tag{4.5}$$

where $\text{MLP}(\cdot)$ denotes a multi-layer perceptron, $\text{cat}[\cdot]$ concatenation, and $\psi(\mathbf{x}, \mathbf{F})$ a bilinear interpolation from $\mathbf{F}$ to $\mathbf{x}$. The dynamic score $\mathbf{s}_i^t$ of the point $\mathbf{x}_i^t$ is then decoded from the motion feature $\mathbf{f}_{\text{motion},i}^t$ using another MLP, and supervised similar to the foreground segmentation, with a loss $\mathcal{L}_{\text{motion}}^{\bullet} = \mathcal{L}_{\text{bce}}(\mathbf{s}_i^t, \overline{\mathbf{s}}_i^t) + \mathcal{L}_{\text{ls}}(\mathbf{s}_i^t, \overline{\mathbf{s}}_i^t)$, where $\overline{\mathbf{s}}_i^t$ denotes the ground-truth motion label of point $\mathbf{x}_i^t$.

4.3.4. Spatio-temporal instance association

To segment the dynamic points (extracted by thresholding the $\mathbf{s}_i^t$) into individual objects and associate them over time, we perform spatio-temporal instance association. Different from the common tracking-by-detection Dendorfer et al. (2021); Weng et al. (2020) paradigm, we propose to directly *cluster the spatio-temporal point cloud*, which simultaneously provides instance masks and the corresponding associations. However, naive clustering of the ego-motion aligned point clouds often fails due to LiDAR sparsity and fast object motions, hence we predict a per-point offset vector $\boldsymbol{\delta}_i^t$ pointing towards the (motion-compensated) instance center:

$$\boldsymbol{\delta}_i^t = \text{MLP}\big(\text{cat}[\psi(\mathbf{x}_i^t, \mathbf{F}_{\text{motion}}), \text{MLP}(\mathbf{x}_i^t)]\big). \tag{4.6}$$

The DBSCAN Ester et al. (1996) algorithm is subsequently applied over the deformed point set $\{\mathbf{x}_i^t + \boldsymbol{\delta}_i^t \mid \forall i, \forall t\}$ to obtain an instance index for each point. This association scheme is simple yet robust, and can seamlessly handle occlusions and mis-detections. Similar to 3DIS Lahoud et al. (2019) we supervise the offset predictions $\boldsymbol{\delta}_i^t$ with both an L_1-distance loss and a directional loss:

$$\mathcal{L}_{\text{offset}}^{\bullet} = \frac{1}{n} \sum_{\{i,t\}}^{n} \left(\left\| \boldsymbol{\delta}_i^t - \overline{\boldsymbol{\delta}}_i^t \right\|_1 + 1 - \langle \frac{\boldsymbol{\delta}_i^t}{\|\boldsymbol{\delta}_i^t\|}, \frac{\overline{\boldsymbol{\delta}}_i^t}{\|\overline{\boldsymbol{\delta}}_i^t\|} \rangle \right), \tag{4.7}$$

where $\overline{\boldsymbol{\delta}}$ is the ground truth offset $\mathbf{o} - \mathbf{x}$ from the associated instance centroid $\mathbf{o}$ in the target frame, and $\langle \cdot \rangle$ is the inner product.

4.3.5. Dynamic object motion modelling

Once we have spatio-temporally segmented objects, we must recover their motions at each frame. As LiDAR points belonging to a single object are sparse and explicit inter-frame correspondences are hard to find, we take a different approach from the one used in the ego-motion head and construct a novel TubeNet to directly regress the transformations. Specifically, TubeNet takes T frames of the same instance $\mathbf{X}_k$ as input, and regresses its rigid motion parameters $\mathbf{T}_k^t$ as:

$$\mathbf{T}_k^t = \text{MLP}\left(\text{cat}[\tilde{\mathbf{f}}_{\text{motion}}, \tilde{\mathbf{f}}_{\text{ego}}, \tilde{\mathbf{f}}_{\text{pos}}^t, \tilde{\mathbf{f}}_{\text{pos}}^1]\right), \tag{4.8}$$

[1] We predict motion labels only for foreground and treat background points as static.

where $\tilde{\mathbf{f}}_{\text{motion}}$ and $\tilde{\mathbf{f}}_{\text{ego}}$ are instance-level global features obtained by applying PointNet Qi et al. (2017) to the respective point-level features of that instance, $\tilde{\mathbf{f}}_* = \text{PN}(\{\mathbf{f}^t_{*,i} \mid \mathbf{x}^t_i \in \mathbf{X}^t_k\})$. Recall that point-level features $\mathbf{f}^t_{*,i}$ are computed from $\mathbf{F}_*$ via the interpolation scheme described in Eq. (4.5). Here, $\tilde{\mathbf{f}}_{\text{motion}}$ encodes the overall instance motion while $\tilde{\mathbf{f}}_{\text{ego}}$ supplements additional geometric cues. The feature $\tilde{\mathbf{f}}^t_{\text{pos}} = \text{PN}(\mathbf{X}^t_k)$ is a summarized encoding over individual frames and provides direct positional information for accurate transformation estimation. The transformations are initialised to identity and TubeNet is applied in iterative fashion to regress residual transformations relative to the last iteration, similar to RPMNet Yew and Lee (2020).

For the loss function, we choose to parameterise each $\mathbf{T}$ as an un-normalised quaternion $\mathbf{q} \in \mathbb{R}^4$ and translation vector $\mathbf{t} \in \mathbb{R}^3$, and supervise it with:

$$\mathcal{L}^{\bullet}_{\text{obj}} = \mathcal{L}_{\text{trans}} + \frac{1}{T-1}\sum_{t=2}^{T}\left(\left\|\bar{\mathbf{t}}^t_k - \mathbf{t}^t_k\right\|_2 + \lambda\left\|\bar{\mathbf{q}}^t_k - \frac{\mathbf{q}^t_k}{\|\mathbf{q}^t_k\|}\right\|_2\right), \tag{4.9}$$

where $\bar{\mathbf{t}}^t_k$ and $\bar{\mathbf{q}}^t_k$ are the ground truth transformation, and λ is a constant weight, set to 50 in our experiments. $\mathcal{L}_{\text{trans}}$ is the same as in the ego-motion (§ 4.3.2).

4.3.6. Comparison to related work

WsRSF Gojcic et al. (2021). Our proposed method differs from WsRSF in several ways: *(i)* WsRSF is a pair-wise scene flow estimation method, while we can handle multiple frames; *(ii)* unlike WsRSF we perform motion segmentation for a more complete understanding of the scene dynamics; *(iii)* our method outputs instance-level associations, while WsRSF simply connects each instance to the complete foreground of the other point cloud.

MotionNet Wu et al. (2020a). Similar to our method, MotionNet also deals with sequential point clouds and uses a BEV representation. However, MotionNet *(i)* assumes that ground truth ego-motion is available, while we estimate it within our network; and *(ii)* does not provide object-level understanding, rather it only separates the scene into a static and a dynamic part.

4.3.7. Implementation details

Our model is implemented in pytorch Paszke et al. (2021) and can be trained on a single RTX 3090 GPU. During training we minimize Eq. (5.15) with the Adam Kingma and Ba (2014) optimiser, with an initial learning rate 0.0005 that exponentially decays at a rate of 0.98 per epoch. For both *Waymo* and *nuScenes*, the size of the pillars is $(\delta_x, \delta_y, \delta_z) = (0.25, 0.25, 8)$ m. We sample $N_{\text{ego}} = 1024$ points for ego-motion estimation and set $\tau = 0.5$ for foreground/background segmentation. The feature dimensions of $\mathbf{F}^t_{\text{base}}$, $\mathbf{F}^t_{\text{ego}}$, $\mathbf{F}_{\text{motion}}$ are 32, 64, 64 respectively. During inference we additionally use ICP Besl and McKay (1992) to perform test-time optimisation of the ego-motion as well as the transformation parameters of each dynamic object. ICP thresholds for ego-motion and dynamic object motion are 0.1/0.2 and 0.15/0.25 m for *Waymo* / *nuScenes*.

4.4. Conclusion

We have looked at the analysis of 3D point cloud sequences from a fresh viewpoint, as point cloud accumulation across time. In that view we integrate point cloud registration, motion segmentation, instance segmentation, and piece-wise rigid scene flow estimation into a complete multi-frame 4D scene analysis method. By jointly considering sequences of frames, our model is able to disentangle scene dynamics and recover accurate instance-level rigid-body motions. The model processes (ordered) raw point clouds, and can operate online with low latency. A major *limitation* is that our approach is fully supervised and heavily relies on annotated data: it requires instance-level segmentations as well as ground truth motions, although we demonstrate some robustness to label noise from interpolated pseudo ground truth. Also, our system consists of multiple processing stages and cannot fully recover from mistakes in early stages, like incorrect motion segmentation.

In *future work* we hope to explore our method's potential for downstream scene understanding tasks. We also plan to extend it to an incremental setting, where longer sequences of frames can be summarized into our holistic, dynamic scene representation in an online fashion.

4.5. Appendix

In this supplementary document, we first present additional information about our network architecture and implementation in § 4.5.1. We then elaborate on technical details of ego-motion estimation, iterative pose refinement, and scene reconstruction in § 5.7.3. Precise definitions of loss functions and evaluation metrics are provided in § 4.5.3. Further analysis of the two datasets, *nuScenes* and *Waymo*, is presented in § 4.5.4, followed by additional quantitative results including scene flow estimation, instance association, and the TubeNet motion model in § 4.5.5. Finally, we show more qualitative results in § 4.5.6.

4.5.1. Network and implementation

Network architecture. The detailed network architecture is depicted in Fig. 4.8. Our network is a sequential model consisting of (i) per-frame feature extraction used to estimate ego-motion, (ii) multi-frame feature extraction to segment dynamic objects and regress offset vectors towards the associated instance center, and (iii) TubeNet to regress the rigid motions of dynamic objects. The Pillar encoder and the two UNets operate without Batch Normalisation Ioffe and Szegedy (2015), this speeds up training and inference without any loss in performance Peng et al. (2020). Our network achieves flexibility w.r.t. the number of input frames via global max-pooling along the temporal dimension in the InitConv3D block (Fig. 4.8).

Implementation details. We use `torch_scatter`[2] to efficiently convert point-wise features to pillar/instance-level global features. To spatially align the backbone features we use `grid_sample`, implemented in PyTorch Paszke et al. (2021). Before clustering, we apply voxel down-sampling implemented in TorchSparse Tang et al. (2022) to reduce the point density and improve clustering efficiency. The voxel size is set to 15 cm. Instance labels at full resolution are recovered by indexing points to their associated voxel cell.

4.5.2. Methodology

Ego-motion estimation. Given two sets $(\mathbf{P}^1, \mathbf{P}^t)$ of pillar centroid coordinates and associated L_2-normalised features $(\mathbf{F}^1_{\text{ego}}, \mathbf{F}^t_{\text{ego}})$, we first compute the cost matrix $\mathbf{M}^t = 2 - 2\langle \mathbf{F}^t_{\text{ego}}, \mathbf{F}^1_{\text{ego}} \rangle$ and an Euclidean distance matrix $\mathbf{D}^t_{l,m} = \|\mathbf{p}^t_l - \mathbf{p}^1_m\|_2$ from pillar coordinates. We then pad $\mathbf{M}^t$ with a learnable slack row and column to accommodate outliers, before iteratively alternating between row normalisation and column normalisation[3] for five times to approximate a doubly stochastic permutation matrix $\mathbf{S}^t$ that satisfies

$$\sum_{l=1}^{N_{\text{ego}}+1} \mathbf{S}^t_{l,m} = 1, \forall m = 1, ..., N_{\text{ego}}, \quad \sum_{m=1}^{N_{\text{ego}}+1} \mathbf{S}^t_{l,m} = 1, \forall l = 1, ..., N_{\text{ego}}, \quad \mathbf{S}^t_{l,m} \geq 0. \tag{4.10}$$

[2] `https://github.com/rusty1s/pytorch_scatter`

[3] To improve training stability, row and column normalisations operate in log-space.

Here $\mathbf{S}^t_{l,m}$ represents the probability of $(\mathbf{p}^t_l, \mathbf{p}^1_m)$ being in correspondence. $\mathbf{p}^t_l$ is considered as an outlier and should be ignored during pose estimation if its slack column value $\mathbf{S}^t_{l,-1} \rightarrow 1$. We further mask $\mathbf{S}^t$ using a support matrix $\mathbf{I}^t$ computed from $\mathbf{D}^t$ as:

$$\mathbf{I}^t = \left(\mathbf{D}^t < s\right), \quad s = v \cdot \Delta t \,, \tag{4.11}$$

where v is the maximum speed and Δt is the interval between two frames. The final corresponding point $\phi(\mathbf{p}^t_l, \mathbf{P}^1)$ of $\mathbf{p}^t_l$ and its weight w^t_l are computed as

$$\phi(\mathbf{p}^t_l, \mathbf{P}^1) = (\mathbf{I}^t \odot \mathbf{D}^t)_{[l,:-1]}\mathbf{X}^1, \quad w^t_l = \sum_{m=1}^{N_{\text{ego}}} (\mathbf{I}^t \odot \mathbf{D}^t)_{l,m} \,, \tag{4.12}$$

with $\odot$ the Hadamard product. 4.4 is solved with the Kabsch algorithm. For a detailed derivation, please refer to Gojcic et al. (2021). The value of v is dataset-specific, we set it to 30 m/s for the *Waymo*, respectively 10 m/s for *nuScenes*.

Iterative refinement of TubeNet estimates. To improve the estimation of the transformation parameters for dynamic objects, we unroll TubeNet for two iterations, as often done in point cloud registration Yew and Lee (2020); Gojcic et al. (2020b). Specifically, for a dynamic object $\mathbf{X}_k$, we first estimate the initial rigid transformation $\mathbf{T}^{0,t}_k$ of the t^{th} frame $\mathbf{X}^t_k$ following 4.8. We then obtain the transformed points $\mathbf{X}^{t'}_k = \mathbf{T}^{0,t}_k \circ \mathbf{X}^t_k$. Next, we update the positional feature $\tilde{\mathbf{f}}^{t'}_{\text{pos}} = \text{PN}(\mathbf{X}^{t'}_k)$ and regress the residual transformation matrix $\mathbf{T}^{t,1}_k$ again, according to 4.8. The final transformation is $\mathbf{T}^{t,1}_k \cdot \mathbf{T}^{t,0}_k$. For better stability during training, the gradients between the two iterations are detached. We assign higher weight to the latter iteration to improve accuracy. The overall loss $\mathcal{L}^{\bullet}_{\text{obj}}$ is:

$$\mathcal{L}^{\bullet}_{\text{obj}} = 0.7 \cdot \mathcal{L}^{\bullet,0}_{\text{obj}} + \mathcal{L}^{\bullet,1}_{\text{obj}} \tag{4.13}$$

Scene reconstruction. To show the benefits of our method for downstream tasks, we use the points accumulated with different methods as a basis for 3D surface reconstruction, see Fig. 4.6 and 4.7. Specifically, we use the accumulated points as input to the Poisson reconstruction Kazhdan et al. (2006), implemented in Open3D Zhou et al. (2018). To estimate point cloud normals, the neighborhood radius is set to 0.5 m and the maximum number of neighbors is set to 128. The depth for the Poisson method is set to 10, and the reconstructed meshes are filtered by removing vertices with densities below the 15$^{\text{th}}$ percentile.

4.5.3. Loss functions and evaluation metrics

Weighted BCE loss. To compensate for class imbalance, we use a weighted BCE and compute the weights of each class on the fly. Specifically, for a mini-batch with N_{pos} positive and N_{neg}

negative samples, the associated weights w_{pos} and w_{neg} are computed as:

$$w_{\text{pos}} = \min(\sqrt{\frac{N_{\text{pos}} + N_{\text{neg}}}{N_{\text{pos}}}}, w_{\text{max}}) \qquad w_{\text{neg}} = \min(\sqrt{\frac{N_{\text{pos}} + N_{\text{neg}}}{N_{\text{neg}}}}, w_{\text{max}}) \ , \tag{4.14}$$

where w_{max} is the maximum weight of a class.[4] The final weighted BCE loss $\mathcal{L}_{\text{bce}}(\mathbf{x}, \overline{\mathbf{x}})$ is

$$\mathcal{L}_{\text{bce}}(\mathbf{x}, \overline{\mathbf{x}}) = \frac{1}{|\mathbf{x}|} \sum_{i=1}^{|\mathbf{x}|} w_i(\overline{\mathbf{x}}_i \log(\mathbf{x}_i) + (1 - \overline{\mathbf{x}}_i) \log(1 - \mathbf{x}_i)) \ , \tag{4.15}$$

with $\mathbf{x}$ and $\overline{\mathbf{x}}$ are predicted and ground truth labels, and w_i the weight of the i^{th} sample, computed as

$$w_i = \begin{cases} w_{\text{pos.}}, & \text{if } \overline{\mathbf{x}}_i = 1 \\ w_{\text{neg.}}, & \text{otherwise} \end{cases} \ . \tag{4.16}$$

Lovász-Softmax loss. The Jaccard index (ratio of Intersection over Union) is commonly used to measure segmentation quality. In the binary classification setting, we can set the ground truth labels as $\overline{\mathbf{x}}_i \in \{-1, 1\}$, then the Jaccard index of the foreground class J_1 is computed as

$$J_1(\overline{\mathbf{x}}, \mathbf{x}) = \frac{|\{\overline{\mathbf{x}} = 1\} \cap \{\text{sign}(\mathbf{x}) = 1\}|}{|\{\overline{\mathbf{x}} = 1\} \cup \{\text{sign}(\mathbf{x}) = 1\}|} \ , \quad J_1(\overline{\mathbf{x}}, \mathbf{x}) \in [0, 1] \ , \tag{4.17}$$

with $\mathbf{x}$ the prediction and $\text{sign}()$ the sign function. The corresponding loss $\Delta_{J_1}(\overline{\mathbf{x}}, \mathbf{x})$ to minimise the empirical risk is

$$\Delta_{J_1}(\overline{\mathbf{x}}, \mathbf{x}) = 1 - J_1(\overline{\mathbf{x}}, \mathbf{x}) \ . \tag{4.18}$$

However, this is not differentiable and cannot be directly employed as a loss function. The authors of Berman et al. (2018) have proposed to optimise it using a Lovász extension. The Lovász extension $\dddot{\Delta}$ of a set function Δ is defined as:

$$\dddot{\Delta}(\mathbf{m}) = \sum_{i=1}^{p} \mathbf{m}_i \, g_i(\mathbf{m}) \ , \tag{4.19}$$

with

$$g_i(\mathbf{m}) = \Delta(\{\pi_1, \ldots, \pi_i\}) - \Delta(\{\pi_1, \ldots, \pi_{i-1}\}), \tag{4.20}$$

where π denotes a permutation that places the components of $\mathbf{m}$ in decreasing order. Considering $\mathbf{m}_i = \max(1 - \mathbf{x}_i \overline{\mathbf{x}}_i, 0)$, the Lovász-Softmax loss $\mathcal{L}_{ls}(\overline{\mathbf{x}}, \mathbf{x})$ is defined as

$$\mathcal{L}_{ls}(\overline{\mathbf{x}}, \mathbf{x}) = \dddot{\Delta}_{J_1}(\mathbf{m}). \tag{4.21}$$

Inlier loss. Previous works Yew and Lee (2020); Gojcic et al. (2021) have observed that the entropy-regularized optimal transport Cuturi (2013) has a tendency to label most points as

[4] We find that in some extreme cases, there are very few (< 10) positive samples and way more negative samples. We thus bound w_{max} at 50 to ensure stability.

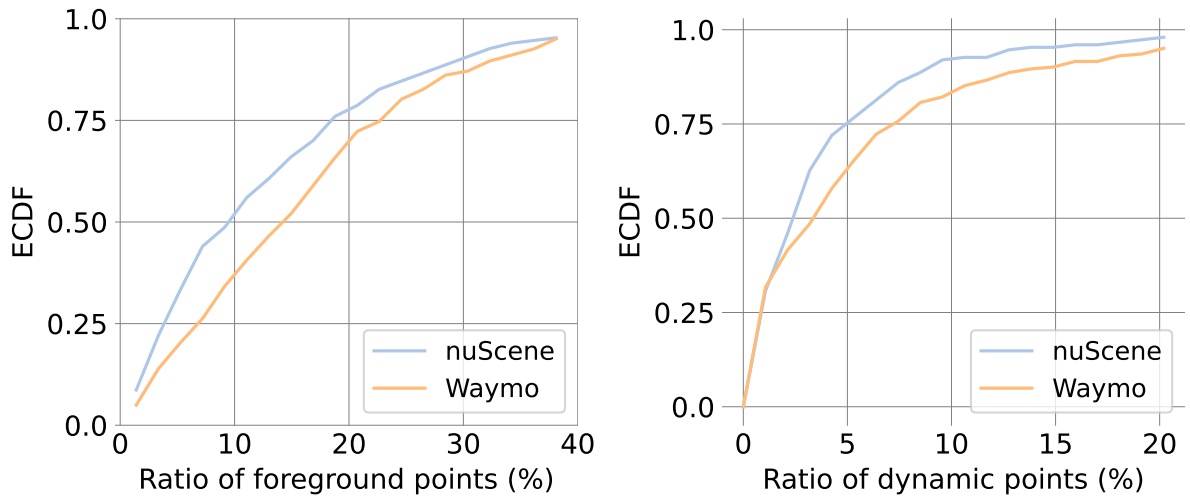


Figure 4.3.: *ECDF* curve of points lying on foreground objects and on dynamic objects, for both the *Waymo* and *nuScenes* datasets.

outliers. To alleviate this issue, we follow Yew and Lee (2020); Gojcic et al. (2021) and use an inlier loss $\mathcal{L}_{\text{inlier}}$ on the matching matrix $\mathbf{D}^t$, designed to encourage inliers. The inlier loss is defined as

$$\mathcal{L}^t_{\text{inlier}} = \frac{1}{2N_{\text{ego}}}(2N_{\text{ego}} - \sum_{l=1}^{N_{\text{ego}}}\sum_{m=1}^{N_{\text{ego}}} \mathbf{D}^t_{l,m}) \ . \tag{4.22}$$

Instance association metrics. To quantitatively measure the spatio-temporal instance association quality, we report weighted coverage (*WCov*) as well as recall and precision at a certain threshold. Given the ground truth clusters $\mathcal{G}$ and the estimated clusters $\mathcal{O}$, recall measures the ratio of clusters in $\mathcal{G}$ that have an overlap above some threshold with a cluster in $\mathcal{O}$, while precision does the same in the opposite direction. Weighted coverage $WCov(\mathcal{G}, \mathcal{O})$ is computed as

$$WCov(\mathcal{G}, \mathcal{O}) = \sum_{i=1}^{|\mathcal{G}|} \frac{1}{|\mathcal{G}|} w_i \max_j \text{IoU}(r_i^G, r_j^0), \qquad w_i = \frac{|r_i^G|}{\sum_k |r_k^G|} \ , \tag{4.23}$$

where r_i^G and r_j^O are clusters from $\mathcal{G}$ and $\mathcal{O}$, and $\text{IoU}(r_i^G, r_j^O)$ denotes the overlap between two clusters.

ECDF. The Empirical Cumulative Distribution Function (ECDF) measures the distribution of a set of values:

$$\text{ECDF}(x) = \frac{\left|\{o_i < x\}\right|}{|O|} \ , \tag{4.24}$$

where $O = \{o_i\}$ is a set of samples and $x \in [\min\{O\}, \max\{O\}]$.

4.5.4. Dataset analysis

In total, we have 150, respectiverly 202 scenes as held-out test sets in *nuScenes* and *Waymo*. The *ECDF* curve of points belonging to foreground and dynamic objects are shown in Fig. 4.3.

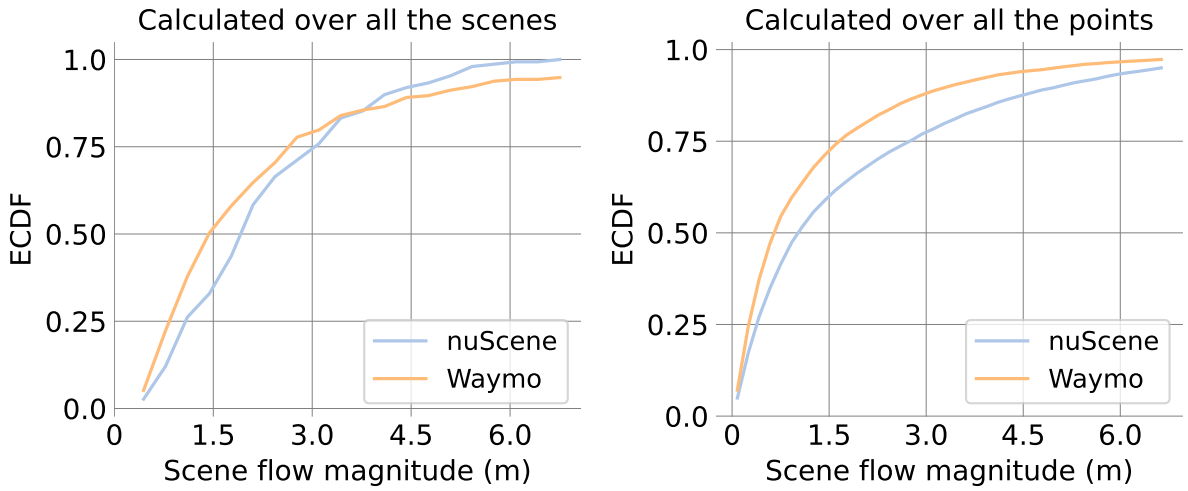


Figure 4.4.: Scene flow magnitudes of dynamic objects in the *Waymo* and *nuScenes* datasets.

As can be seen, the ratios of foreground and dynamic points span a large range (40% and 20%). Recalling that the scene flow estimation performance of the dynamic parts falls far behind that of the static parts (**??**), this large range of ratios of dynamic objects hints at different difficulties across the scenes. The median fractions of foreground points are 16.2%/9.4% in *Waymo*/*nuScenes*, the median fractions of points on moving objects are 3.5%/2.4%. In other words, roughly 75% of all foreground objects are static. This motivates our strategy to start with motion segmentation, so as to make explicit the large static component (including many objects that could move) whose scene flow is identical to the ego-motion.

In Fig. 4.4, we show the *ECDF* curve of scene flow magnitudes (L_2-norm of scene flow vectors) for the dynamic portions of the two datasets. The motions span a large range, but 75% of the flow vectors are of moderate magnitude $<$3 m. *nuScenes* has slightly larger overall flow magnitudes than *Waymo*, but *Waymo* contains more instances of large motions (Fig. 4.4 (left)).

4.5.5. Additional results

Results averaged over scenes. In **??** we report evaluation metrics calculated over all the points in the test set. However, this does not fully reveal the difficulties encountered in different scenes. Here, we first calculate evaluation metrics per scene, then report the average over scenes in Tab. 4.1. For *EPE avg.*, we additionally report the standard deviations. We can see that for both static and dynamic parts, all methods have large standard deviations, which indicates varying difficulty of the scenes, as well as gross errors from challenging samples. Our model still achieves the smallest flow errors and standard deviations under this evaluation setting, for both datasets .

Spatio-temporal instance association. We plot instance association metrics at different thresholds in Fig. 4.5. As can be seen, offset prediction improves association recall and precision by $>$5%, across a range of thresholds. Such improvement becomes more significant as one increases the *IoU* threshold, reaching $\approx$10% at $IoU = 0.9$. We conclude that offset prediction

Dataset	Method	Static part				Dynamic foreground			
		EPE avg.↓	AccR↑	AccS↑	ROutlier↓	EPE avg. ↓	AccR↑	AccS↑	ROutliers ↓
Waymo	PPWC-Net Wu et al. (2020b)	0.475 ± 0.543	35.0	14.2	13.5	0.658 ± 0.696	27.1	7.9	22.9
	FLOT Puy et al. (2020)	0.381 ± 0.516	68.8	51.8	13.0	0.772 ± 0.711	30.1	11.2	31.9
	WsRSF Gojcic et al. (2021)	1.415 ± 1.352	34.6	23.0	56.9	1.764 ± 1.744	21.0	8.6	61.6
	NSFPrior Li et al. (2021b)	0.159 ± 0.231	87.1	73.5	4.3	0.355 ± 0.456	63.7	41.3	14.3
	Ours	**0.088** ± 0.237	**91.6**	**81.9**	**2.3**	**0.169** ± 0.259	**76.8**	**52.9**	**5.3**
nuScenes	PPWC-Net Wu et al. (2020b)	0.488 ± 0.402	34.2	12.7	17.5	0.784 ± 0.547	22.8	6.9	35.0
	FLOT Puy et al. (2020)	0.597 ± 0.582	53.3	35.1	26.6	1.156 ± 0.714	13.2	3.7	56.5
	WsRSF Gojcic et al. (2021)	0.658 ± 0.483	47.5	31.1	31.5	0.925 ± 0.627	29.8	15.0	42.2
	NSFPrior Li et al. (2021b)	0.501 ± 0.344	57.8	37.7	21.3	0.743 ± 0.537	39.1	19.9	31.1
	Ours	**0.226** ± 0.206	**72.3**	**46.7**	**7.4**	**0.394** ± 0.26	**47.8**	**22.7**	**17.3**

Table 4.1.: Scene flow estimation results on *Waymo* and *nuScenes* datasets. Numbers are averaged over all test scenes.

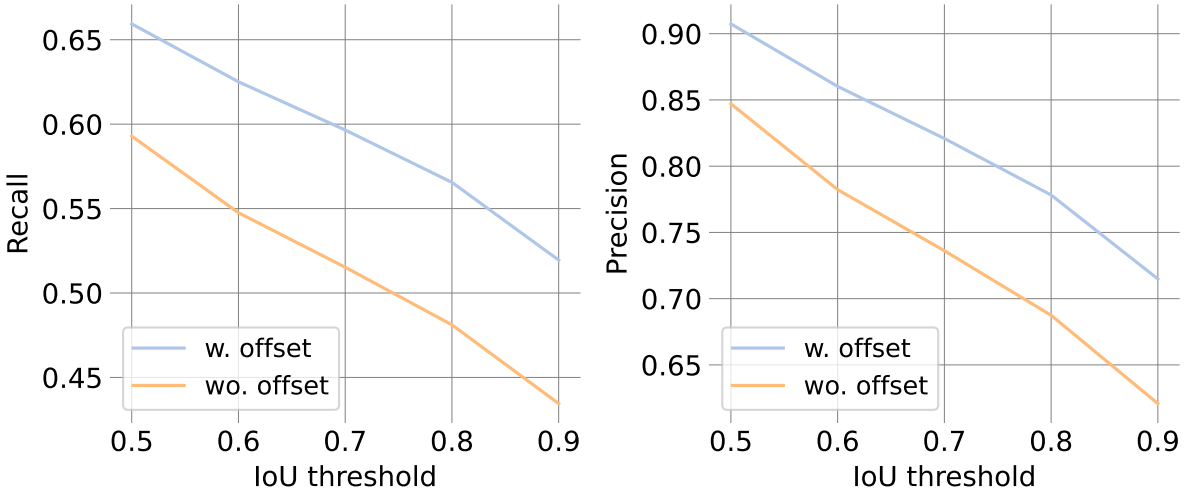


Figure 4.5.: Spatio-temporal instance association performance on *Waymo* with and without offset prediction.

is important to retain high-quality spatio-temporal instances, which can subsequently improve the accuracy of motion modelling for the dynamic parts (*AccS* increases by 9.2% in **??**).

Dynamic object motion modelling. We additionally compare the proposed TubeNet to two baseline methods. We naively align each frame $\mathbf{X}_k^t$ $(t > 1)$ of an instance $\mathbf{X}_k$ to frame $\mathbf{X}_k^0$ by translating the centroids, and term this method *center*. For *center+ICP* we refine the simple translational alignment by a subsequent ICP. The detailed comparison is shown in Tab. 4.2. Our learned TubeNet achieves the best performance on both datasets. The improvement is larger on the challenging *nuScenes* data, where the point clouds are sparser and less complete, so centroids computed from partial observations are not an accurate proxy for the object location. Our learned TubeNet can implicitly exploit prior knowledge about object shape and surface-level correspondence, leading to more robust and accurate motion modelling.

		EPE avg.↓	EPE med.↓	AccS↑	AccR↑	ROutliers↓
Waymo	center	0.265	0.095	36.9	62.9	9.9
	center + ICP	0.212	0.047	61.2	80.0	7.7
	Ours	0.197	0.062	53.3	77.5	5.9
	Ours + ICP	**0.173**	**0.043**	**69.1**	**86.9**	**5.1**
nuScenes	center	0.553	0.258	13.5	32.7	28.2
	center + ICP	0.525	0.179	23.8	43.7	25.5
	Ours	0.301	0.146	26.6	53.4	**12.1**
	Ours + ICP	**0.301**	**0.135**	**32.7**	**56.7**	13.7

Table 4.2.: Comparison to centroid-based motion estimation baseline.

4.5.6. Qualitative results

We show additional qualitative results in Fig. 4.6 and Fig. 4.7. Benefiting from the explicit *multi-body* assumption, our model achieves accurate scene flow estimation of both static parts (Fig. 4.6 (1) and Fig. 4.7 (2)) and dynamic parts (Fig. 4.6 (3) and Fig. 4.7(1)). Errors in the automatically generated pseudo-ground truth are shown in Fig. 4.7(3), in this case our model achieves more accurate flow estimation and reconstruction.

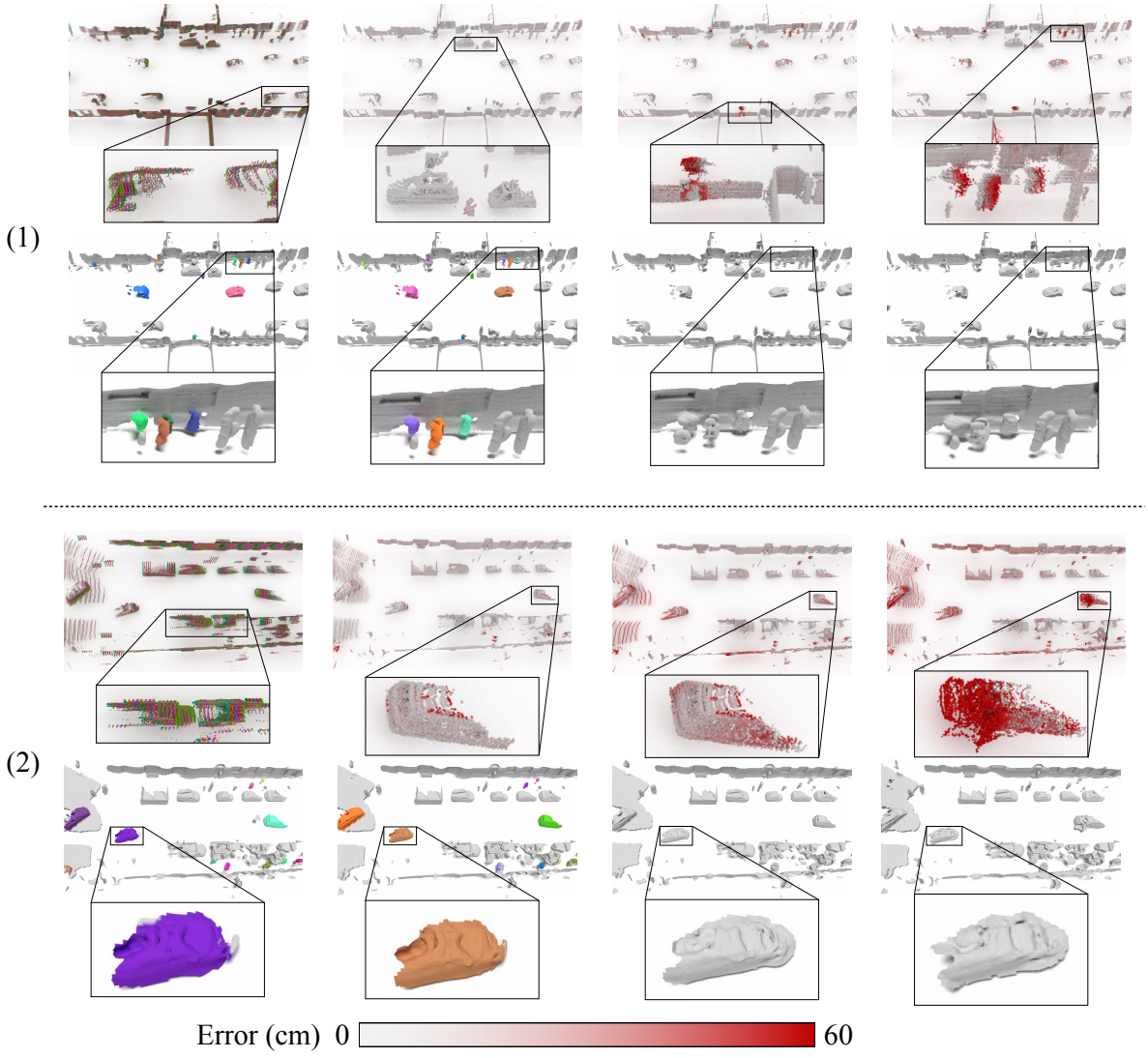


Figure 4.6.: Qualitative results showing scene flow estimation (top) and surface reconstruction (bottom) for three example scenes from the *Waymo* dataset.

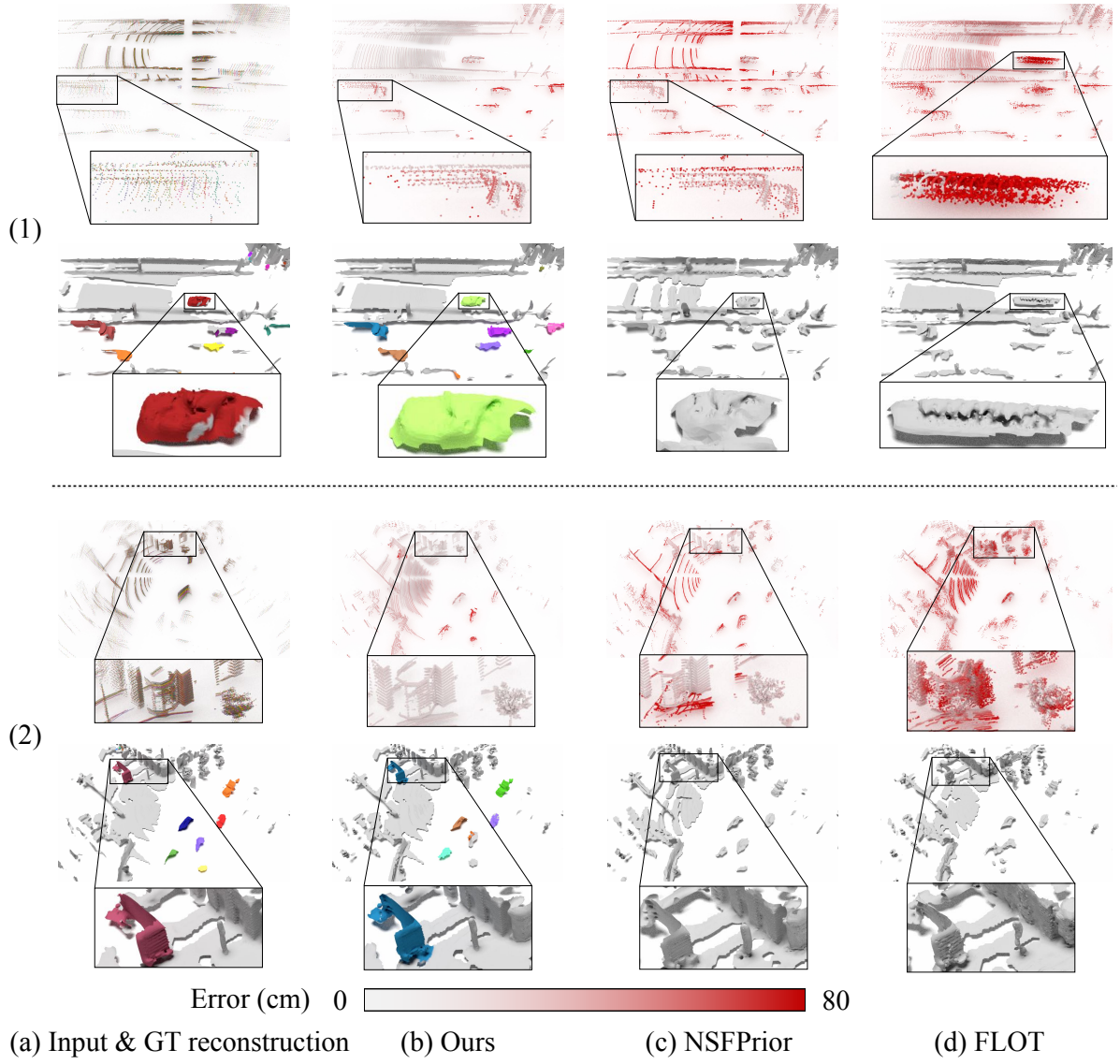


Figure 4.7.: Qualitative results showing scene flow estimation (top) and surface reconstruction (bottom) for three example scenes from the *nuScenes* dataset.

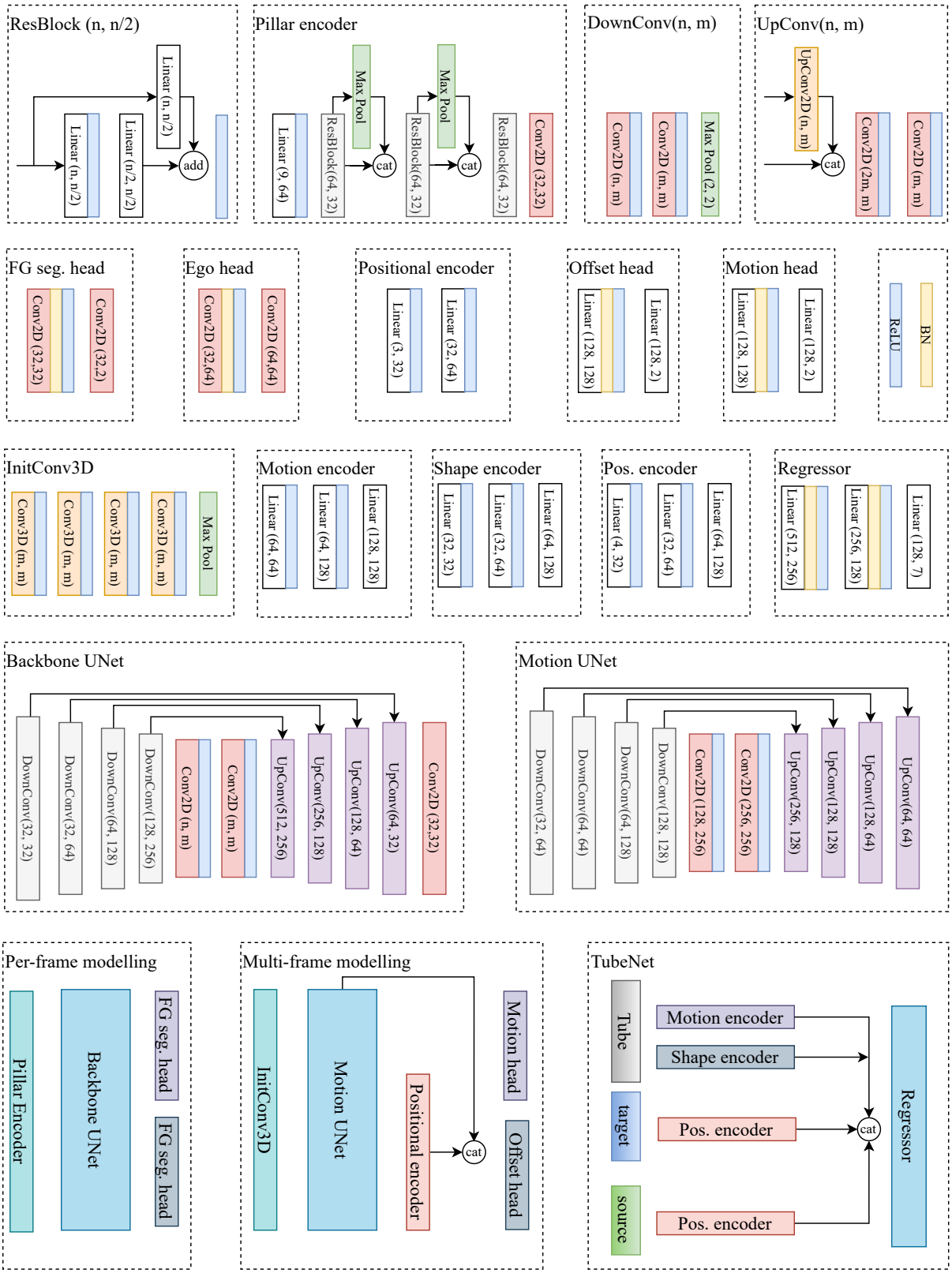


Figure 4.8.: Detailed network architecture. All convolutional layers have kernel size 3×3. (n, m) in Conv, UpConv, DownConv, and Linear layers denote the input and output feature dimensions.

5 Neural LiDAR Fields for Novel View Synthesis

Shengyu Huang, Zan Gojcic, Zian Wang, Francis Williams, Yoni Kasten, Sanja Fidler, Konrad Schindler, Or Litany
IEEE International Conference on Computer Vision, 2023

(Author version; for typeset version please refer to the original conference paper.)

Abstract

We present Neural Fields for LiDAR (NFL), a method to optimise a neural field scene representation from LiDAR measurements, with the goal of synthesizing realistic LiDAR scans from novel viewpoints. NFL combines the rendering power of neural fields with a detailed, physically motivated model of the LiDAR sensing process, thus enabling it to accurately reproduce key sensor behaviors like beam divergence, secondary returns, and ray dropping. We evaluate NFL on synthetic and real LiDAR scans and show that it outperforms explicit reconstruct-then-simulate methods as well as other NeRF-style methods on LiDAR novel view synthesis task. Moreover, we show that the improved realism of the synthesized views narrows the domain gap to real scans and translates to better registration and semantic segmentation performance.

5.1. Introduction

The goal of novel view synthesis is to generate a view of a 3D scene, from a viewpoint at which no real sensor image has been captured. This offers the possibility to observe *real* scenes from a *virtual*, unobserved perspective. Among other applications, it has tremendous potential for autonomous driving: synthetic novel views may be used to train and test perception algorithms across a wider range of viewing conditions, thus enhancing robustness and generalization. Moreover, novel view synthesis becomes critical when the desired viewpoints are not known in advance, *e.g.*, during training of a planning module whose decisions determine future vehicle locations.

Neural radiance fields (NeRFs) have led to unprecedented visual quality when synthesizing novel camera views Mildenhall et al. (2020); Barron et al. (2021); Yu et al. (2021); Müller et al. (2022). These methods represent the 3D scene in form of continuous density and radiance fields, from which images can be generated through volume rendering, mimicking the image acquisition process. The inductive bias of neural networks imparts NeRFs the ability to interpolate complex lighting and reflectance behaviours with a high degree of realism.

While most prior works focused on synthesizing camera views, 3D perception in the autonomous driving context typically relies partly (or even exclusively) on LiDAR measurements. Synthesizing realistic LiDAR scans from novel viewpoints thus has a lot of potential for data augmentation and closed-loop testing of autonomous navigation systems.

The problem of synthesizing novel LiDAR views has previously been addressed in two stages Manivasagam et al. (2020). First, extract an explicit surface representation such as surfels or a triangular mesh from the scanned point clouds. Then, simulate LiDAR measurements from a novel viewpoint by casting rays and intersecting them with the surface model. Like for images, explicit reconstruction (which is not optimised towards the subsequent synthesis step) suffers from discretization artifacts and introduces noticeable errors Waechter et al. (2014). Moreover, the rendering assumes an idealised ray model and neglects the divergence of the LiDAR beams, which causes frequent second returns from distant surfaces.

Here, we instead build on a main insight of NeRF Mildenhall et al. (2020): directly optimizing an implicit scene representation for novel view synthesis can produce more realistic outputs than the reconstruct-then-simulate approach. Specifically, we propose Neural Fields for LiDAR (NFL), a NeRF-style representation for synthesizing novel LiDAR viewpoints.

Several NeRF extensions have utilized range measurements as additional supervision, and have shown that constraining the scene geometry more tightly can yield better (camera) view synthesis Deng et al. (2021); Rematas et al. (2021). Yet, the output of those methods are synthetic images, not LiDAR scans, consequently they have not paid attention to effects specific to LiDAR sensing: a laser scanner does *not* directly sense range, rather it measures the returned light energy per ray and determines the range based on the waveform. This includes the possibilities that there are multiple returns[1] from the emitted ray, or no return at all.

Our formulation closely adheres to the principles of the LiDAR measurement process and incorporates them into the neural field framework. Specifically, we (i) **devise volume**

[1] In principle there can be >2 returns, but automotive LiDAR sensors typically record the first two echos.

rendering for LiDAR sensors; (ii) **incorporate beam divergence** and (iii) **propose truncated volume rendering** to account for secondary returns and improve range prediction.

We evaluate our method on both synthetic and real LiDAR data. To this end, we (iv) **develop a LiDAR simulator** for synthesizing scenes from 3D assets that serve as a test bed for viewpoints far from the original scan locations, and to study the effect of different scan patterns. Real data from the Waymo Sun et al. (2020a) dataset is used to evaluate NFL against real scans at held-out viewpoints, including real-world intensities, ray drops and secondary returns. Additionally, we (v) **propose a novel closed-loop evaluation protocol** that leverages real data to evaluate view synthesis in challenging views. As an end-to-end test for downstream tasks, we further evaluate the performance of state-of-the-art segmentation and registration networks when trained on real scans and tested on novel views generated by NFL.

5.2. Related Work

LiDAR simulation.. Simulating realistic LiDAR data is useful for training perception models. Different from real-world LiDAR data that requires annotation efforts, simulated data can be automatically generated with ground truth labels, *e.g*. object bounding boxes and semantic segmentation. Unrealistic LiDAR simulation will prevent the trained models from generalizing to real data. Traditional simulation engines, such as those proposed in Dosovitskiy et al. (2017); Koenig and Howard (2004), require the specification of sensor parameters and 3D scene assets and use ray-casting methods for simulation. Although these point clouds can accurately represent scene geometry, they often exhibit a discrepancy, or "domain gap", compared to real data, due to the lack of modeling for sensor noise, such as ray drop and Gaussian beam. Furthermore, this approach relies heavily on the creation of 3D scene assets, which can be time-consuming and expensive. To address these challenges, LiDARsim Manivasagam et al. (2020) reconstructs the static and dynamic scene assets from real data using surfel Pfister et al. (2000) representation and models the ray-drop pattern for improved realism. BaiduSim Fang et al. (2020a) proposes a probability map to model scene compositions in order to reduce the domain gap. Most recent work Guillard et al. (2022) learns to enhance existing simulated LiDAR intensity and ray-drop patterns, using the available corresponding RGB images.

Weather conditions, such as fog or rain, can significantly impact the quality of LiDAR data, and downstream models trained solely on ideal weather conditions may fail to generalize to these effects. Recent methods and datasets Hahner et al. (2021, 2022); Kilic et al. (2021); Bijelic et al. (2020) have been proposed to address this issue. SnowSIM Hahner et al. (2022) and FogSIM Hahner et al. (2021) sample snow particles and model the impulse response from atmospheric attenuation, respectively, to alter the range measurements of each ray. Other approaches Shih et al. (2022); Kilic et al. (2021); Kurup and Bos (2021) simulate LiDAR data on rainy days and the spray effects, in a similar fashion.

NeRF for Novel View Synthesis.. NeRF Mildenhall et al. (2020) maps 5D position and direction to density and radiance scene values, and uses volume rendering Max (1995); Max and Chen (2005) to estimate pixel color. This technique has proven effective for generating realistic images at unseen camera views. Many methods have been proposed to improve robustness

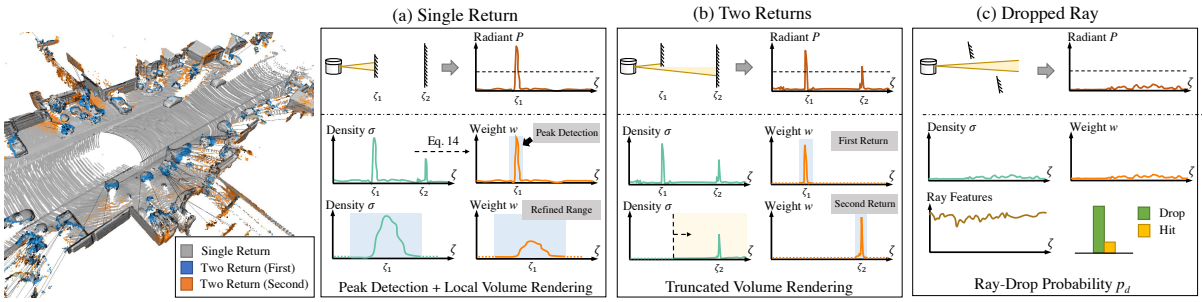


Figure 5.1.: Left: real LiDAR scan demonstrating key LiDAR return properties: a single return and two returns (first return shown in blue and second return in orange). Right: NFL models the waveform and accurately reproduces these properties. (a) Top: the LiDAR energy is fully scattered by the first surface. Bottom: NFL estimates range via peak detection on the computed weights w followed by volume rendering based range refinement. (b) Top: secondary returns resulting from a beam hitting two surfaces. Bottom: NFL employs beam divergence and a truncated volume rendering to estimate the second return. (c) Top: beams that do not hit a surface do not return detectable signal. Bottom: NFL utilizes geometric and semantic features to predict the ray drop probability. Refer to section 5.4.3 for more details.

to camera poses Lin et al. (2021); Chng et al. (2022), handle dynamics Ost et al. (2021a); Pumarola et al. (2021a), anti-alias Zhang et al. (2020); Barron et al. (2021, 2022), and speed up optimisation Liu et al. (2020); Yu et al. (2021); Müller et al. (2022) *etc*. Despite its high-quality novel view synthesis capacity, the underlying geometry of NeRF is considered inaccurate and noisy Oechsle et al. (2021), making it less favoured for geometry reconstruction, especially in sparse-views settings. Oechsle et al. (2021); Yariv et al. (2021); Wang et al. (2021) address this challenge by using implicit surface representations, and defining the density functions based on them to enabling volume rendering. DS-NeRF Deng et al. (2021) and DenseDS-NeRF Roessle et al. (2021) use sparse depth supervision from SfM Schönberger and Frahm (2016) points to regularise the density field. Urban Radiance Field Rematas et al. (2021) leverages LiDAR data for depth supervision.

Neural fields beyond regular cameras.. Neural fields are a natural and continuous representation Xie et al. (2022) for spatio-temporal information including SDFs Park et al. (2019), occupancy Mescheder et al. (2019) and radiance field Mildenhall et al. (2020) *etc*. While in its original form, NeRF Mildenhall et al. (2020) performs novel view synthesis using tonemapped low dynamic range images, RawNeRF Mildenhall et al. (2022) extends it to operate over the high dynamic range images, enabling additional adjustments to focus, exposure, and tonemapping. Törf Attal et al. (2021) incorporates the image formation model for continuous-wave Time-of-Flight (ToF) cameras into NeRF, allowing it to jointly process RGB and ToF sensor data and improve reconstruction robustness to large motions. EventNeRF Rudnev et al. (2022) and ENeRF Klenk et al. (2022) optimise the scene representation for Novel View Synthesis (NVS) from sparse event streams that contain asynchronous per-pixel brightness change signals. Other works Qadri et al. (2022); Luo et al. (2022) explore the use of acoustic signals for surface reconstruction or NVS.

5.3. Background

We start by reviewing the principles of volume rendering (§ 5.3.1) and the sensor model for LiDAR (§ 5.3.2). This sets the stage for the proposed formulation of Neural LiDAR Fields (Sec. 5.4).

5.3.1. Volume rendering for passive sensors

In the following, we provide a brief summary of camera-based volume rendering as used by NeRF Mildenhall et al. (2020); Tagliasacchi and Mildenhall (2022). This will serve as the basis to derive volume rendering equations for the active LiDAR sensor.

Density and transmittance.. For a ray $\mathbf{r}(\mathbf{o}, \mathbf{d})$ emitted from the origin $\mathbf{o} \in \mathbb{R}^3$ in direction $\mathbf{d} \in \mathbb{R}^3$, the *density* σ_ζ at range ζ is a scalar function that indicates the differential likelihood of hitting a reflective particle at position $\mathbf{r}_\zeta = \mathbf{o} + \zeta\mathbf{d}$. *Transmittance* T_ζ indicates the probability of traversing the interval $[0, \zeta)$ without hitting anything. Taking a differential step $d\zeta$ along the ray, the probability of *not* hitting anything is $T_{\zeta+d\zeta} = T_\zeta \cdot (1 - \sigma_\zeta d\zeta)$. Integrating over an interval $[\zeta_0, \zeta)$ yields the probability $T_{\zeta_0 \to \zeta}$ of traversing the interval unhindered,

$$T_{\zeta_0 \to \zeta} \equiv \frac{T_\zeta}{T_{\zeta_0}} = \exp\left(- \int_{\zeta_0}^{\zeta} \sigma_t dt \right) , \tag{5.1}$$

leading to the decomposition: $T_\zeta = T_{0 \to \zeta_0} \cdot T_{\zeta_0 \to \zeta}$.

Integration over homogeneous media.. Assuming a homogeneous medium along the ray segment $[\zeta_j, \zeta_{j+1}]$ with constant radiance $\mathbf{c} \in \mathbb{R}^3$ and density σ, the accumulated radiance from that segment evaluates to

$$\mathbf{c}(\zeta_j \to \zeta_{j+1}) = \mathbf{c}_{\zeta_j} \int_{\zeta_j}^{\zeta_{j+1}} T_{\zeta_j \to \zeta} \cdot \sigma_\zeta \, d\zeta = \alpha_{\zeta_j} \mathbf{c}_{\zeta_j} , \tag{5.2}$$

with $\alpha_{\zeta_j} = 1 - \exp\big(-\sigma_{\zeta_j}(\zeta_{j+1} - \zeta_j)\big)$ being the *opacity*.

Volume rendering.. By discretizing the ray into N segments with piecewise constant densities and radiance values, we obtain the total irradiance (color to be rendered):

$$\mathbf{c} = \sum_{j=1}^{N} \int_{\zeta_j}^{\zeta_{j+1}} T_\zeta \cdot \sigma_\zeta \mathbf{c}_\zeta \, d\zeta = \sum_{j=1}^{N} w_j \mathbf{c}_{\zeta_j} , \tag{5.3}$$

where w_j is the *weight* for the j-th segment:

$$w_j = \alpha_{\zeta_j} \prod_{k=1}^{j-1} (1 - \alpha_{\zeta_k}) . \tag{5.4}$$

5.3.2. LiDAR model

LiDAR emits laser beam pulses and determines the distance from the sensor to the nearest reflective surface by measuring the time of flight. Often the LiDAR beams are pictured as ideal straight-line segments ending on a 3D surface point. In reality, things are more complicated: real lasers emit a pulse with non-zero divergence and finite pulse width, while real receivers employ signal processing techniques like radiant thresholding and binning to detect the return. This leads to phenomena such as discretization errors, over- and underestimation biases (*c.f.* Fig. 5.2), and multiple returns from one beam (or no return at all). In the following, we discuss key aspects of the LiDAR acquisition process and explain the effects that emerge, which inspire our model design. We also built a LiDAR simulator that accounts for these mechanisms, see § 5.5.1.

Beam divergence.. LiDAR beams diverge as they travel away from the sensor. The size of laser beams can become wider over distance, and typically not negligible in street scenes. Consequently, the illuminated area grows and the irradiance (radiant power per area) decreases with increasing range. The size of the beam's footprint is characterised by the divergence angle $(2\gamma_0)$ and the range ζ. Let $\mathbf{r}^\gamma$ be an ideal ray within the beam's cross-section, $\gamma \leq \gamma_0$, then its irradiance $E(\zeta, \gamma)$ at range ζ can be approximated by a Gaussian function in the ray coordinate system Wagner et al. (2006):

$$E(\zeta, \gamma) = \frac{2I_0}{\pi(\gamma_0\zeta)^2} g(\gamma), \quad g(\gamma) = \exp\left(-2\frac{\gamma^2}{\gamma_0^2}\right), \tag{5.5}$$

where I_0 is the pulse peak power.

Pulse waveform.. When the emitted LiDAR pulse returns to the sensor, the range to the reflective surface can be determined from its travel time and the speed of light c. Since the pulse has finite duration τ_H, the time of return is found by analysing the received intensity profile. The transmitted pulse power over time can be characterised as Carlsson et al. (2001):

$$P_e(t) \propto \left(\frac{t}{\tau}\right)^2 \exp\left(-\frac{t}{\tau}\right), \quad \tau = \frac{\tau_H}{1.75} \, . \tag{5.6}$$

The range-dependent received radiant power $P(\zeta)$ is the result of convolving the pulse power with the systems impulse response $H(\zeta)$ Rasshofer et al. (2011); Hahner et al. (2021, 2022):

$$P(\zeta) = \int_0^{2\zeta/c} P_e(t) H(\zeta - \frac{ct}{2}) \, dt \, , \tag{5.7}$$

where the impulse response $H(\zeta)$ is a composition of the target and the receiver responses: $H(\zeta) = H_T(\zeta) H_C(\zeta)$. Assuming a Lambertian surface, the target response due to a surface located at range ζ_0 depends on the incidence angle θ and the reflectance ρ:

$$H_T(\zeta) = \frac{\rho}{\pi} \cos(\theta) \delta(\zeta - \zeta_0) \, , \tag{5.8}$$

with $\delta(\cdot)$ the Dirac delta function. The receiver response $H_C(\zeta)$ is computed by integrating

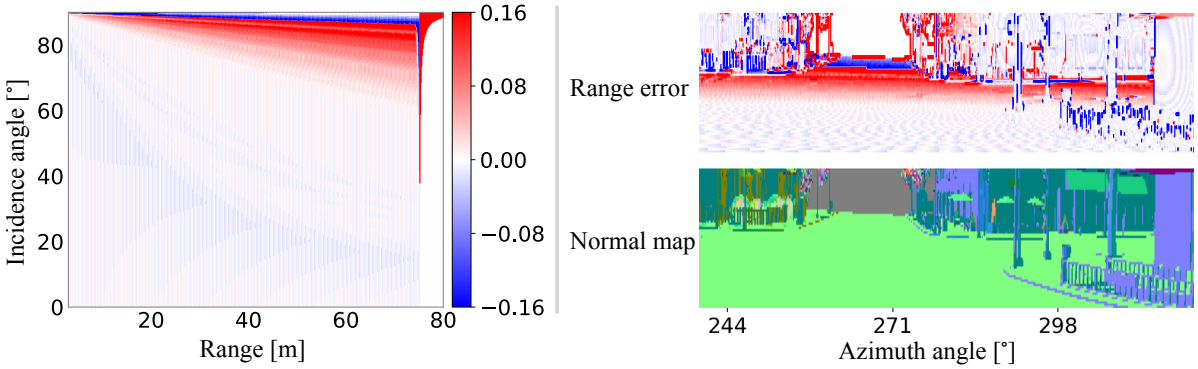


Figure 5.2.: The range accuracy of the LiDAR sensor is affected by waveform discretization and beam divergence. The LiDAR sensor has a tendency to overestimate range in high incidence angle regime, which becomes increasingly pronounced at higher range regimes (left). This is also reflected on *TownReal* dataset (right).

over the solid angle spanned by the receiver's effective area A_e:

$$H_C(\zeta) = T_\zeta^2 \frac{A_e}{\zeta^2} \ , \tag{5.9}$$

where $T_\zeta \in [0, 1]$ is the one-way transmittance, squared to account for the two-way trip.

Beam discretization.. In practice, we follow Winiwarter et al. (2022) and approximate the Gaussian beam profile using $M = 37$ rays that are radially distributed around the central ray with different divergence angles γ_i. The total radiant power $P(\zeta)$ is the weighted sum over those rays: $P(\zeta) = \sum_{i=1}^{M} g(\gamma_i) P_i(\zeta)$. Taking into account the beam divergence is important to reproduce two important phenomena: range biases and multiple returns, see Fig. 5.1 and Fig. 5.2. As different rays hit a slanted surface at different ranges the integrated waveform peak may shift, causing over- or underestimations. Along object edges, rays within the same beam may hit different surfaces, causing multiple peaks (respectively, range readings), in the return waveform.

Range estimation..

One common approach to estimate the surface range from the received waveform is to locate its peak. To that end the signal is discretized in time to obtain a histogram, and local maxima above a certain threshold are declared detections Winiwarter et al. (2022). The associated range values are then corrected to remove known biases stemming from the pulse waveform (*c.f.* Eq. (5.6)) and, optionally, biases due to the radiant power Winiwarter et al. (2022). By modeling the binning and thresholding procedure one can reproduce further LiDAR behaviors: systematic discretization errors in the range resolution (*c.f.* Fig. 5.2), and the dropping of rays with low returned power (*c.f.* Fig. 5.1).

5.4. LiDAR Novel View Synthesis

We now turn to constructing a neural field model tailored for LiDAR scans, along with a differentiable volume rendering scheme to enable LiDAR novel view synthesis. We first formulate the problem setting, then set up a corresponding neural scene representation (§ 5.4.1) and derive volume rendering for active sensing (§ 5.4.2). Finally, we describe the rendering procedure used to synthesize novel views (§ 5.4.3) and our optimisation scheme (§ 5.4.4).

Problem setting.. Consider a collection of LiDAR scans $\mathcal{X} = \{\mathbf{X}_v\}_{v=1}^{n_v}$ captured by a moving sensor (*e.g.*, mounted on a vehicle). Each scan $\mathbf{X}_v$ is associated with a sensor pose $\mathbf{T}_v \in \mathrm{SE}(3)$ and consists of n_r rays. Every ray $\mathbf{r}(\mathbf{o}, \mathbf{d})$ records observations $(\zeta_1, e_1, p_d, p_s, \zeta_2, e_2)$: the range ζ_1 and intensity e_1 of the first return; a ray drop flag $p_d \in \{0, 1\}$; a two-return mask $p_s \in \{0, 1\}$; and range ζ_2 and intensity e_2 values of the second return. Our goal is to reconstruct a (continuous) volumetric representation of the scene in terms of density σ and reflectance ρ, from which we can subsequently render virtual LiDAR scans $\mathbf{X}_{tgt}$ from novel sensor poses $\mathbf{T}_{tgt}$.

5.4.1. Neural scene representation

We encode the scene as a neural field $F : (\mathbf{x}, \mathbf{d}) \mapsto (\sigma, \rho, p_d)$ that takes as input a location $\mathbf{x} \in \mathbb{R}^3$ and viewing direction $\mathbf{d} \in \mathbb{R}^3$, and returns a density σ, a reflectance ρ and a ray drop probability p_d. We found it beneficial to additionally return also a local contribution $p_d \in [0, 1]$ to the probability of ray drop, which will be discussed below. Technically, we use a hash encoding Müller et al. (2022) to map coordinates $\mathbf{x}$ to positional features $\mathbf{f}_{\text{pos}} \in \mathbb{R}^{32}$ and project the view direction onto the first 16 coefficients of the spherical harmonics basis, $\mathbf{f}_{\text{dir}} \in \mathbb{R}^{16}$. The neural field is parameterized by four Multi-Layer Perceptrons (MLPs): $[\sigma; \mathbf{f}_{\text{geo}}] = f_\sigma(\mathbf{f}_{\text{pos}})$ regresses density and extracts an additional geometry feature $\mathbf{f}_{\text{geo}} \in \mathbb{R}^{15}$ that supports the other networks; $\rho = f_\rho(\mathbf{f}_{\text{geo}}, \mathbf{f}_{\text{dir}})$ regresses reflectance; $p_d = f_{\text{drop}}(\mathbf{f}_{\text{geo}}, \mathbf{f}_{\text{dir}})$ classifies whether a ray drop occurs; and $p_s = f_{\text{sr}}(\mathbf{f}_{\text{beam}})$ classifies the existance of a second return. The feature $\mathbf{f}_{\text{beam}}$ will be detailed in § 5.4.3.

5.4.2. Volume rendering for LiDAR rays

In contrast to passive sensors like cameras that rely on ambient illumination, LiDAR actively illuminates the scene and measures the back-scattered radiance. This two-way transmittance alters the volume rendering formulation.

Radiant power integration.. As discussed in § 5.3.2 the radiant power along a LiDAR ray is a delta function that is non-zero only at reflecting surfaces. To incorporate this forward model into the volumetric representation we combine Eq. (6.3) and Eq. (5.9) to obtain the probabilistic radiant power:

$$P_\zeta = C \frac{T_\zeta^2 \cdot \sigma_\zeta \rho_\zeta}{\zeta^2} \cos(\theta) \ , \tag{5.10}$$

where C is a system constant, ρ_ζ is the differentiable reflectance, and θ is the incidence angle. In

a homogeneous medium with constant reflectance ρ and density σ, the integrated $P(\zeta_j \rightarrow \zeta_{j+1})$ evaluates to:

$$P(\zeta_j \rightarrow \zeta_{j+1}) = \int_{\zeta_j}^{\zeta_{j+1}} C \frac{T^2_{\zeta_j \rightarrow \zeta} \sigma_\zeta \rho_\zeta}{\zeta^2} \cos(\theta_j)\, d\zeta \approx \alpha_{\zeta_j} \rho'_{\zeta_j}, \tag{5.11}$$

where we approximate $\zeta \in [\zeta_j, \zeta_{j+1}]$ by $\frac{\zeta_j + \zeta_{j+1}}{2}$, and

$$\alpha_{\zeta_j} = \frac{1}{2}\Big(1 - e^{-2\sigma_{\zeta_j}\delta_j}\Big)\ ,\ \rho'_{\zeta_j} = C\rho_{\zeta_j} \frac{4\cos(\theta_j)}{(\zeta_j + \zeta_{j+1})^2}. \tag{5.12}$$

Volume rendering.. The observed power at the active sensor can be evaluated by plugging Eq. (5.11) into Eq. (5.3):

$$P = \sum_{j=1}^{N} \int_{\zeta_j}^{\zeta_{j+1}} C \frac{T^2_\zeta \cdot \sigma_\zeta \rho_\zeta}{\zeta^2} \cos(\theta_j)\, d\zeta = \sum_{j=1}^{N} w_j \rho'_{\zeta_j}, \tag{5.13}$$

where the weights w_j are now evaluated as (*c.f.* Eq. (5.4)):

$$w_j = 2\alpha_{\zeta_j} \cdot \prod_{k=1}^{j-1} (1 - 2\alpha_{\zeta_k})\ . \tag{5.14}$$

5.4.3. Assembling the beam from multiple rays

Next, we apply the adapted volume rendering formulation to multiple rays within a single LiDAR beam.

First range estimation.. We adopt a two-stage approach to extract range values from the neural field,[2] as shown in Fig. 5.1 (a). To estimate the range for an ideal ray $\mathbf{r}$, we uniformly sample N^c points and query their density values, then compute the weights $\{w^c_j\}_{j=1}^{N^c}$ using Eq. (5.14). A coarse peak estimate ζ_p is obtained by finding the point with the highest weight along the ray: $p = \text{argmax}_j \{w^c_j\}_{j=1}^{N^c}$. Next, we uniformly sample N^f points from the local interval $\zeta_j \in [\zeta_p - \epsilon, \zeta_p + \epsilon]$. The weights w^f_j at these points are recomputed and normalized to then obtain the final, refined range estimate ζ_f as: $\zeta_f = \sum_{j=1}^{N^f} w^f_j \cdot \zeta_j$.

Second range estimation.. As discussed in § 5.3.2 a single LiDAR beam might have multiple returns if enough energy was reflected from surfaces further away than the first return. To capture this behavior in our scene representation, we employ *truncated* volume rendering to estimate the radiant power beyond the first return (see Fig. 5.1 (b)).

Specifically, for each beam, we first predict a two-return mask p_s, by classifying its features $\mathbf{f}_{\text{beam}} = (\bar{\mathbf{f}}_{\text{geo}}, \mathbf{f}_{\text{dir}}, \mathbf{f}_{\text{range}})$, where $\bar{\mathbf{f}}_{\text{geo}}$ is the volume-rendered geometric feature, and $\mathbf{f}_{\text{range}}$

[2]Note the similarity to the detector in the instrument that first finds the peak of the waveform, then corrects for pulse shape.

describes the standard deviation and maximum discrepancy of range estimates at the first return. Intuitively, $\bar{\mathbf{f}}_{\text{geo}}$ describes the local geometry (*e.g.* an edge), $\mathbf{f}_{\text{dir}}$ encodes the relation of the beam to the geometry, and $\mathbf{f}_{\text{range}}$ characterizes the beam's prior interaction with the scene.

For beams that have two returns, we then perform *truncated* volume rendering as follows. We first add a buffer ξ[3] to the estimated range ζ_1 of the first return. We then reset the transmittance $T_{\zeta_1+\xi}$ to 1 by zeroing out the densities up to $(\zeta_1 + \xi)$ and recalculate the weights to ensure that they adhere to Eq. (5.14). Finally, we repeat the range estimation described above to estimate the range of the second return ζ_2. Note that for beams with two returns, the estimated range ζ_1 denotes the minimum range of all rays within the beam diameter, i.e. we perform volume rendering on all rays of a beam and pick the closest one as the first return. This is different from the beams with a single return where we directly use the central ray to estimate ζ_1.

Reflectance estimation.. At every detected surface point we can also retrieve reflectance from the neural field, using the relation $\rho = \sum_{j=1}^{N^f} w_j^f \cdot \rho_j$.

Ray drop probability.. In real LiDAR sensors, some emitted beams return no range measurement at all. This happens when the observed return signal has either too low amplitude or no clear peak (Fig. 5.1 (c)). However, this effect is hard to model in a fully physics-based way,[4] because it depends on (usually undisclosed) details of the detection logic. We empirically observe that the ray drop probability can be learned from LiDAR measurements. To this end, we augment the neural scene representation with a dedicated variable for the local probability of *not* back-scattering radiant power $p_d(\zeta) \in \{0, 1\}$[5]. Volume rendering integrates that quantity into a ray drop probability: $p_d(\mathbf{r}) = \sum_{j=1}^{N_c} w_j^c \cdot p_d(\zeta_j)$.

5.4.4. Training the neural LiDAR field

Given a set of posed LiDAR scans, we optimise our neural field model by minimising the loss

$$\mathcal{L} = \mathcal{L}_{\text{range}} + \lambda_e \mathcal{L}_e + \lambda_d \mathcal{L}_d + \lambda_s \mathcal{L}_s \ , \tag{5.15}$$

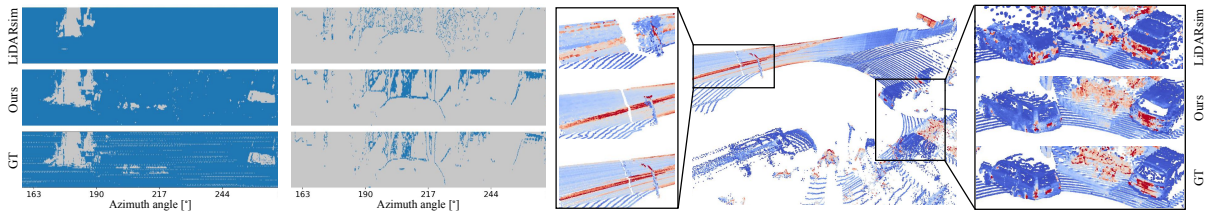
consisting of a reconstruction loss $\mathcal{L}_{\text{range}}$ for range estimation, reflectance loss $\mathcal{L}_e$, and classification losses $\mathcal{L}_d$ for ray drops and $\mathcal{L}_s$ for two returns.

Range reconstruction.. We add two separate losses for the coarse range ζ_p and the refined range ζ_f, $\mathcal{L}_{\text{range}} = \mathcal{L}^c_{\text{range}} + \mathcal{L}^f_{\text{range}}$. For coarse range, we impose a Gaussian distribution Rematas et al. (2021) around the ground truth $\hat{\zeta}$,

$$\mathcal{L}^c_{\text{range}} = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r} \in \mathcal{R}} \left(1 - \sum_{w_j \in \mathcal{X}_c^n} w_j \hat{w}_j + \sum_{w_k \in \mathcal{X}_c^e} w_k^2 \right) \ , \tag{5.16}$$

where $\mathcal{R}$ is the set of LiDAR rays, $\mathcal{X}_c^n$ and $\mathcal{X}_c^e$ denote points sampled within and outside the

[3]The buffer ξ is sensor specific and describes the minimum spacing between two distinct returns.
[4]Beyond simple thresholding, which our beam model would support.
[5]Please refer to the supplementary for discussions on this design choice.



(a) First return mask (b) Second return mask (c) LiDAR scans coloured by intensity values

Figure 5.3.: Qualitative results of LiDAR novel view synthesis on *Waymo Interp.* dataset. On the left, we color-code rays with and without return. On the right side, LiDAR intensity values are color-coded as :0 0.25.

Method	First range			Second range					Intensity		Ray drop		
	Recall@50↑	MAE ↓	MedAE ↓	Seg. recall ↑	Seg. precision ↑	Recall@50 ↑	MAE ↓	MedAE ↓	MSE^{1st} ↓	MSE^{2nd} ↓	Recall ↑	Precision ↑	IoU ↑
LiDARsim Manivasagam et al. (2020)	74.1	105.4	18.5	3.5	11.5	1.0	2258.0	1898.2	0.013	0.018	32.5	**85.5**	30.5
Central ray	92.8	32.8	5.6	79.8	62.9	61.1	589.1	21.8	0.004	0.009	64.3	81.7	**57.1**
Ours Diverged beam	92.3	36.1	5.7	82.1	55.6	67.4	505.1	13.4	**0.004**	**0.008**	**65.1**	78.0	56.1
GT mask	**93.2**	**29.7**	**5.6**	**100.0**	**100.0**	**79.8**	**116.0**	**8.1**	0.004	0.011	65.1	78.0	56.1

Table 5.1.: Comprehensive ray measurement evaluation of LiDAR novel view synthesis on *Waymo Interp.* dataset.

interval $[\hat{\zeta} - \epsilon, \hat{\zeta} + \epsilon]$. The ground truth weight $\hat{w}_j$ is calculated by integrating the Gaussian distribution. The range refinement loss is defined as: $\mathcal{L}^f_{\text{range}} = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r}\in\mathcal{R}} |\hat{\zeta} - \zeta_f|$.

Reflectance reconstruction. is optimized by minimizing an L2 loss w.r.t. the ground truth intensity $\hat{e}$: $\mathcal{L}_e = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r}\in\mathcal{R}} (\hat{e} - e)^2$.

Ray drop and dual return masks. are trained as classification tasks, by minimizing the combination of a binary cross entropy loss $\mathcal{L}_{bce}$ and a Lovasz loss $\mathcal{L}_{ls}$ Berman et al. (2018):

$$\mathcal{L}_* = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r}\in\mathcal{R}} \left(\mathcal{L}_{bce}(p_*, \hat{p}_*) + \mathcal{L}_{ls}(p_*, \hat{p}_*)\right) \ . \tag{5.17}$$

5.5. Experiments

We start by describing our LiDAR simulator, datasets, evaluation metrics, and baselines in § 5.5.1. In § 5.5.2, we evaluate NFL directly on the LiDAR novel view synthesis task. Finally, in § 5.5.3 we evaluate the suitability of our synthesized LiDAR data for two low-level tasks, point cloud registration and semantic segmentation.

5.5.1. Datasets and Evaluation setting

LiDAR simulator – TownReal dataset.. To enable quantitative evaluation in a controlled environment, we build a LiDAR simulator that allows us to virtually scan synthetic 3D assets represented either as triangular meshes or surfels. Specifically, we follow the LiDAR model

Method	TownClean			TownReal			Waymo interp.			Waymo NVS		
	MAE ↓	MedAE ↓	CD ↓	MAE ↓	MedAE ↓	CD ↓	MAE ↓	MedAE ↓	CD ↓	MAE ↓	MedAE ↓	CD ↓
i-NGP Müller et al. (2022)	42.2	4.1	17.4	49.8	4.8	19.9	**26.4**	5.5	**11.6**	**30.4**	7.3	15.3
DS-NeRF Deng et al. (2021)	41.7	3.9	16.6	48.9	4.4	18.8	28.2	6.3	14.5	30.4	7.2	16.8
URF Rematas et al. (2021)	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 Manivasagam et al. (2020)	159.6	**0.8**	23.5	162.8	3.8	27.4	116.3	15.2	27.6	160.2	16.2	34.7
Ours	**32.0**	2.3	**9.0**	**39.2**	**3.0**	**11.5**	30.8	**5.1**	12.1	32.6	**5.5**	**13.2**

Table 5.2.: Results of LiDAR novel view synthesis for the first range.

Method	TownClean			Waymo Interp.		
	MAE ↓	MedAE ↓	CD ↓	MAE ↓	MedAE ↓	CD ↓
i-NGP Müller et al. (2022)	41.0 (-1.2)	4.1 (0.0)	17.6 (0.2)	25.3 (-1.1)	4.5 (-1.0)	10.5 (-1.1)
DS-NeRF Deng et al. (2021)	37.4 (-4.2)	3.0 (-0.9)	14.4 (-2.2)	27.4 (-0.8)	5.4 (-1.0)	13.6 (-0.9)
URF Rematas et al. (2021)	46.4 (3.0)	4.5 (0.3)	18.4 (1.6)	28.3 (0.1)	5.3 (-0.1)	13.1 (0.2)
Ours	32.0 (-2.1)	2.3 (-2.5)	9.0 (-3.9)	30.8 (-2.1)	5.1 (-2.0)	12.1 (-2.3)

Table 5.3.: Ablation study of volume rendering for active sensing.

described in § 5.3.2 and allow control over the angular resolution, beam divergence, and pulse shape of the LiDAR sensor.

We use this simulator in combination with a 3D asset of a town Kasiopy to synthesize the *Town* dataset. We generate four scenes by splitting the 3D asset into four non-overlapping areas. Training and test scans are created from different trajectories. We use two different configurations of the LiDAR sensor: *(i) TownClean*, in which LiDAR scans are simulated using an idealized, non-divergent ray; and *(ii) TownReal*, with a diverged beam profile approximated via 37 subrays. See the supplementary material for further details.

Waymo Open dataset.. For evaluation on real-world data we use Waymo open dataset Sun et al. (2020a) which was captured by a 64-beam LiDAR sensor at 10 Hz. Here, we select four static scenes (see sequence IDs in supplementary material) and extract a five-second clip from each, resulting in 50 scans per scene. We hold out every 5-th frame as a test view and use the remaining 40 scans for training (*Waymo Interp.*)

To evaluate the methods in a more challenging setting we propose a novel evaluation protocol based on a closed-loop simulation (*Waymo NVS*). The protocol involves training and testing on all scans of a scene by first optimizing on the input views to synthesize novel views from a changed trajectory (shift the sensor by [1.5, 1.5, 0.5] meters[6]). The novel view are then used to re-optimize the method, synthesize scans in the original view and compare to the original scans to gauge performance. This formulation allows us to control task difficulty and could also be applied to evaluate camera-based novel view synthesis methods.

Evaluation metrics.. To evaluate range accuracy, we report four metrics: mean and median absolute errors (*MAE* [cm], *MedAE* [cm]), two-way Chamfer distance (*CD* [cm]), and *recall@50*, which denotes the percentage of rays with range errors below 50 cm. We additionally measure the two return segmentation recall (*Seg. recall*) and precision (*Seg. precision*). Intensity is

[6]Please refer to the supplementary for ablations on different sensor shift configurations.

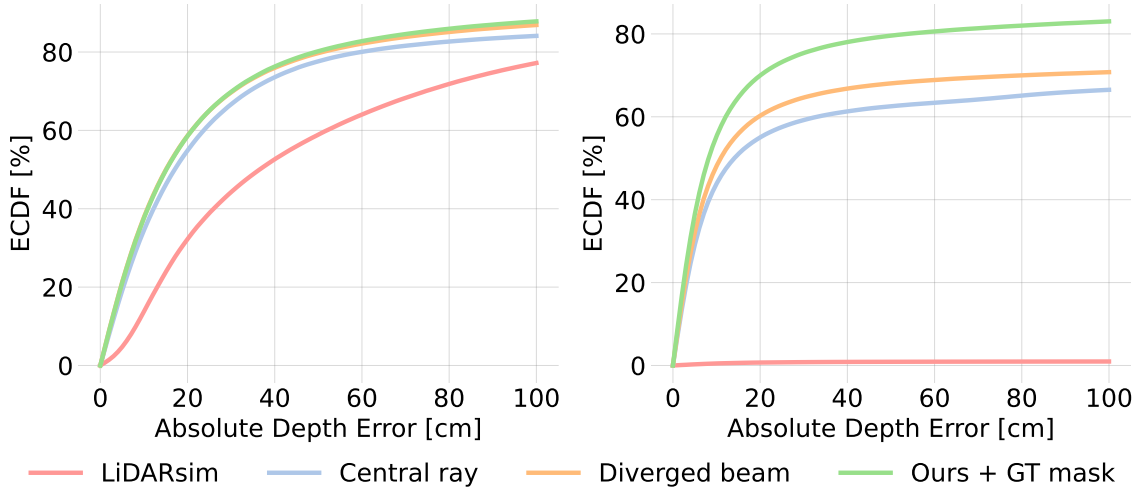


Figure 5.4.: Beam divergence modeling improves range accuracy of rays with dual returns. This is evident in the improved error distribution of the first (left) and second return range (right).

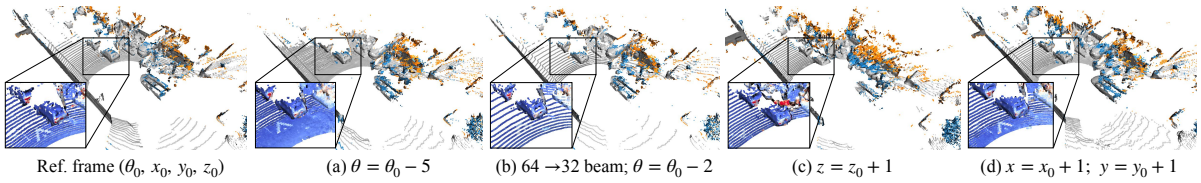


Figure 5.5.: LiDAR novel view synthesis by changing the sensor elevation angle $\theta[^\circ]$, pose $(x, y, z)[m]$ and number of beams. Zoom-in points are color-coded by intensity values.

evaluated using mean squared error (*MSE*). For ray drop segmentation, we report recall and precision [%], and intersection-over-union (*IoU* [%]). For point cloud registration, we report rotation error (*RE* [°]) and translation error (*TE* [cm]).

Baselines.. We compare NFL to four baselines. Closest to our problem setup is LiDARsim Manivasagam et al. (2020) which was designed for LiDAR synthesis based on surface reconstruction and ray-surfel casting. We re-implement LiDARsim and augment it with a diverged beam profile to enable synthesis of the second returns. Additionally, we adapt three NeRF-like methods that were originally proposed for image synthesis i-NGP Müller et al. (2022), DS-NeRF Deng et al. (2021), and URF Rematas et al. (2021) by modifying their volumetric rendering to improve their range predictions. Additional details are available in the supplementary.

5.5.2. Evaluation of LiDAR novel view synthesis

Ray measurement..

Using the *Waymo Interp.* dataset we conduct a comprehensive analysis of all the ray measurements and present the results in Tab. 5.1 and Fig. 5.3. Only NFL and LiDARsim Mani-

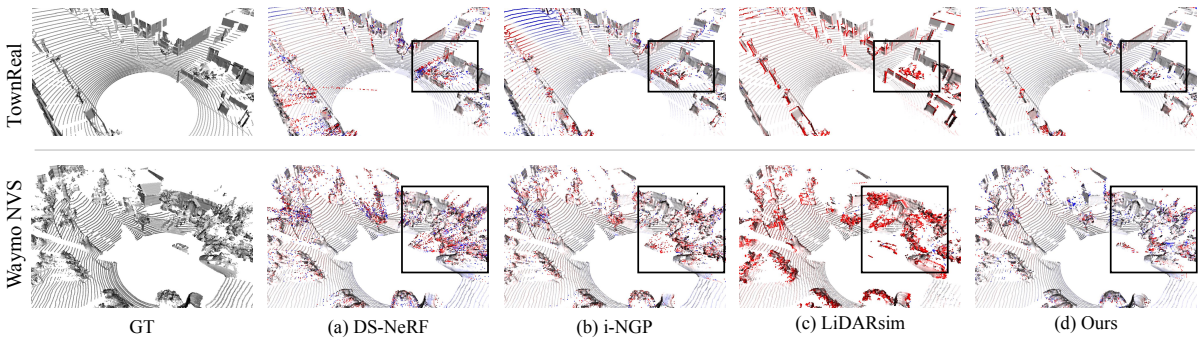


Figure 5.6.: Qualitative comparison of first range estimation. Regions with gross errors (-100 100 cm) are highlighted.

vasagam et al. (2020) are used for this experiment, as other baselines can only support a single return. LiDARsim's surface representation is explicit and not optimized for novel view synthesis nor accounts for view-dependent effects, which results in inferior range prediction and difficulties in retrieving secondary returns. In contrast, NFL directly optimizes the neural field for view synthesis while accounting for LiDAR acquisition process characteristics, resulting in significantly reduced range errors and superior performance in intensity and ray drop probability estimation. Notably, equipping our model with a *diverged beam* representation improves range estimation for both first and second returns for rays with dual returns(*c.f*. Fig. 6.3). However, diverged beam does slightly degrade the overall first-range accuracy likely due to imprecise two-return mask estimation. This hypothesis is supported by results using the ground truth two-return mask (*GT mask*). In Fig. 6.7 we show more qualitative results of novel view synthesis by NFL.

First range..

The results of estimating the range of the first return on all datasets are presented in Tab. 5.2 and Fig. 5.6. As demonstrated by the results on *TownClean*, *TownReal*, and *Waymo NVS*, the proposed volume rendering formulation of NFL effectively regularizes the density field resulting in superior performance in challenging cases. Even in the easier setting (resembling overfitting) on *Waymo Interp.* dataset, our method achieves competitive performance. In contrast, NeRF-like formulations (i-NGP Müller et al. (2022), DS-NeRF Deng et al. (2021), and URF Rematas et al. (2021)) perform poorly when evaluated on real novel views due to their inability to account for the active sensing principle. LiDARsim achieves promising results on datasets with simple geometry and clean LiDAR measurements, as evidenced by low *MedAE* scores on *TownClean* and *TownReal*. However, its explicit representation struggles with complex geometry in noisy real-world scenes, *e.g*., the vegetation regions in the *Waymo* dataset, resulting in high *MedAE* scores.

Ablation study of volume rendering for active sensing.. To evaluate the effectiveness of our volume rendering formulation for active sensors, we replace the volume rendering Mildenhall et al. (2020) formulation initially developed for passive sensing in all NeRF-based baselines and report performance difference in Tab. 5.3. Our formulation improves range accuracy across all settings, without any hyper-parameter tuning.

Method	TownClean			TownReal			Waymo NVS		
	Rec@5 ↑	RE ↓	TE ↓	Rec@5 ↑	RE ↓	TE ↓	Rec@2 ↑	RE ↓	TE ↓
i-NGP Müller et al. (2022)	70.3	0.1	4.2	76.0	0.1	4.2	60.2	0.1	1.9
DS-NeRF Deng et al. (2021)	58.3	0.2	5.1	56.2	0.2	5.1	42.3	0.1	2.4
URF Rematas et al. (2021)	61.5	0.2	5.0	59.9	0.1	4.7	32.1	0.1	2.7
LiDARsim Manivasagam et al. (2020)	**82.8**	0.1	**3.4**	79.2	0.1	**3.4**	62.8	0.1	1.8
Ours	80.2	0.1	3.7	**85.9**	0.1	**3.4**	**71.9**	0.1	**1.7**

Table 5.4.: Point cloud registration results on three datasets.

Method	Vehicle			Background		
	Recall ↑	Precision ↑	IoU ↑	Recall ↑	Precision ↑	IoU ↑
i-NGP Müller et al. (2022)	93.2	85.9	80.9	98.3	99.2	97.6
DS-NeRF Deng et al. (2021)	90.7	**87.1**	80.2	**98.5**	98.9	97.4
URF Rematas et al. (2021)	87.8	81.7	73.7	98.0	98.4	96.5
Lidarsim Manivasagam et al. (2020)	90.5	70.5	65.9	94.9	99.0	94.0
Ours	**95.9**	87.0	**83.9**	98.3	**99.5**	**97.8**

Table 5.5.: Semantic segmentation results on *Waymo NVS* dataset.

5.5.3. Downstream evaluation of novel views

Having demonstrated NFL's improved ability to synthesize high-quality LiDAR scans through various metrics, we proceed to evaluate their perceptual quality by using them as input for two low-level perception tasks: point cloud registration Huang et al. (2021b) and semantic segmentation Tang et al. (2020).

Point cloud registration.. To evaluate the extent to which synthesized scans preserve local geometric features, we apply the same point cloud registration model Huang et al. (2022b) pre-trained on Waymo Sun et al. (2020a) to both GT LiDAR scans and scans synthesized using different methods. Tab. 5.4 shows that NFL outperforms the baseline methods on datasets with complex geometry and higher noise levels (*TownReal* and *Waymo NVS*) that are more susceptible to artifacts occurring as a result of the LiDAR acquisition process.

Semantic segmentation.. To probe the potential domain gap between real and synthetic scans we apply the same, pre-trained semantic segmentation model Tang et al. (2020) to both and compare the predictions. Tab. 5.5 depicts the performance for both the *vehicle* and *background* classes. Notably, NFL achieves the highest recall for the *vehicle* class, which is strongly affected by dual returns and ray drops. Example predictions are shown in Fig. 5.7.

5.6. Limitations and future work

We have presented NFL, a neural field-based approach for synthesizing LiDAR scans from novel viewpoints. NFL combines the benefits of volume rendering with a physically based model of LiDAR acquisition process to faithfully model LiDAR characteristics including beam

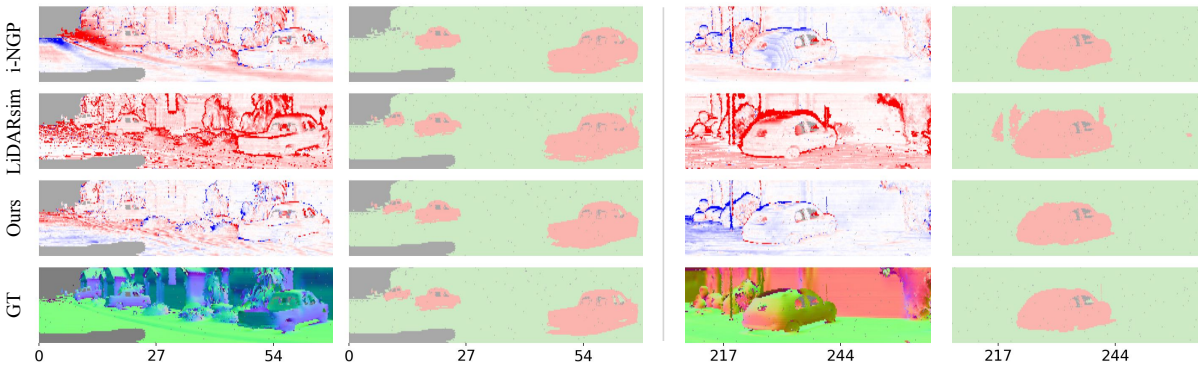


Figure 5.7.: Semantic segmentation results on synthesised *Waymo NVS* dataset. Geometry in-accuracy (-100 100 cm) leads to erroneous semantic segmentation (dropped rays, vehicle, pedestrian, background).

divergence, secondary returns, and ray dropping. Even though NFL significantly outperforms explicit reconstruct-then-simulate methods as well as other NeRF-style methods, it still has some limitations that we would like to address in future work. Firstly, with NFL, we try to seek a balance between adhering to the physical principles of LiDAR and incorporating semantic features and learning. While our formulation already shows improved performance over baselines, there is still potential for further improvements. For example, while real-world LiDAR sensors perform range detection on the integrated beam radiant, we actually found that using the density-based weights separately for each ray leads to improved performance. Prediction of the second return mask enables us to model secondary returns and to further improve the estimation of the first return. Yet, as indicated by the oracle study, improving the mask prediction could lead to further improvements. Finally, our method is based on a NeRF-style representation and therefore requires per-scene optimization. Generalization across scenes and handling dynamic environments are key challenges that we plan to address in future work.

Acknowledgements.. We sincerely thank Benjamin Naujoks, Steven Butrimas, Tomislav Medić, Yu Han, and Prof. Dr. Andreas Wieser for helpful discussions around LiDAR models. We are grateful for the feedback on figures from Rodrigo Caye Daudt. This appreciation extends to Zvi Greenstein for organisation support.

5.7. Appendix

In this supplementary document, we first present additional information about our dataset, evaluation setting, implementation details in § 5.7.1. We then elaborate on technical details of our methods in § 5.7.3. Additional results of the two return mask segmentation, more quantitative and qualitative results are provided in § 5.7.4.

5.7.1. Datasets and implementation details

Town dataset. To simulate *TownReal* dataset, we approximate a diverged beam profile using 37 subrays and the divergence angle $\gamma_0 = 2$ mrad Glennie (2012). We use the subray distribution proposed from Winiwarter et al. (2022) (*c.f*. Fig. 5.8). The dataset is shown in Fig. 5.10.

Waymo dataset. We use the following 4 scenes (*c.f*. Fig. 5.11) that are mostly static from *Waymo* Sun et al. (2020a) dataset

	Scene ID
Scene 1	10017090168044687777_6380_000_6400_000
Scene 2	10096619443888687526_2820_000_2840_000
Scene 3	10061305430875486848_1080_000_1100_000
Scene 4	10275144660749673822_5755_561_5775_561

***Waymo NVS* setting**. We simulate the new trajectory by shifting the sensor by [1.5, 1.5, 0.5] meters (see Fig. 5.11), yielding an overall displacement of $\approx$2.18 meters. This displacement magnitude corresponds to the requirements of various tasks, such as lane changes or adapting the sensor rig from a car to a truck. Moreover, our displacement from the trajectory is similar Yang et al. (2023b) or even larger Ost et al. (2022) than used in prior NVS works. Nevertheless, we run additional experiments by varying the displacements and report results in Tab. 5.6. NFL consistently outperforms baseline methods under different settings, and the improvement is more pronounced under large displacements.

Point cloud registration task. We utilize 49 paired consecutive frames per scene, with a relative displacement of $\approx$1 meter. *TE* is reported in centimeters and *RE* is reported in degrees.

5.7.2. Implementation details

Our method.. Our model is implemented based on *torch-ngp* Tang (2022); Müller et al. (2022) and can be trained on a single RTX 3090 GPU. During training we minimize 5.6 using the Adam Kingma and Ba (2014) optimiser, with an initial learning rate of 0.005 which linearly

	i-NGP	DS-NeRF	URF	LiDARsim	Ours
(0.5, 0.5, 0.5)	7.0 / 14.4	7.0 / 16.0	9.0 / 19.6	16.1 / 33.1	**5.4 / 13.0**
(1.5, 1.5, 1.0)	8.4 / 17.6	7.8 / 18.5	11.0 / 27.5	16.5 / 37.9	**5.8 / 14.3**
(2.5, 2.5, 1.5)	11.6 / 28.0	9.3 / 22.8	13.9 / 35.5	17.2 / 46.3	**6.4 / 18.4**

Table 5.6.: Varying the displacement on *Waymo NVS* dataset. Numbers are reported as *MedAE* / *CD* [cm].

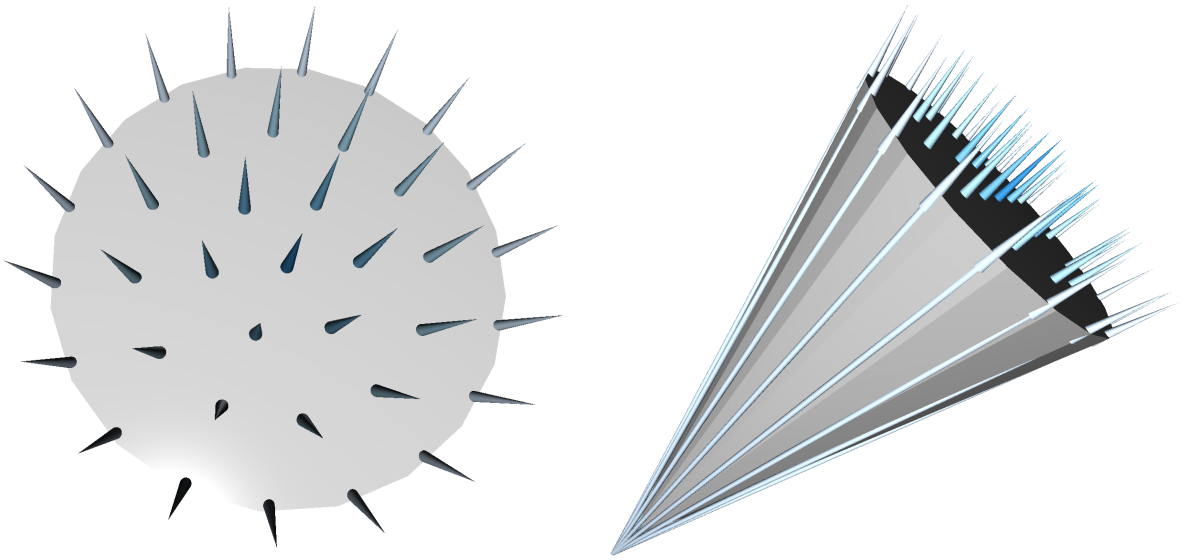


Figure 5.8.: Example diverged beam profile approximated via 37 diverged rays.

decays to 0.0005 towards the end of training. We clip the gradient magnitudes of all parameters to 1.0 to stabilize the optimisation. In the first stage, we sample $N^c = 768$ points and in the second stage $N^f = 64$ points for each ray. The window size ϵ for volume rendering is set to 0.8 m, and the buffer value ξ between two returns is set to 2 m. The weights in the loss function, i.e., λ_e, λ_d, and λ_s, are set to 50, 0.15, and 0.15, respectively.

LiDARsim.. Because the original implementation is not publicly available, we re-implemented LiDARsim Manivasagam et al. (2020) following the paper as close as possible. Specifically, for all points in the training set, we first estimate pointwise normal vectors using all points within a 20 cm radius ball. Then, we apply voxel down-sampling Tang et al. (2022) with a voxel size of 4 cm and reconstruct a disk surfel[7] for each point. Here, the input point represents the disk center and it orientation is defined by the estimated normal vector. At inference time, we perform ray-surfel intersection to determine the intersection points. We empirically observed that LiDARsim's Manivasagam et al. (2020) performance is sensitive to the selected surfel radius. Therefore, we have experimented with both a distance-dependent and fixed surfel radius and found that fixed surfel radius of 6 cm and 12 cm for *Waymo* and *Town* dataset, respectively lead to best range accuracy. To enable second range estimation, we augment LiDARsim with a di-

[7]We use the implementation from Point-Cloud-Utils Williams (2022) library.

verged beam profile approximated using 7 rays. To obtain the second return mask, we consider a LiDAR beam to have two returns if the maximum range difference between all subrays is larger than a threshold[8]. The first return is defined as the closest ray-surfel intersection, while the second return is the nearest one that is at least two meters away. To train the ray drop module, we utilize 40k samples from the Waymo dataset Sun et al. (2020a), and only apply this module after the ray-surfel intersection to refine the ray drop patterns. Please see Fig. 5.12 for more qualitative results.

Other NeRF methods.. We also use *torch-ngp* Tang (2022) codebase to implement other methods, using the same network and sampling configurations as used in ours. To estimate the range, we remove the radiance MLP and instead, apply volume rendering of the sampled ζ along the ray. For DS-NeRF Deng et al. (2021) and URF Rematas et al. (2021), we replace their positional encoding with a hash-grid Müller et al. (2022) to facilitate a fair comparison with i-NGP Müller et al. (2022). Moreover, we substitute the original L2 loss with the L1 loss, as it results in better performance. Finally, we follow the original paper and augment DS-NeRF Deng et al. (2021) and URF Rematas et al. (2021) with the ray distribution loss and line-of-sight loss, respectively, to regularise the underlying geometry.

5.7.3. Methodology and loss functions

First range estimation. If the maximum weight at the first stage w_p^c is below a predefined threshold $\eta = 0.1$, we assume that the network is uncertain about the reconstruction and the resulting range estimate may be inaccurate. In these cases, we only apply the coarse stage volume rendering and directly estimate the range as: $\zeta = \sum_{j=1}^{N^c} w_j^c \cdot \zeta_j$.

Range reconstruction loss. For coarse range, we impose a Gaussian distribution around the ground truth $\hat{\zeta}$ and we anneal the standard deviation δ during training, the annealing procedure is defined as:

$$\delta_k = \delta_{\max} \left(\frac{\delta_{\min}}{\delta_{\max}} \right)^{k/k_{\max}} \tag{5.18}$$

where k denotes the iteration number, $k_{\max}$ is the maximum iteration, and $\delta_{\max}$ and $\delta_{\min}$ correspond to empirically determined bounds for the standard deviation. The annealing parameters $\delta_{\min}$ and $\delta_{\max}$ are set to 0.25/0.3 and 1.2/1.6, respectively, for the *Town* and *Waymo* datasets. The maximum iteration $k_{\max}$ is set to 16000/24000 for the *Town* and *Waymo* datasets. The ground truth weight $\hat{w}_j$ is computed as:

$$\hat{w}_j = \int_{\zeta_j}^{\zeta_{j+1}} \frac{1}{\delta\sqrt{2\pi}} \exp\left(-\frac{(x-\hat{\zeta})^2}{2\delta^2} \right) \, dx. \tag{5.19}$$

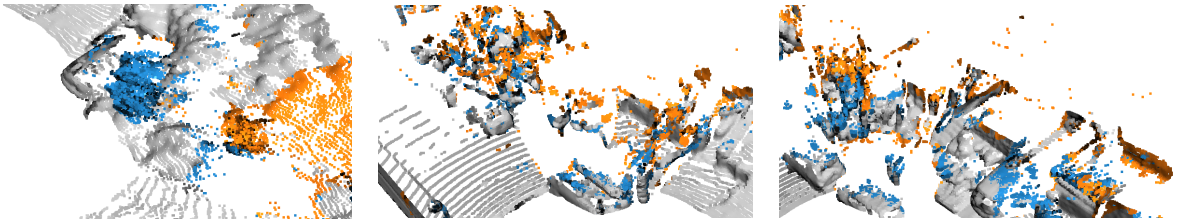


Figure 5.9.: Rendered secondary returns are color-coded in yellow.

Features			Two return segmentation			Second range		
$\bar{\mathbf{f}}_{\text{geo}}$	$\mathbf{f}_{\text{dir}}$	$\mathbf{f}_{\text{range}}$	Recall ↑	Precision ↑	IoU ↑	Recall@0.5 ↑	MAE ↓	MedAE ↓
✓			78.0	61.6	52.8	60.1	620.1	26.7
✓	✓		79.8	62.9	54.5	61.1	589.1	21.8
✓	✓	✓	82.1	55.6	49.8	67.4	505.1	13.4
threshold depth std.			30.8	24.2	14.8	24.7	1532.2	1461.4

Table 5.7.: Qualitative results of two return segmentation on *Waymo Interp*. dataset.

5.7.4. Additional results

Runtime analysis. Our *central ray* version takes 4.1 ms per frame to render the single returns on an RTX 3090 GPU, while other NeRF-style methods require 2.4 ms. Only around 10% of rays have second returns, resulting in low computational overhead. While our *diverged beam* incurs additional costs due to querying diverged rays, it can be disabled if needed, without compromising first return performance (*c.f*. Tab. 1). Our re-implementation of LiDARsim achieves 10 Hz runtime, but could be further improved using accelerated ray-tracing, *e.g*. OptiX. Note that all methods already match or even (greatly) exceed the normal LiDAR measurement frequency (≈10 Hz).

Ray drop modelling. There clearly is a link between ray drops and beam divergence. However, we found that modeling it through the beam feature yields worse performance, possibly because $\mathbf{f}_{\text{beam}}$ uses $\mathbf{f}_{\text{range}}$, which encodes the statistics of returns and is less meaningful for dropped rays. In future work, beam divergence could instead be incorporated through Intergrated Positional Encoding Barron et al. (2023) to model ray drops.

Two return mask prediction. We conduct an ablation study to investigate different design choices for predicting the two return mask and summarize the results in Tab. 5.7. We observe that concatenating the range feature $\mathbf{f}_{\text{range}}$ with the beam feature $\mathbf{f}_{\text{beam}}$ improves the segmentation recall and, consequently, the second range estimation. In addition to predicting the two return mask from the beam feature, we experiment with a simple heuristic-based baseline that thresholds the depth standard deviation of sub-rays. Specifically, we considered a LiDAR beam

[8] Sensor-specific parameter, 2 m on *Waymo* dataset.

Method	Vehicle			Background		
	Recall ↑	Precision ↑	IoU ↑	Recall ↑	Precision ↑	IoU ↑
i-NGP Müller et al. (2022) + L2	71.1	**97.0**	69.4	**99.6**	96.5	96.2
i-NGP Müller et al. (2022)	94.8	89.7	85.6	98.7	99.4	98.1
DS-NeRF Deng et al. (2021)	91.4	88.9	82.2	98.7	99.1	97.8
URF Rematas et al. (2021)	93.8	89.0	84.1	98.6	99.3	97.9
Lidarsim Manivasagam et al. (2020)	92.2	74.4	70.2	95.9	99.1	95.1
Ours	**95.7**	91.2	**87.6**	98.8	**99.5**	**98.3**

Table 5.8.: Semantic segmentation results on *Waymo Interp.* dataset.

Method	TownClean			TownReal			Waymo interp.			Waymo NVS		
	MAE ↓	MedAE ↓	CD ↓	MAE ↓	MedAE ↓	CD ↓	MAE ↓	MedAE ↓	CD ↓	MAE ↓	MedAE ↓	CD ↓
i-NGP Müller et al. (2022) + L2	63.6	14.8	37.1	78.2	18.4	44.5	41.4	14.7	24.9	47.3	17.6	29.5
i-NGP Müller et al. (2022)	42.2	4.1	17.4	49.8	4.8	19.9	**26.4**	5.5	**11.6**	**30.4**	7.3	15.3
DS-NeRF Deng et al. (2021)	41.7	3.9	16.6	48.9	4.4	18.8	28.2	6.3	14.5	30.4	7.2	16.8
URF Rematas et al. (2021)	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 Manivasagam et al. (2020)	159.6	**0.8**	23.5	162.8	3.8	27.4	116.3	15.2	27.6	160.2	16.2	34.7
Ours	**32.0**	2.3	**9.0**	**39.2**	**3.0**	**11.5**	30.8	**5.1**	12.1	32.6	**5.5**	**13.2**

Table 5.9.: Results of LiDAR novel view synthesis for the first range.

to have two returns if the standard deviation is above 30[9] cm. However, as shown in Table 5.7, this approach achieves limited success and performs much worse than the learned methods. More qualitative results are presented in Fig. 5.13.

Importance of the second return. Multiple returns are critical for vegetation analysis in remote sensing Lim et al. (2003). NFL is the first work to model the second return by combining beam divergence and *truncated* volume rendering. Unfortunately, second returns do not have semantic annotations in the Waymo dataset, which precluded a quantitative analysis. Nevertheless, qualitatively the rendered second returns are located mostly in vegetation regions, as shown in Fig. 5.9. This correlation suggests that secondary returns could indeed be useful for detecting vegetation.

Semantic segmentation on *Waymo Interp.* dataset.

We report additional semantic segmentation results on *Wamo Interp.* dataset in Tab. 5.8. NFL achieves the best performance for vehicle segmentation. Please note that *Waymo Interp.* is of significantly smaller size (10 test frames *vs*. 50 frames per scene in other datasets).

Quantitative results.

We perform further experiments to evaluate an additional baseline method, denoted as *i-NGPMüller et al. (2022) + L2*, which optimizes the range estimation through L2 loss Deng et al. (2021); Rematas et al. (2021). The comprehensive results of our experimentation are presented in Tab. 5.9, Tab. 5.10, Tab. 5.11, and Tab. 5.12. Our findings reveal that the L2 loss performs inferior to its L1 loss counterpart (*i.e*. i-NGP Müller et al. (2022)). However, replacing the standard volume rendering with the proposed formulation for active sensors, still

[9]Empirically determined as it leads to the best Intersection-of-Union score.

Method	TownClean			Waymo Interp.		
	MAE ↓	MedAE ↓	CD ↓	MAE ↓	MedAE ↓	CD ↓
i-NGP Müller et al. (2022) + L2	60.8 (-2.8)	12.6 (-2.2)	34.4 (-2.7)	40.8 (-0.6)	13.1 (-1.6)	24.0 (-0.8)
i-NGP Müller et al. (2022)	41.0 (-1.2)	4.1 (0.0)	17.6 (0.2)	25.3 (-1.1)	4.5 (-1.0)	10.5 (-1.1)
DS-NeRF Deng et al. (2021)	37.4 (-4.2)	3.0 (-0.9)	14.4 (-2.2)	27.4 (-0.8)	5.4 (-1.0)	13.6 (-0.9)
URF Rematas et al. (2021)	46.4 (3.0)	4.5 (0.3)	18.4 (1.6)	28.3 (0.1)	5.3 (-0.1)	13.1 (0.2)
Ours	32.0 (-2.1)	2.3 (-2.5)	9.0 (-3.9)	30.8 (-2.1)	5.1 (-2.0)	12.1 (-2.3)

Table 5.10.: Ablation study of volume rendering for active sensing.

Method	TownClean			TownReal			Waymo NVS		
	Rec@5 ↑	RE ↓	TE ↓	Rec@5 ↑	RE ↓	TE ↓	Rec@2 ↑	RE ↓	TE ↓
i-NGP Müller et al. (2022) + L2	40.6	0.2	6.2	39.6	0.2	6.7	26.5	0.1	3.2
i-NGP Müller et al. (2022)	70.3	0.1	4.2	76.0	0.1	4.2	60.2	0.1	1.9
DS-NeRF Deng et al. (2021)	58.3	0.2	5.1	56.2	0.2	5.1	42.3	0.1	2.4
URF Rematas et al. (2021)	61.5	0.2	5.0	59.9	0.1	4.7	32.1	0.1	2.7
LiDARsim Manivasagam et al. (2020)	**82.8**	**0.1**	**3.4**	79.2	**0.1**	**3.4**	62.8	**0.1**	1.8
Ours	80.2	0.1	3.7	**85.9**	0.1	**3.4**	**71.9**	**0.1**	**1.7**

Table 5.11.: Point cloud registration results on three datasets.

leads to improved performance, as demonstrated in Tab. 5.10.

Qualitative results. We show additional qualitative results in Fig. 5.14, Fig. 5.15, Fig. 5.16, and Fig. 5.17. We sample the middle frame of each dataset and present the first range errors in range-view projection.

Method	Vehicle			Background		
	Recall ↑	Precision ↑	IoU ↑	Recall ↑	Precision ↑	IoU ↑
i-NGP Müller et al. (2022) + L2	68.4	**90.2**	64.1	**99.3**	96.3	95.6
i-NGP Müller et al. (2022)	93.2	85.9	80.9	98.3	99.2	97.6
DS-NeRF Deng et al. (2021)	90.7	87.1	80.2	**98.5**	98.9	97.4
URF Rematas et al. (2021)	87.8	81.7	73.7	98.0	98.4	96.5
Lidarsim Manivasagam et al. (2020)	90.5	70.5	65.9	94.9	99.0	94.0
Ours	**95.9**	87.0	**83.9**	98.3	**99.5**	**97.8**

Table 5.12.: Semantic segmentation results on *Waymo NVS* dataset.

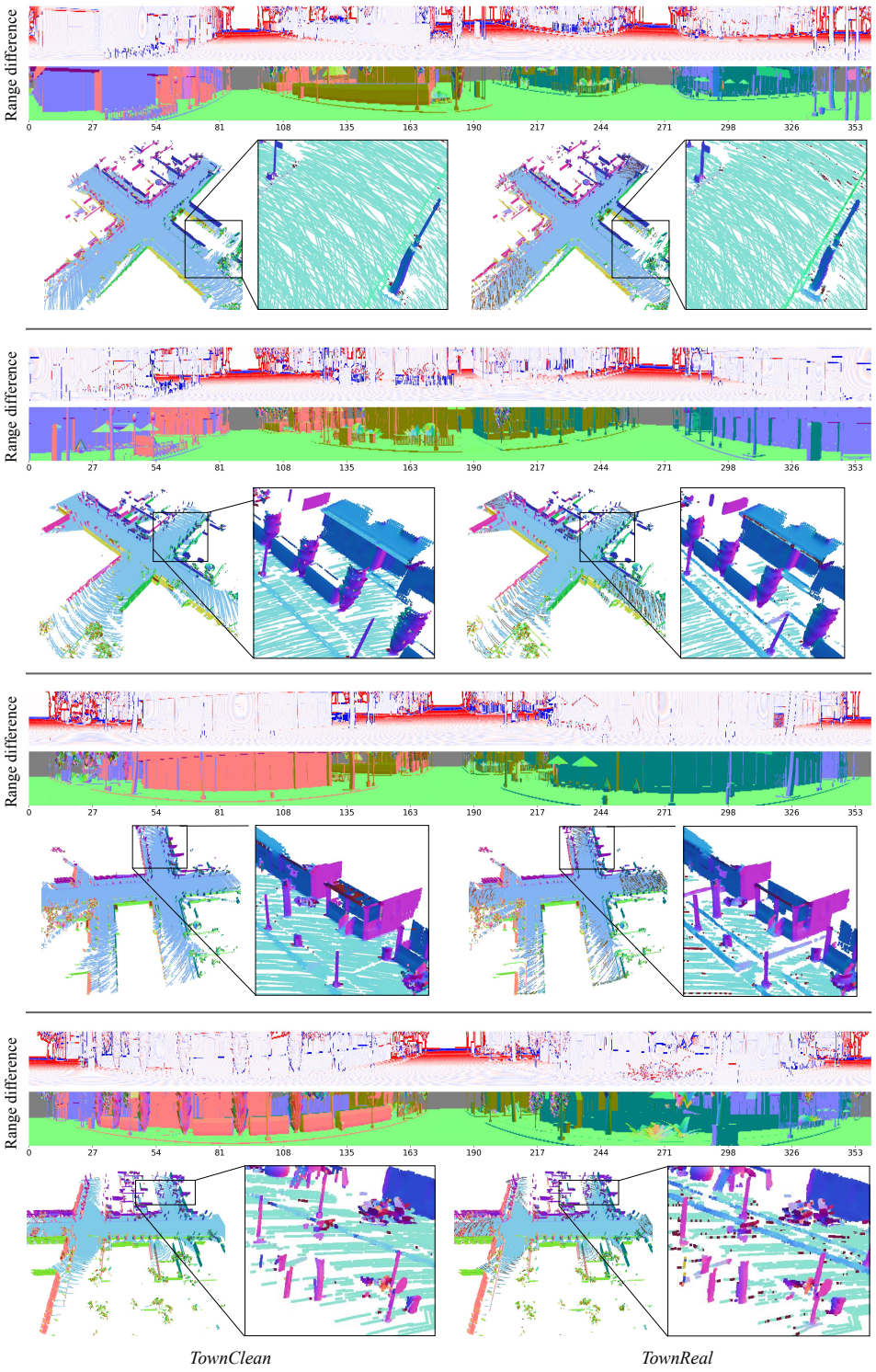
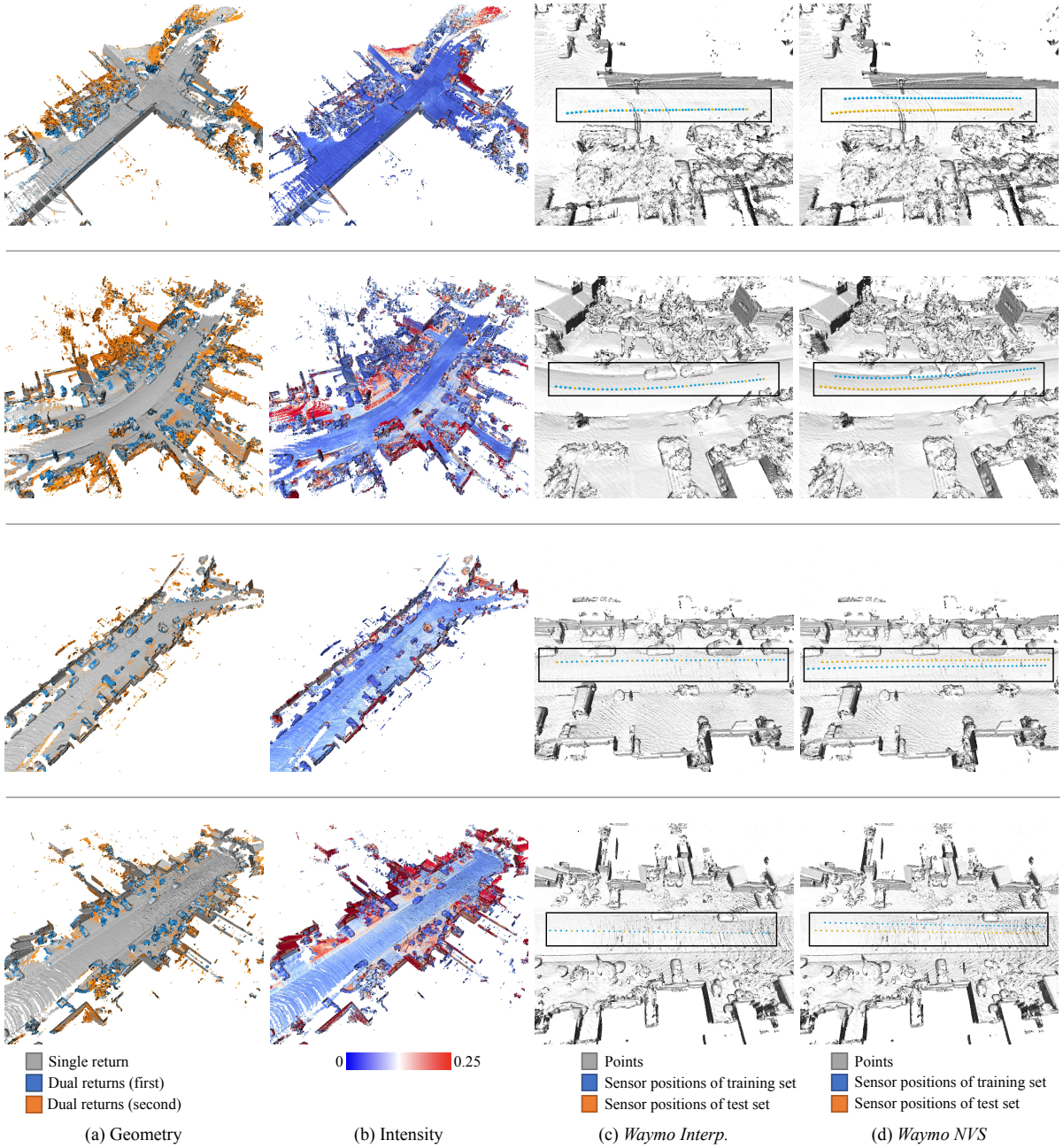


Figure 5.10.: Visualisation of *Town* dataset. Employing a diverged beam profile in range simulation results in an overestimation of range in the high range regime (-16 16 cm). Such range difference is also reflected on delicate structures, as evidenced by the point cloud view.



(a) Geometry (b) Intensity (c) *Waymo Interp.* (d) *Waymo NVS*

Figure 5.11.: Visualisations of *Waymo* dataset. We accumulate all 50 frames for each scene and show their geometry, intensity profile, and sensor positions of training and test sets on *Waymo Interp.* and *Waymo NVS* datasets.

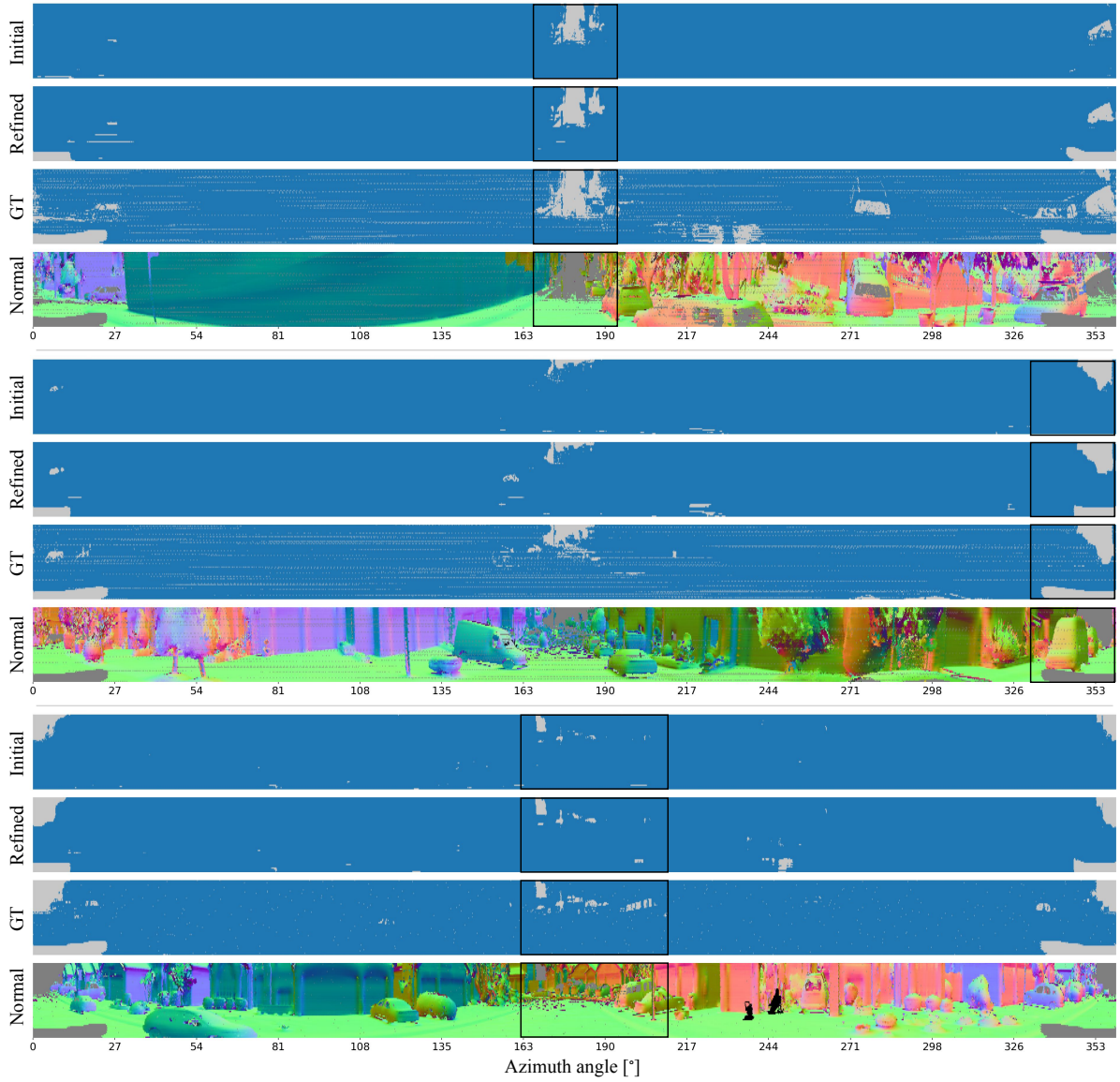


Figure 5.12.: Ray drop segmentation on *Waymo Interp.* dataset using LiDARsim Manivasagam et al. (2020). We show both the initial ray drop mask from ray-surfel query and the refined masks using learned ray-drop model.

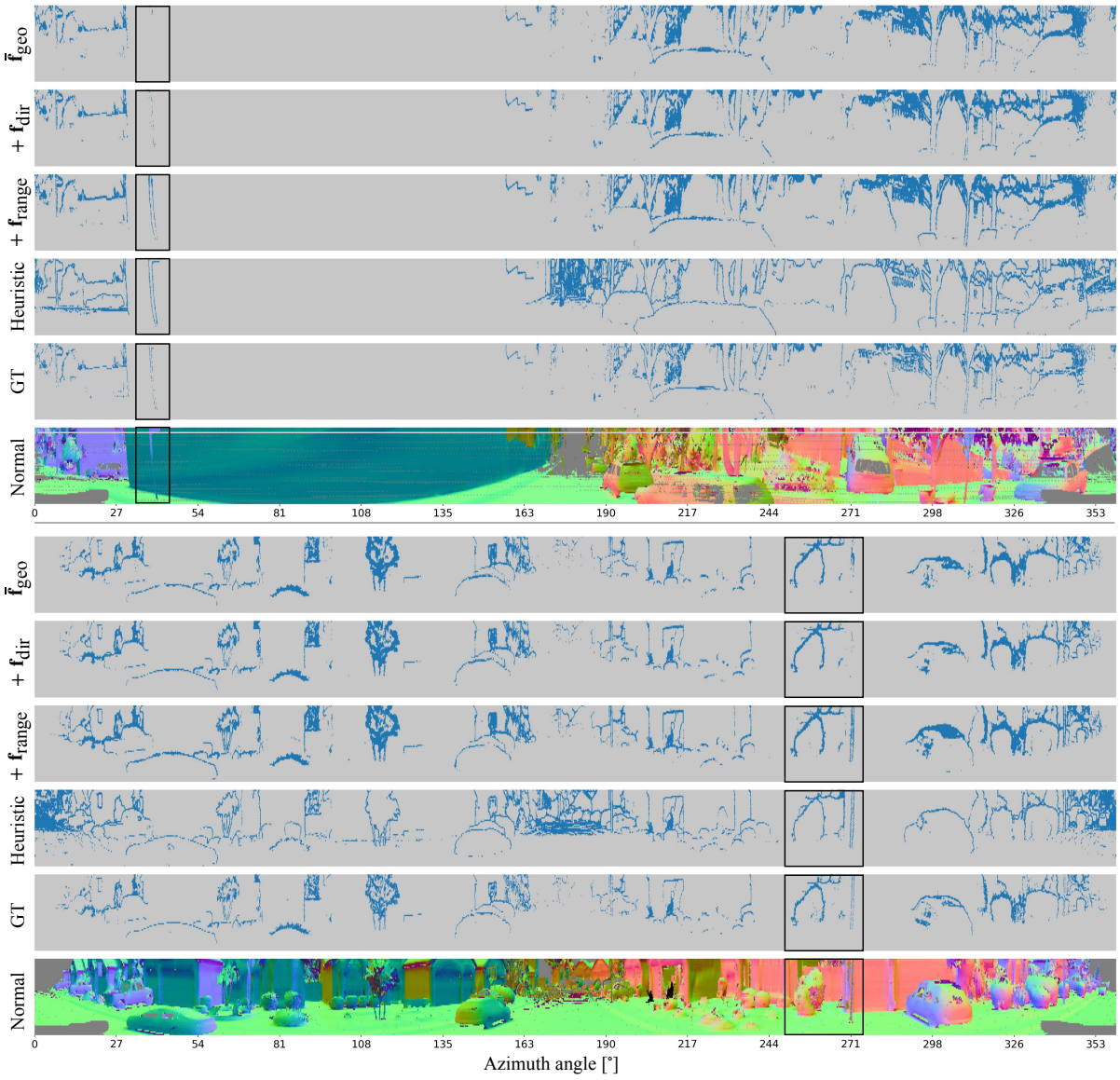


Figure 5.13.: Qualitative results of two return mask segmentation.

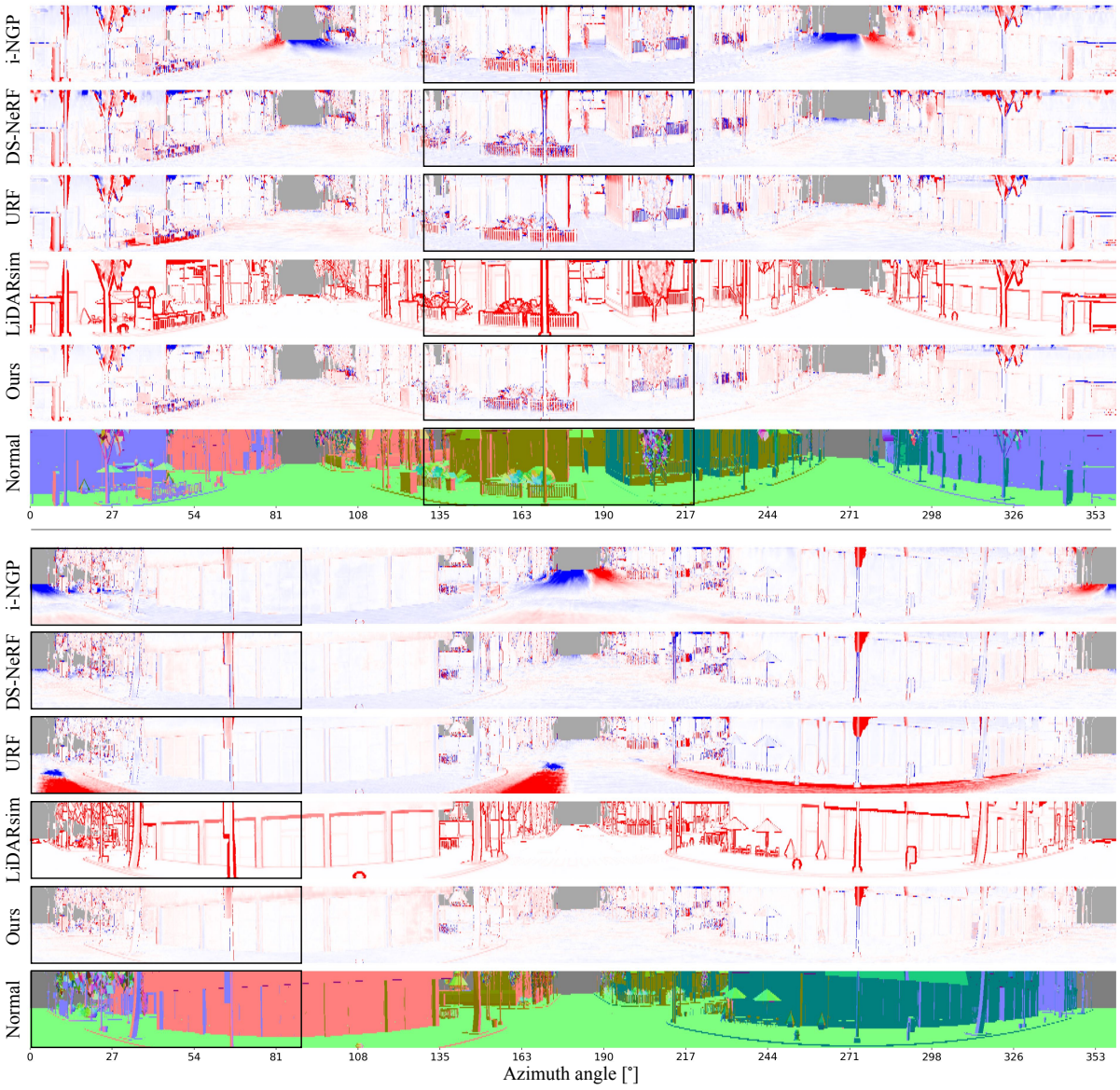


Figure 5.14.: Qualitative results of first range estimation on *TownClean* dataset.

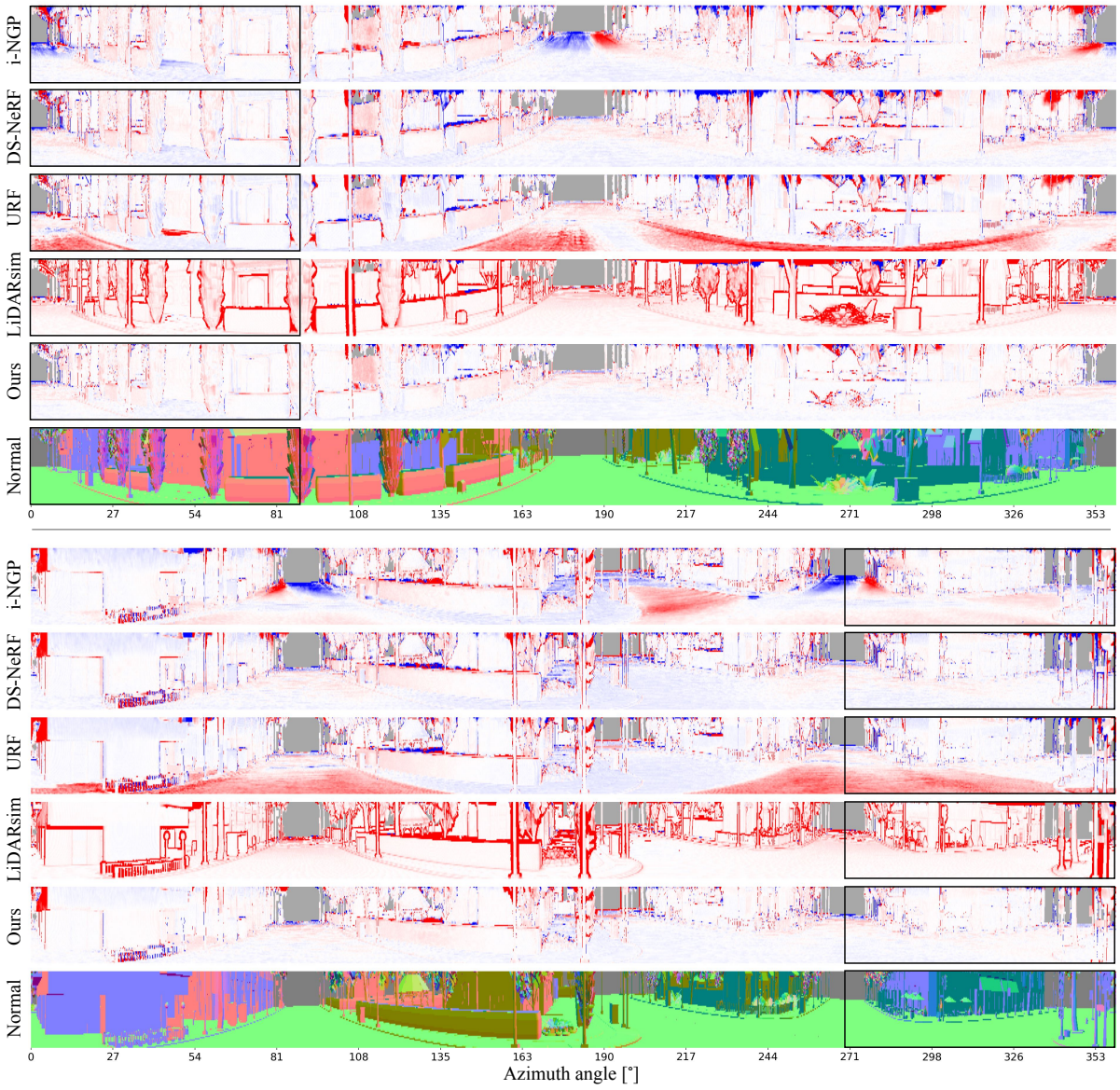


Figure 5.15.: Qualitative results of first range estimation on *TownReal* dataset.

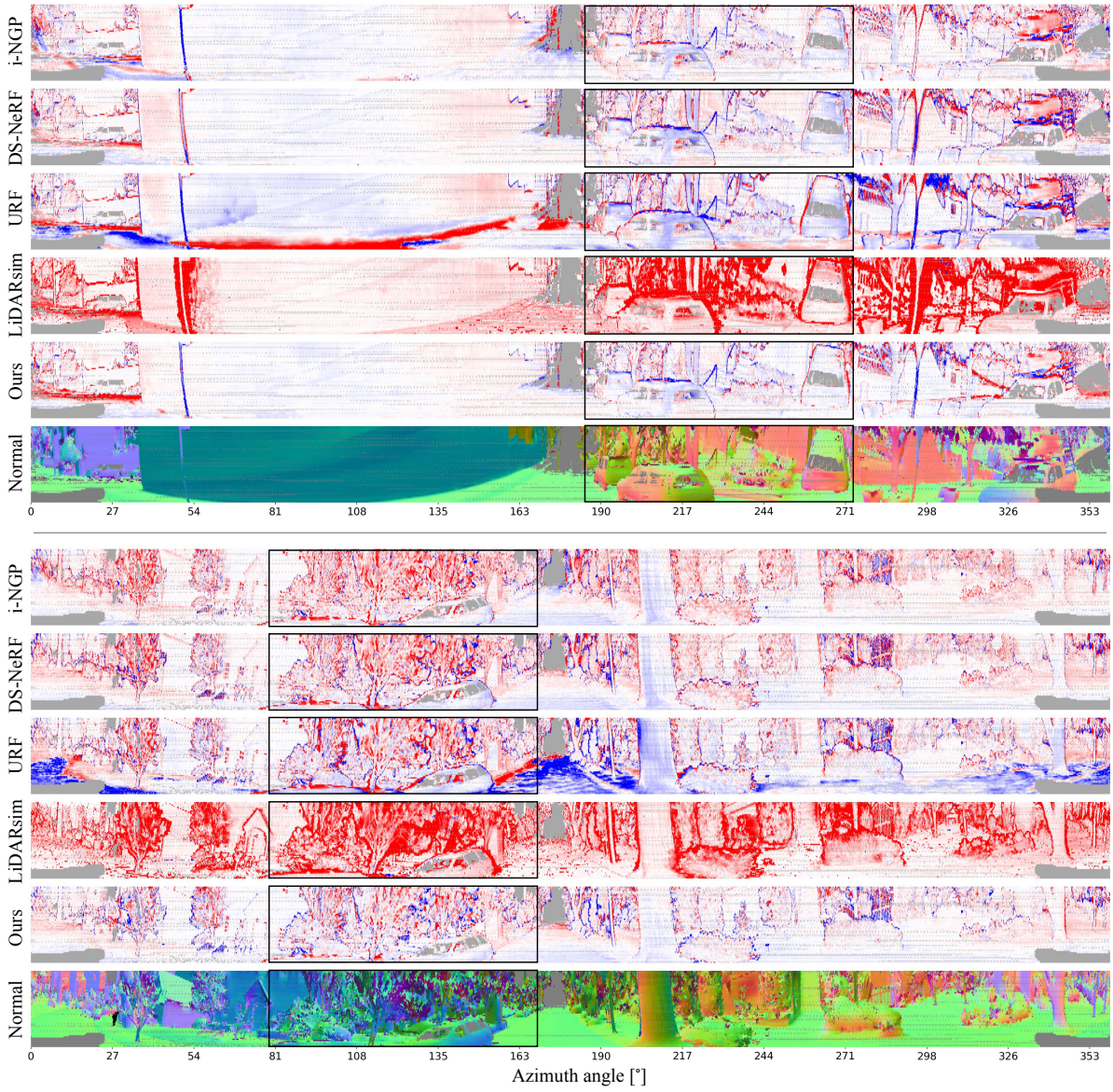


Figure 5.16.: Qualitative results of first range estimation on *Waymo NVS* dataset.

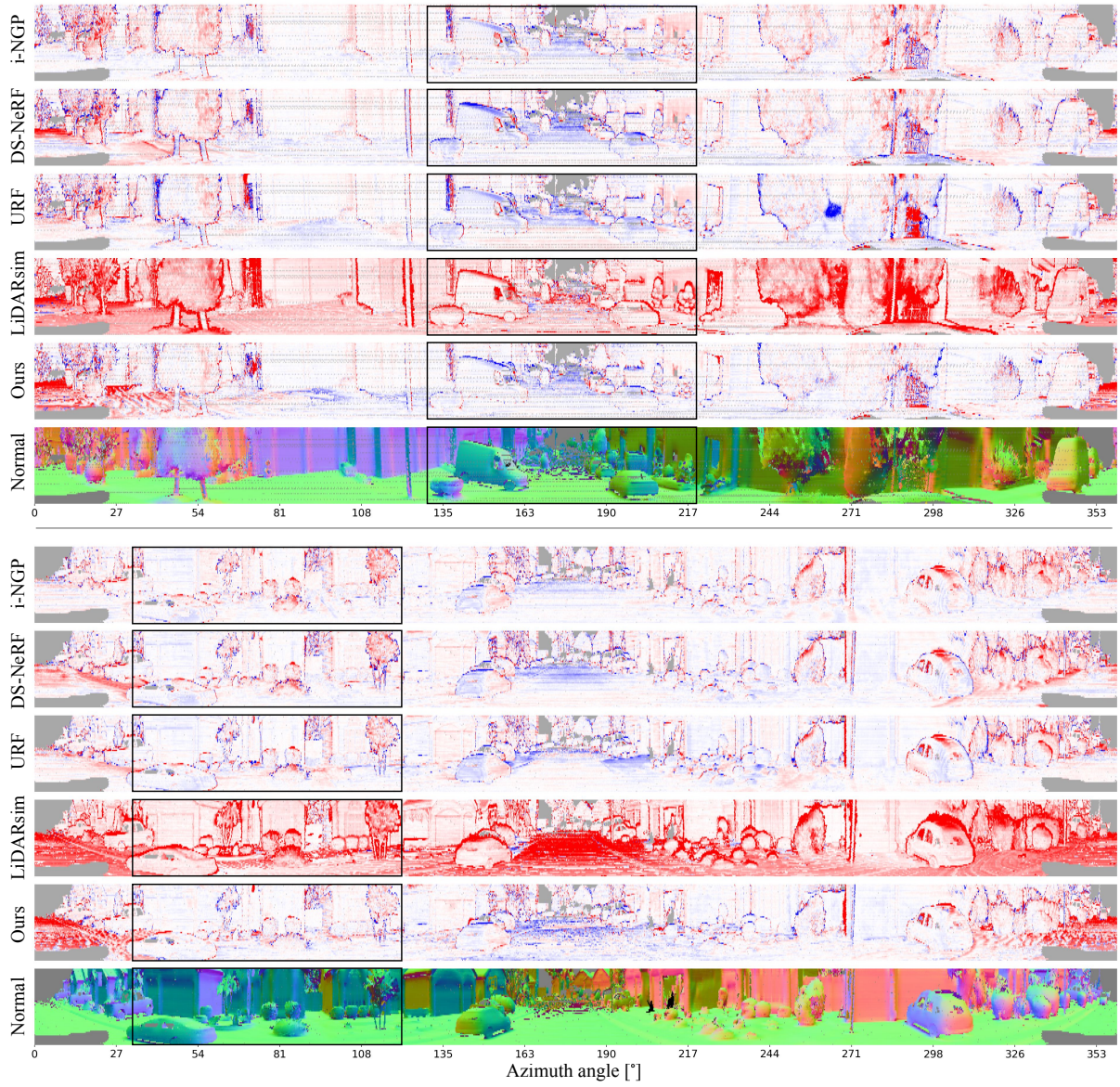


Figure 5.17.: Qualitative results of first range estimation on *Waymo Interp.* dataset.

6 Dynamic LiDAR Re-simulation using Compositional Neural Fields

Hanfeng Wu, Xingxing Zuo, Stefan Leutenegger, Or Litany, Konrad Schindler, Shengyu Huang **IEEE Computer Vision and Pattern Recognition, 2024**

(Author version; for typeset version please refer to the original conference paper.)

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.

6.1. 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

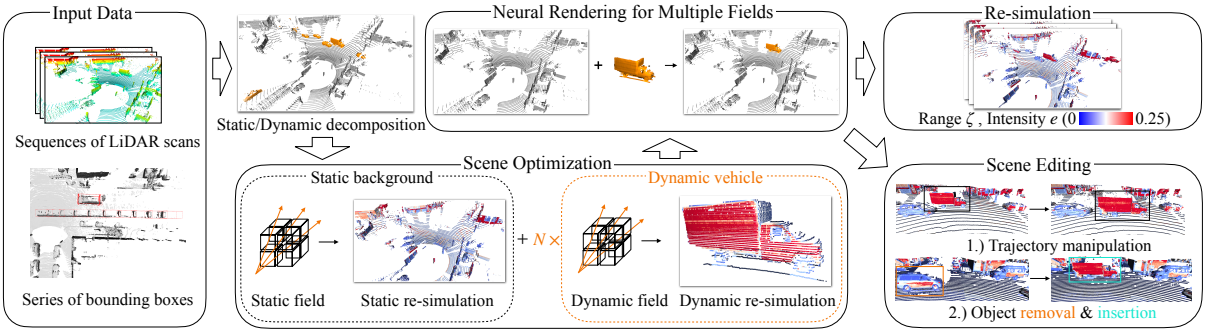


Figure 6.1.: Overview of DyNFL. Our method takes LiDAR scans and tracked bounding boxes of dynamic vehicles as input. DyNFL first decomposes the scene into a static background and N dynamic vehicles, each modelled using a dedicated neural field. These neural fields are then composed to re-simulate LiDAR scans in dynamic scenes. Our composition technique supports various scene edits, including altering object trajectories, removing and adding reconstructed neural assets between scenes.

limitations. Conventional simulators such as CARLA Dosovitskiy et al. (2017) and NVIDIA DRIVE Sim are capable of modeling LiDAR sensors. However, their reliance on manually designed 3D simulation assets requires significant human effort. LiDARsim Manivasagam et al. (2020) 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 Guillard et al. (2022). Following NeRF's Mildenhall et al. (2020) success in camera view synthesis, some works have applied neural fields for LiDAR modeling Huang et al. (2023); Tao et al. (2023); Zhang et al. (2023). In particular, Neural LiDAR Fields (NFL)Huang et al. (2023) 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 Yang et al. (2023c) followed Neural Scene Graph Ost et al. (2021b) 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 Huang et al. (2023).

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 Huang et al. (2023). 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 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.

6.2. Related work

6.2.1. Neural radiance fields and volume rendering

Neural Radiance Fields (NeRF) Mildenhall et al. (2020) 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 Barron et al. (2021, 2022); Verbin et al. (2022); Chen et al. (2022); Fridovich-Keil et al. (2022) have subsequently advanced NeRF's rendering quality by addressing challenges such as reducing aliasing artifacts Barron et al. (2021), scaling to unbound large-scale scenarios Barron et al. (2022), and capturing specular reflections on glossy surfaces Verbin et al. (2022). Certain works Chen et al. (2022); Fridovich-Keil et al. (2022); Müller et al. (2022); Kerbl et al. (2023) have explored more effective representations of radiance fields. TensorsRF Chen et al. (2022) employs multiple compact low-rank tensor components, such as vectors and matrices, to represent the radiance field. Plenoxels Fridovich-Keil et al. (2022) accelerates NeRF training by replacing MLPs with explicit plenoptic elements stored in sparse voxels and factorizing appearance through spherical-harmonic functions. Müller et al. Müller et al. (2022) 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. Kerbl et al. (2023) 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.

6.2.2. Dynamic neural radiance fields

Neural fields Xie et al. (2022) can be extended to represent dynamic scenes. On top of the *canonical* scene representation, some methods Pumarola et al. (2021b); Park et al. (2021a,b); Yuan et al. (2021) additionally model the 4D deformation fields. Meanwhile, some other works learn a space-time correlated Sara Fridovich-Keil and Giacomo Meanti et al. (2023); Li et al. (2021c); Attal et al. (2023); Liu et al. (2023), or decomposed Turki et al. (2023); Wu et al. (2022b); Yang et al. (2023a) 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 Ost et al. (2021b) and Panoptic Neural Fields Kundu et al. (2022) 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. UniSimYang et al. (2023c) employs neural SDF representation to model dynamic scenes in driving scenarios, and render in a similar way to Neural Scene Graph Ost et al. (2021b).

6.2.3. 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 Wang et al. (2021); Yariv et al. (2021). To address this, implicit surface representations like Occupancy Niemeyer et al. (2020); Oechsle et al. (2021) and signed distance functions (SDF) Wang et al. (2021); Yariv et al. (2021); Yu et al. (2022); Sun et al. (2022); Wang et al. (2022); Zuo et al. (2023); Li et al. (2023b); Wang et al. (2023a) in grid maps are commonly integrated into neural volume rendering techniques.

NeuS Wang et al. (2021) introduces a neural SDF representation for surface reconstruction, proposing an unbiased weight function for the appearance composition process in volume rendering. Similarly, VolSDF Yariv et al. (2021) 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 Li et al. (2023b) improves surface reconstruction using the multi-resolution hash encoding (MHE) Müller et al. (2022) and SDF-based volume rendering Wang et al. (2021). While these methods might deliver satisfying dense surface reconstructions, their training is time-consuming, taking hours for a single scene. Voxurf Wu et al. (2022a) offers a faster surface reconstruction method through a two-stage training procedure, recovering the coarse shape first and refining details later. Wang et al. Wang et al. (2023a) 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 Chang et al. (2023); Wang et al. (2023b) render depth by accumulating volume density and minimizing depth discrepancies between LiDAR and render depth during training. Rematas et al. Rematas et al. (2022) enforces empty space between the actual surface and the ray origin.

6.2.4. LiDAR simulation

While simulators like CARLA Dosovitskiy et al. (2017) and AirSim Shah et al. (2018) 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 Caccia et al. (2019); Zyrianov et al. (2022) offer an alternative but often lack control and produce distorted geometries Li et al. (2023a). Learning-based approaches Li et al. (2023a); Fang et al. (2020b); Manivasagam et al. (2020) try to enhance realism by transferring real scan properties to simulations. For example, Guillard et al. (2022) uses a RINet trained on RGB and real LiDAR data to augment simulated scan qualities. LiDARsim Manivasagam et al. (2020) employs ray-surfel casting with explicit disk surfels for more accurate simulations. Huang et al. Huang et al. (2023) 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 Yang et al. (2023c) constructs neural scene representations from realistic LiDAR and camera data, using SDF-based volume rendering for sensor measurement generation at novel viewpoints.

6.3. 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[1] 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. 6.1.

6.3.1. Neural scene decomposition

We leverage the inductive bias that driving scenes can be decomposed into a static component and N rigidly-moving dynamic components Huang et al. (2022b); Gojcic et al. (2021). 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) Müller et al. (2022) 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:

$$(s, \mathbf{f}_{\text{geo}}) = f_s(\mathbf{f}_{\text{pos}}), \quad e = f_e(\mathbf{f}_{\text{ray}}), \quad p_d = f_{\text{drop}}(\mathbf{f}_{\text{ray}}). \tag{6.1}$$

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

[1] We assume that the ground truth object detection and tracking annotations are available.

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.

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 $\mathbf{B}_1^v = \mathbf{T}_t\mathbf{B}_t^v$[2]. 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 $\mathbf{F}_{\text{static}}$.

6.4. 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 Huang et al. (2023) in § 6.4.1. Subsequently, we explore the extension of this formulation to SDF-based neural scene representation in § 6.4.2. Finally, we provide a detailed discussion on rendering LiDAR measurements from individual neural fields in § 6.4.3 and the process of composing results from different neural fields in § 6.4.4.

6.4.1. 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 Hahner et al. (2021, 2022); Huang et al. (2023):

$$P(\zeta) = \int_0^{2\zeta/c} P_e(t)H(\zeta - \frac{ct}{2})\, dt\ . \tag{6.2}$$

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:

$$H_T(\zeta) = \frac{\rho}{\pi}\cos(\theta)\delta(\zeta - \bar{\zeta})\ , \quad H_s(\zeta) = T^2(\zeta)\frac{A_e}{\zeta^2}\ , \tag{6.3}$$

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. (6.2) is non-differentiable and is thus not suitable for solving the inverse

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

problem. NFL Huang et al. (2023) solves it by extending it into a probabilistic formulation given by:

$$P(\zeta) = C \cdot \frac{T^2(\zeta) \cdot \sigma_\zeta \rho_\zeta}{\zeta^2} \cos(\theta) \ . \tag{6.4}$$

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:

$$P = \sum_{j=1}^{N} \int_{\zeta_j}^{\zeta_{j+1}} C \frac{T^2(\zeta) \cdot \sigma_\zeta \rho_\zeta}{\zeta^2} \cos(\theta_j) \ d\zeta = \sum_{j=1}^{N} w_j \rho'_{\zeta_j}, \tag{6.5}$$

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 Huang et al. (2023) for more details.

6.4.2. 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 Wang et al. (2021). 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 Wang et al. (2021), we compute the opaque density $\tilde{\sigma}_{\zeta_i}$ as follows:

$$\tilde{\sigma}_{\zeta_i} = \max\left(\frac{-\frac{\mathrm{d}\Phi_s}{\mathrm{d}\zeta_i}(f(\zeta_i))}{\Phi_s(f(\zeta_i))}, 0 \right), \tag{6.6}$$

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. (6.5) with opaque density from Eq. (6.6) and re-evaluate the radiant power and weights as:

$$P = \sum_{j=1}^{N} \mathcal{T}_{\zeta_j}^2 \tilde{\alpha}_{\zeta_j} \rho'_{\zeta_j}, \quad \tilde{w}_j = 2\tilde{\alpha}_{\zeta_j} \cdot \prod_{i=1}^{j-1}(1 - 2\tilde{\alpha}_{\zeta_i}) \ . \tag{6.7}$$

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

$$\tilde{\alpha}_{\zeta_j} = \max\left(\frac{\Phi_s(f(\zeta_j))^2 - \Phi_s(f(\zeta_{j+1}))^2}{2\Phi_s(f(\zeta_j))^2}, 0 \right). \tag{6.8}$$

Please refer to the supplementary for more details.

6.4.3. Volume rendering for LiDAR measurements

Consider rendering the LiDAR measurements from a single neural field, we employ the hierarchical samplingWang et al. (2021) technique to sample a total of $N_s = N_u + N_i$ points along each 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. (6.8). 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 $y \in \{\zeta, e, p_d\}$.

6.4.4. Neural rendering for multiple fields

Our full neural scene representation comprises $N + 1$ neural fields as discussed in Sec. 6.3.1. 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 § 6.3.1, 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) Shoemake (1985). 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.

6.5. 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:

$$\mathcal{L} = w_\zeta \mathcal{L}_\zeta + w_s \mathcal{L}_s + w_{\text{eik}} \mathcal{L}_{\text{eik}} + w_e \mathcal{L}_e + w_{\text{drop}} \mathcal{L}_{\text{drop}}, \tag{6.9}$$

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 Gropp et al. (2020), 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 Li et al. (2023b) to compute $\nabla s(\mathbf{p})$ as:

$$\nabla s(\mathbf{p}) = \frac{s\left(\mathbf{p}+\boldsymbol{\epsilon}\right) - s\left(\mathbf{p}-\boldsymbol{\epsilon}\right)}{2\epsilon}\ , \tag{6.10}$$

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 Huang et al. (2023) to supervise the ray drop estimation with a combination of a binary cross entropy loss $\mathcal{L}_{bce}$ and a Lovasz loss $\mathcal{L}_{ls}$ Berman et al. (2018) as:

$$\mathcal{L}_{\text{drop}} = \frac{1}{|\mathcal{R}|}\sum_{\mathbf{r}\in\mathcal{R}}\left(\mathcal{L}_{bce}(p_{d,est}, p_{d,gt}) + \mathcal{L}_{ls}(p_{d,est}, p_{d,gt})\right)\ . \tag{6.11}$$

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.

6.6. Experiments

6.6.1. Datasets and evaluation protocol

Real-world Dynamic scenes.. We construct *Waymo Dynamic* dataset by selecting four representative scenes from Waymo Open dataset Sun et al. (2020a), 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.

Real-world static scenes.. We also evaluate our method on four static scenes as introduced in Huang et al. (2023). There are two settings, *Waymo Interp* applies the same evaluation

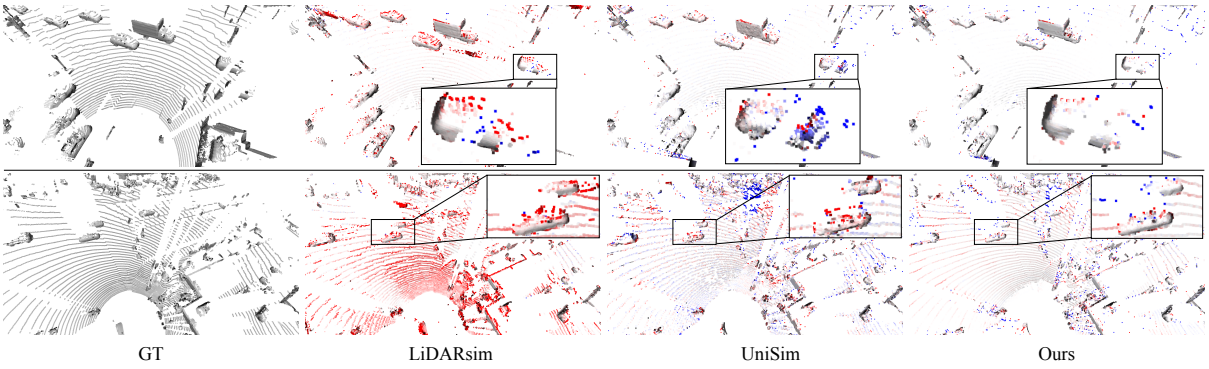


Figure 6.2.: Qualitative comparison of range estimation on *Waymo Dynamic* dataset. Dynamic vehicles are zoomed in, and points are color-coded by range errors (-100 100 cm).

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 Huang et al. (2023) for more details about this setting.

Synthetic static scenes.. *TownClean* and *TownReal* are synthetic static scenes introduced in NFL Huang et al. (2023). 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* Manivasagam et al. (2020), 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 Huang et al. (2023) and LiDARsimManivasagam et al. (2020) are two closest baselines to compare to. Additionally, we include i-NGP Müller et al. (2022), DS-NeRF Deng et al. (2022), and URF Rematas et al. (2022) for comparison. As for simulation on dynamic scenes, we compare to LiDARsim Manivasagam et al. (2020) and UniSim Yang et al. (2023c)[3]. Please refer to the supplementary for implementation details.

[3]We re-implement LiDARsim Lee et al. (2015) and UniSim Yang et al. (2023c) as they are not open-sourced.

Method	MAE ↓	MedAE ↓	CD ↓	MedAE Dyn ↓	Intensity RMSE ↓
LiDARsim Manivasagam et al. (2020)	170.1	11.5	31.1	16.0	0.10
Unisim Yang et al. (2023c)	35.6	6.1	14.3	14.3	**0.05**
Ours	**30.8**	**3.0**	**10.9**	**8.5**	**0.05**

Table 6.1.: 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 Müller et al. (2022)	42.2	4.1	17.4	49.8	4.8	19.9	**26.4**	5.5	**11.6**	30.4	7.3	15.3
DS-NeRF Deng et al. (2022)	41.7	3.9	16.6	48.9	4.4	18.8	28.2	6.3	14.5	30.4	7.2	16.8
URF Rematas et al. (2022)	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 Manivasagam et al. (2020)	159.6	0.8	23.5	162.8	3.8	27.4	116.3	15.2	27.6	160.2	16.2	34.7
NFLHuang et al. (2023)	32.0	2.3	9.0	39.2	3.0	11.5	30.8	5.1	12.1	32.6	5.5	13.2
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 6.2.: Evaluation of LiDAR NVS on static scenes.

6.6.2. LiDAR novel view synthesis evaluation

LiDAR NVS in dynamic scenes..

Quantitative comparisons with baseline methods are detailed in Tab. 6.1. DyNFL notably outperforms LiDARsim Manivasagam et al. (2020) and UniSim Yang et al. (2023c) in range reconstruction. This improvement is largely due to our SDF-based neural scene representation, which incorporates the physical aspects of LiDAR sensing. Additionally, 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. 6.2 and further supported by the data in Fig. 6.3.

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. 6.2 and Fig. 6.4. 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. 6.4), especially when compared to NFL Huang et al. (2023), 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.

6.6.3. 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. 6.3. 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.

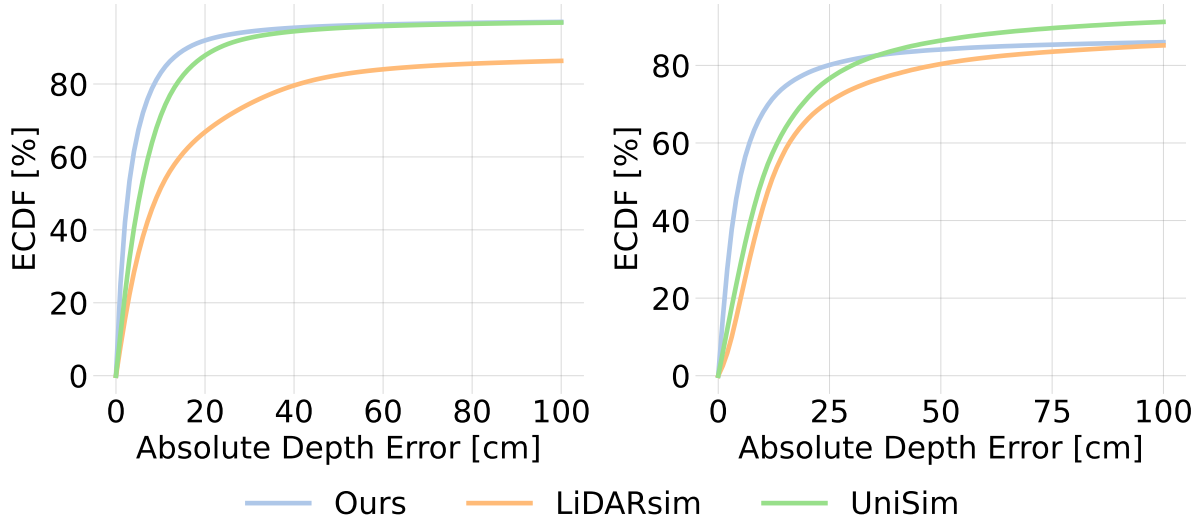


Figure 6.3.: ECDF plots showcasing range errors across all the points (left) and specifically for points associated with dynamic vehicles (right). Our neural fields composition demonstrates superior performance over LiDARsim Manivasagam et al. (2020) and UniSim Yang et al. (2023c), especially in the context of dynamic vehicles.

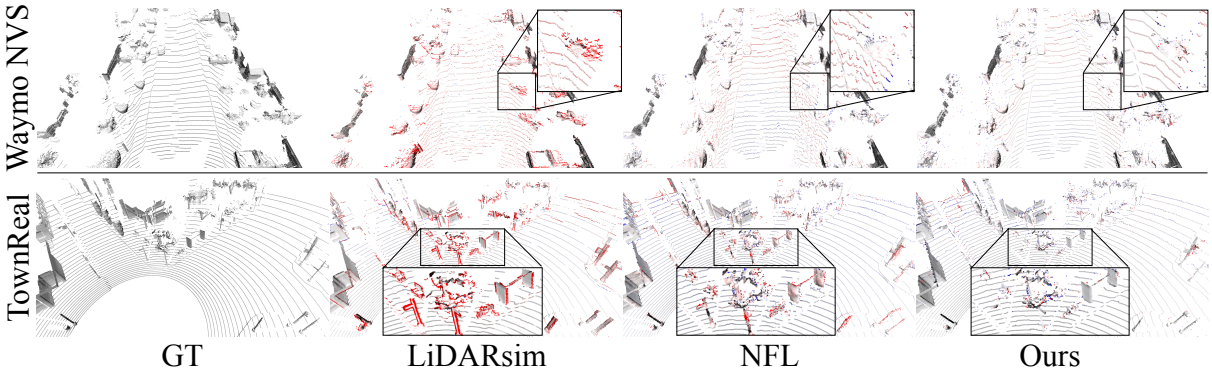


Figure 6.4.: Qualitative results of range estimation. Regions with gross errors (-100 100 cm) are highlighted.

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 Yang et al. (2023c). The results are shown in Tab. 6.1. UniSim Yang et al. (2023c) 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 (*c.f*. Fig. 6.5). This approach leads to a notable improvement in geometry reconstruction over UniSim Yang et al. (2023c), 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 (*c.f*. Fig. 6.3).

Surface points' SDF constraint.. We examine the importance of the surface points' SDF constraint discussed in Sec. 6.5 on *Town Real* and *Waymo Interp* datasets. The results shown in Tab. 6.4 suggest that our method yields improved geometry reconstruction quality by addition-

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 6.3.: 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 6.4.: Ablation study of the surface points' SDF regularisation.

ally enforcing LiDAR points to have zero SDF values.

6.6.4. 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 Fan et al. (2023) 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 Manivasagam et al. (2020), with the findings detailed in Tab. 6.5 and Fig. 6.6. 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 Tang et al. (2020), with the results presented in Tab. 6.6. DyNFL improves over baseline methods according to most evaluation metrics, underscoring the realism of our re-simulated LiDAR scans.

6.6.5. Scene editing

Beyond LiDAR novel view synthesis by adjusting the sensor configurations (*c.f*. Fig. 6.7), we additionally demonstrate the practicality of our compositional neural fields approach through two scene editing applications.

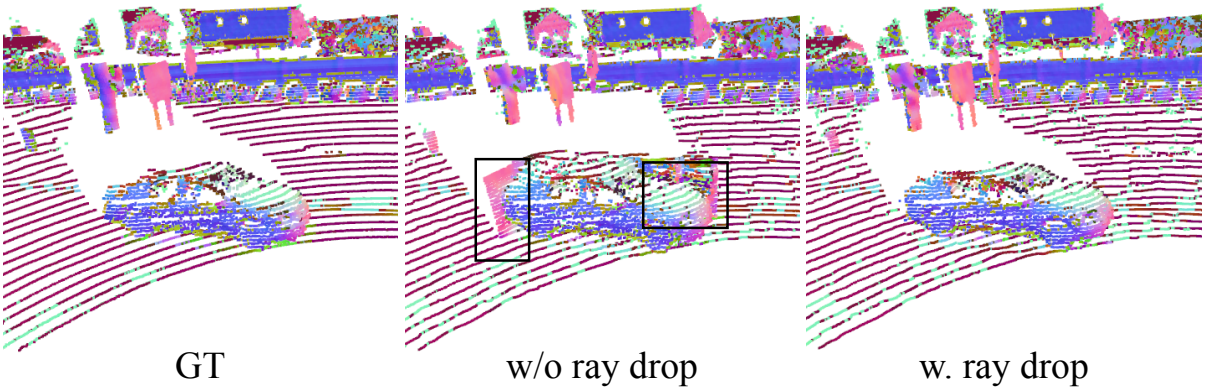


Figure 6.5.: Qualitative results on *Waymo Dynamic* dataset. Our model equipped with a ray drop module effectively composites multiple neural fields, re-simulating LiDAR scans of high quality.

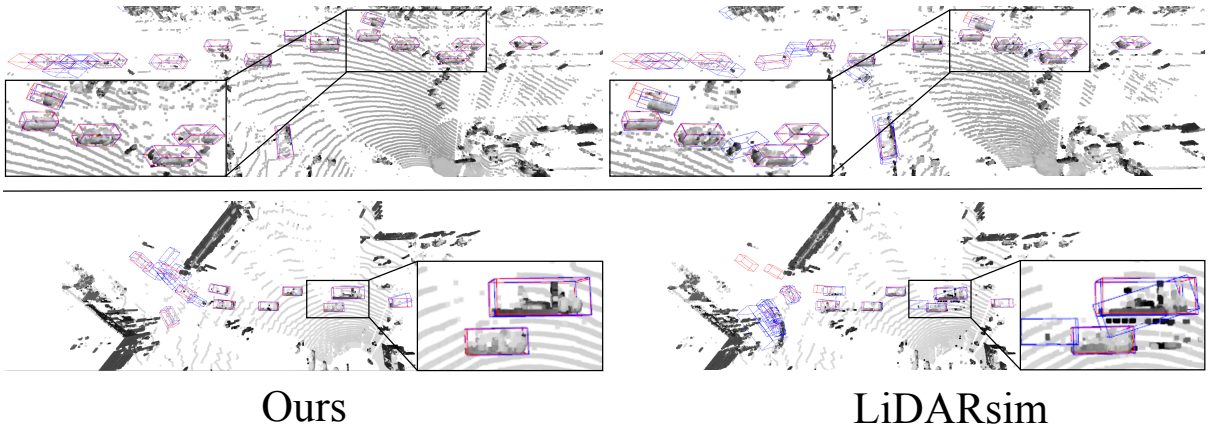


Figure 6.6.: Object detection results on *Waymo Dynamic* dataset. The ground truth and predicted bounding boxes are marked in red and blue, respectively.

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. 6.8, 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 Yang et al. (2023c) 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. 6.9. The high realism of our re-simulation is also indicated by the successful detection of inserted virtual objects.

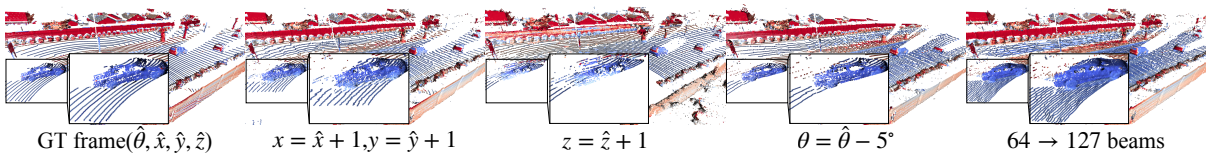


Figure 6.7.: LiDAR novel view synthesis by changing sensor elevation angle (θ), poses (x, y, z) and number of beams on *Waymo Dynamic* dataset. The points are color-coded by the intensity values (0 0.25).

	GT	Ours			LiDARSimManivasagam et al. (2020)		
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 6.5.: Object detection results on *Waymo Dyanmic* datasets.

6.7. 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.

Acknowledgements.. Or Litany is a Taub fellow and is supported by the Azrieli Foundation Early Career Faculty Fellowship.

Method	Vehicle			Background		
	Recall ↑	Precision ↑	IoU ↑	Recall ↑	Precision ↑	IoU ↑
i-NGP Müller et al. (2022)	91.8	83.6	78.1	97.9	99.2	97.1
DS-NeRF Deng et al. (2022)	89.3	84.8	77.3	98.1	98.8	97.0
URF Rematas et al. (2021)	86.9	79.8	72.0	97.7	98.5	96.2
Lidarsim Manivasagam et al. (2020)	89.6	68.9	64.0	94.5	98.9	93.5
NFL Huang et al. (2023)	**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 6.6.: Semantic segmentation results on *Waymo NVS* dataset.

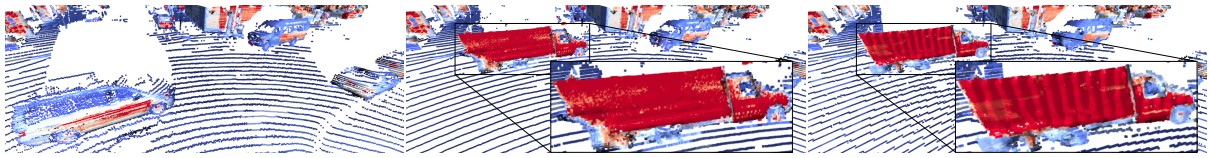


Figure 6.8.: Qualitative results of object removal and insertion. DyNFL seamlessly inserts the neural asset (truck) into a new scene attributed to our superior compositional rendering scheme. In contrast, UniSim Yang et al. (2023c) struggles to accurately model geometry.

Figure 6.9.: Qualitative results of object trajectory manipulation. The truck can be successfully detected after manipulation, indicating high-realism LiDAR re-simulation achieved by DyNFL.

7 Conclusions

placeholder

7.1. Lessons Learned

placeholder

7.2. Limitations

placeholder

7.3. Outlook

placeholder

A Bibliography

Aoki, Y., Goforth, H., Srivatsan, R.A., Lucey, S., 2019. PointnetLK: Robust & efficient point cloud registration using Pointnet, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Arun, K.S., Huang, T.S., Blostein, S.D., 1987. Least-squares fitting of two 3-d point sets. IEEE TPAMI 9, 698–700. doi:10.1109/TPAMI.1987.4767965.

Attal, B., Huang, J.B., Richardt, C., Zollhoefer, M., Kopf, J., O'Toole, M., Kim, C., 2023. HyperReel: High-fidelity 6-DoF video with ray-conditioned sampling, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Attal, B., Laidlaw, E., Gokaslan, A., Kim, C., Richardt, C., Tompkin, J., O'Toole, M., 2021. TöRF: Time-of-flight radiance fields for dynamic scene view synthesis. Advances in Neural Information Processing Systems (NeurIPS) .

Aygun, M., Osep, A., Weber, M., Maximov, M., Stachniss, C., Behley, J., Leal-Taixé, L., 2021. 4d panoptic LiDAR segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Bai, X., Luo, Z., Zhou, L., Fu, H., Quan, L., Tai, C.L., 2020. D3feat: Joint learning of dense detection and description of 3d local features, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P., 2021. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P., 2022. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P., 2023. Zip-nerf: Anti-aliased grid-based neural radiance fields, in: IEEE International Conference on Computer Vision (ICCV).

Baur, S.A., Emmerichs, D.J., Moosmann, F., Pinggera, P., Ommer, B., Geiger, A., 2021. SLIM: Self-supervised LiDAR scene flow and motion segmentation, in: IEEE International Conference on Computer Vision (ICCV).

Behl, A., Paschalidou, D., Donné, S., Geiger, A., 2019. PointFlowNet: Learning representations for rigid motion estimation from point clouds, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J., 2019. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences, in: IEEE International Conference on Computer Vision (ICCV).

Bengio, Y., Louradour, J., Collobert, R., Weston, J., 2009. Curriculum learning, in: IEEE International Conference on Machine Learning (ICML).

Berman, M., Triki, A.R., Blaschko, M.B., 2018. The Lovász-Softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Besl, P.J., McKay, N.D., 1992. Method for registration of 3-d shapes, in: Sensor fusion IV: control paradigms and data structures, International Society for Optics and Photonics. pp. 586–606.

Bijelic, M., Gruber, T., Mannan, F., Kraus, F., Ritter, W., Dietmayer, K., Heide, F., 2020. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Caccia, L., Van Hoof, H., Courville, A., Pineau, J., 2019. Deep generative modeling of LiDAR data, in: IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS).

Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O., 2020. nuScenes: A multimodal dataset for autonomous driving, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Carlsson, T., Steinvall, O., Letalick, D., 2001. Signature simulation and signal analysis for 3-D laser radar. Technical Report. Swedish Defence Research Agency.

Chang, M., Sharma, A., Kaess, M., Lucey, S., 2023. Neural radiance field with LiDAR maps, in: IEEE International Conference on Computer Vision (ICCV).

Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H., 2022. TensorRF: Tensorial radiance fields, in: European Conference on Computer Vision (ECCV).

Chen, C., Yang, B., 2016. Dynamic occlusion detection and inpainting of in situ captured terrestrial laser scanning point clouds sequence. ISPRS Journal of Photogrammetry and Remote Sensing 119, 90–107.

Chen, X., Li, S., Mersch, B., Wiesmann, L., Gall, J., Behley, J., Stachniss, C., 2021. Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data. IEEE RA-L .

Chng, S.F., Ramasinghe, S., Sherrah, J., Lucey, S., 2022. GARF: Gaussian activated radiance fields for high fidelity reconstruction and pose estimation. arXiv preprint arXiv:2204.05735 .

Choi, S., Zhou, Q.Y., Koltun, V., 2015. Robust reconstruction of indoor scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Choy, C., Dong, W., Koltun, V., 2020. Deep global registration, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Choy, C., Gwak, J., Savarese, S., 2019a. 4d spatio-temporal convnets: Minkowski convolutional neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Choy, C., Park, J., Koltun, V., 2019b. Fully convolutional geometric features, in: IEEE International Conference on Computer Vision (ICCV).

Curless, B., Levoy, M., 1996. A volumetric method for building complex models from range images, in: ACM SIGGRAPH.

Cuturi, M., 2013. Sinkhorn distances: Lightspeed computation of optimal transport. Advances in Neural Information Processing Systems (NeurIPS) .

Dendorfer, P., Osep, A., Milan, A., Schindler, K., Cremers, D., Reid, I., Roth, S., Leal-Taixé, L., 2021. MOTchallenge: A benchmark for single-camera multiple target tracking. IJCV .

Deng, H., Birdal, T., Ilic, S., 2018a. PPF-FoldNet: Unsupervised learning of rotation invariant 3d local descriptors, in: European Conference on Computer Vision (ECCV).

Deng, H., Birdal, T., Ilic, S., 2018b. Ppfnnet: Global context aware local features for robust 3d point matching, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Deng, K., Liu, A., Zhu, J.Y., Ramanan, D., 2021. Depth-supervised NeRF: Fewer views and faster training for free. arXiv preprint arXiv:2107.02791 .

Deng, K., Liu, A., Zhu, J.Y., Ramanan, D., 2022. Depth-supervised NeRF: Fewer views and faster training for free, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

DeTone, D., Malisiewicz, T., Rabinovich, A., 2018. Superpoint: Self-supervised interest point detection and description, in: CVPR Workshops.

Dewan, A., Caselitz, T., Tipaldi, G.D., Burgard, W., 2016. Rigid scene flow for 3d lidar scans, in: IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS).

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V., 2017. CARLA: An open urban driving simulator, in: CoRL, PMLR.

Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., Sattler, T., 2019. D2-Net: A trainable CNN for joint detection and description of local features, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise., in: Proc. KDD.

Fan, H., Yang, Y., Kankanhalli, M., 2021. Point 4d transformer networks for spatio-temporal modeling in point cloud videos, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Fan, H., Yu, X., Ding, Y., Yang, Y., Kankanhalli, M., 2020. PSTNet: Point spatio-temporal convolution on point cloud sequences, in: International Conference on Learning Representations (ICLR).

Fan, L., Wang, F., Wang, N., Zhang, Z., 2023. FSD V2: Improving fully sparse 3d object detection with virtual voxels. arXiv preprint arXiv:2308.03755 .

Fang, J., Zhou, D., Yan, F., Zhao, T., Zhang, F., Ma, Y., Wang, L., Yang, R., 2020a. Augmented LiDAR simulator for autonomous driving. IEEE RA-L 5, 1931–1938.

Fang, J., Zhou, D., Yan, F., Zhao, T., Zhang, F., Ma, Y., Wang, L., Yang, R., 2020b. Augmented lidar simulator for autonomous driving. IEEE Robotics and Automation Letters 5, 1931–1938.

Filatov, A., Rykov, A., Murashkin, V., 2020. Any motion detector: Learning class-agnostic scene dynamics from a sequence of lidar point clouds, in: IEEE International Conference on Robotics and Automation (ICRA).

Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A., 2022. Plenoxels: Radiance fields without neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Geiger, A., Lenz, P., Urtasun, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Giancola, S., Zarzar, J., Ghanem, B., 2019. Leveraging shape completion for 3d siamese tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E., 2017. Neural message passing for quantum chemistry, in: IEEE International Conference on Machine Learning (ICML).

Glennie, C., 2012. Calibration and kinematic analysis of the Velodyne HDL-64E S2 lidar sensor. Photogrammetric Engineering & Remote Sensing 78, 339–347.

Gojcic, Z., Litany, O., Wieser, A., Guibas, L.J., Birdal, T., 2021. Weakly supervised learning of rigid 3d scene flow, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Gojcic, Z., Zhou, C., Wegner, J.D., Guibas, L.J., Birdal, T., 2020a. Learning multiview 3d point cloud registration, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Gojcic, Z., Zhou, C., Wegner, J.D., Guibas, L.J., Birdal, T., 2020b. Learning multiview 3d point cloud registration, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Gojcic, Z., Zhou, C., Wegner, J.D., Wieser, A., 2019. The perfect match: 3d point cloud matching with smoothed densities, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Gojcic, Z., Zhou, C., Wieser, A., 2018. Learned compact local feature descriptor for TLS-based geodetic monitoring of natural outdoor scenes., in: ISPRS Annals.

Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y., 2020. Implicit geometric regularization for learning shapes, in: IEEE International Conference on Machine Learning (ICML).

Groß, J., Ošep, A., Leibe, B., 2019. AlignNet-3D: Fast point cloud registration of partially observed objects, in: IEEE International conference on 3D vision (3DV).

Gu, J., Ma, W.C., Manivasagam, S., Zeng, W., Wang, Z., Xiong, Y., Su, H., Urtasun, R., 2020. Weakly-supervised 3d shape completion in the wild, in: European Conference on Computer Vision (ECCV).

Guillard, B., Vemprala, S., Gupta, J.K., Miksik, O., Vineet, V., Fua, P., Kapoor, A., 2022. Learning to simulate realistic LiDARs, in: IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS).

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., Zhang, J., 2014. Performance evaluation of 3D local feature descriptors, in: ACCV.

Hahner, M., Sakaridis, C., Bijelic, M., Heide, F., Yu, F., Dai, D., Van Gool, L., 2022. LiDAR snowfall simulation for robust 3d object detection. arXiv preprint arXiv:2203.15118 .

Hahner, M., Sakaridis, C., Dai, D., Van Gool, L., 2021. Fog simulation on real LiDAR point clouds for 3d object detection in adverse weather, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Halber, M., Funkhouser, T.A., 2016. Structured global registration of RGB-D scans in indoor environments. arXiv preprint arXiv:1607.08539 .

Huang, J., Birdal, T., Gojcic, Z., Guibas, L.J., Hu, S.M., 2022a. Multiway non-rigid point cloud registration via learned functional map synchronization. IEEE T-PAMI .

Huang, J., Wang, H., Birdal, T., Sung, M., Arrigoni, F., Hu, S.M., Guibas, L.J., 2021a. Multi-BodySync: Multi-body segmentation and motion estimation via 3d scan synchronization, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Huang, S., Gojcic, Z., Huang, J., Wieser, A., Schindler, K., 2022b. Dynamic 3d scene analysis by point cloud accumulation, in: European Conference on Computer Vision (ECCV).

Huang, S., Gojcic, Z., Usvyatsov, M., Wieser, A., Schindler, K., 2021b. Predator: Registration of 3d point clouds with low overlap, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Huang, S., Gojcic, Z., Wang, Z., Williams, F., Kasten, Y., Fidler, S., Schindler, K., Litany, O., 2023. Neural lidar fields for novel view synthesis, in: IEEE International Conference on Computer Vision (ICCV).

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: IEEE International Conference on Machine Learning (ICML).

Johnson, A., Hebert, M., 1999. Using spin images for efficient object recognition in cluttered 3d scenes. IEEE TPAMI 21, 433–449.

Jund, P., Sweeney, C., Abdo, N., Chen, Z., Shlens, J., 2021. Scalable scene flow from point clouds in the real world. arXiv preprint arXiv:2103.01306 .

Kabsch, W., 1976. A solution for the best rotation to relate two sets of vectors. Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography 32, 922–923.

Kasiopy, . Town with suburb. https://www.turbosquid.com/3d-models/town-suburb-3d-max/1085661. Last accessed 2023.

Kazhdan, M., Bolitho, M., Hoppe, H., 2006. Poisson surface reconstruction, in: Eurographics.

Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G., 2023. 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics 42, 1–14.

Khoury, M., Zhou, Q.Y., Koltun, V., 2017. Learning compact geometric features, in: IEEE International Conference on Computer Vision (ICCV).

Kilic, V., Hegde, D., Sindagi, V., Cooper, A.B., Foster, M.A., Patel, V.M., 2021. Lidar light scattering augmentation (LISA): Physics-based simulation of adverse weather conditions for 3d object detection. arXiv preprint arXiv:2107.07004 .

Kim, G., Kim, A., 2020. Remove, then revert: Static point cloud map construction using multiresolution range images, in: IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS).

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization .

Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks .

Klenk, S., Koestler, L., Scaramuzza, D., Cremers, D., 2022. E-NeRF: Neural radiance fields from a moving event camera. arXiv preprint arXiv:2208.11300 .

Koenig, N., Howard, A., 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator, in: IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS).

Kundu, A., Genova, K., Yin, X., Fathi, A., Pantofaru, C., Guibas, L., Tagliasacchi, A., Dellaert, F., Funkhouser, T., 2022. Panoptic neural fields: A semantic object-aware neural scene representation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Kurup, A., Bos, J., 2021. DSOR: A scalable statistical filter for removing falling snow from LiDAR point clouds in severe winter weather. arXiv preprint arXiv:2109.07078 .

Lahoud, J., Ghanem, B., Pollefeys, M., Oswald, M.R., 2019. 3d instance segmentation via multi-task metric learning, in: IEEE International Conference on Computer Vision (ICCV).

Lai, K., Bo, L., Fox, D., 2014. Unsupervised feature learning for 3d scene labeling, in: IEEE International Conference on Robotics and Automation (ICRA).

Lee, S., Kang, D., Cho, S., Sim, S., Park, Y.W., Um, K., Cho, K., 2015. Lidar simulation method for low-cost repetitive validation, in: Advances in Computer Science and Ubiquitous Computing: CSA & CUTE, Springer. pp. 237–242.

Li, C., Ren, Y., Liu, B., 2023a. Pcgen: Point cloud generator for lidar simulation, in: IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 11676–11682.

Li, R., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A., 2019. PU-GAN: a point cloud upsampling adversarial network, in: IEEE International Conference on Computer Vision (ICCV).

Li, R., Lin, G., He, T., Liu, F., Shen, C., 2021a. HCRF-Flow: Scene flow from point clouds with continuous high-order crfs and position-aware flow embedding, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Li, X., Kaesemodel Pontes, J., Lucey, S., 2021b. Neural scene flow prior. Advances in Neural Information Processing Systems (NeurIPS) .

Li, Z., Müller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H., 2023b. Neuralangelo: High-fidelity neural surface reconstruction, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Li, Z., Niklaus, S., Snavely, N., Wang, O., 2021c. Neural scene flow fields for space-time view synthesis of dynamic scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Lim, K., Treitz, P., Wulder, M., St-Onge, B., Flood, M., 2003. Lidar remote sensing of forest structure. Progress in physical geography 27, 88–106.

Lin, C.H., Ma, W.C., Torralba, A., Lucey, S., 2021. BARF: Bundle-adjusting neural radiance fields, in: IEEE International Conference on Computer Vision (ICCV).

Liu, L., Gu, J., Zaw Lin, K., Chua, T.S., Theobalt, C., 2020. Neural sparse voxel fields. Advances in Neural Information Processing Systems (NeurIPS) .

Liu, X., Qi, C.R., Guibas, L.J., 2019a. FlowNet3D: Learning scene flow in 3d point clouds, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Liu, X., Yan, M., Bohg, J., 2019b. Meteornet: Deep learning on dynamic 3d point cloud sequences, in: IEEE International Conference on Computer Vision (ICCV).

Liu, Y.L., Gao, C., Meuleman, A., Tseng, H.Y., Saraf, A., Kim, C., Chuang, Y.Y., Kopf, J., Huang, J.B., 2023. Robust dynamic radiance fields, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Lu, F., Chen, G., Liu, Y., Qu, Z., Knoll, A., 2020. Rskdd-net: Random sample-based keypoint detector and descriptor. Advances in Neural Information Processing Systems (NeurIPS) .

Lucas, B.D., Kanade, T., 1981. An iterative image registration technique with an application to stereo vision, in: IJCAI.

Luo, A., Du, Y., Tarr, M.J., Tenenbaum, J.B., Torralba, A., Gan, C., 2022. Learning neural acoustic fields. arXiv preprint arXiv:2204.00628 .

Maas, A.L., Hannun, A.Y., Ng, A.Y., 2013. Rectifier nonlinearities improve neural network acoustic models, in: IEEE International Conference on Machine Learning (ICML).

Manivasagam, S., Wang, S., Wong, K., Zeng, W., Sazanovich, M., Tan, S., Yang, B., Ma, W.C.,

Urtasun, R., 2020. LiDARsim: Realistic LiDAR simulation by leveraging the real world, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Max, N., 1995. Optical models for direct volume rendering. IEEE Transactions on Visualization and Computer Graphics 1, 99–108.

Max, N., Chen, M., 2005. Local and global illumination in the volume rendering integral. Technical Report. Lawrence Livermore National Lab., Livermore, CA.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A., 2019. Occupancy networks: Learning 3d reconstruction in function space, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Mildenhall, B., Hedman, P., Martin-Brualla, R., Srinivasan, P.P., Barron, J.T., 2022. NeRF in the dark: High dynamic range view synthesis from noisy raw images, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R., 2020. NerF: Representing scenes as neural radiance fields for view synthesis, in: European Conference on Computer Vision (ECCV).

Müller, T., Evans, A., Schied, C., Keller, A., 2022. Instant neural graphics primitives with a multiresolution hash encoding. arXiv preprint arXiv:2201.05989 .

Müller, T., Evans, A., Schied, C., Keller, A., 2022. Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. 41, 102:1–102:15.

Nair, V., Hinton, G.E., 2010. Rectified linear units improve restricted boltzmann machines, in: IEEE International Conference on Machine Learning (ICML).

Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A., 2020. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Oechsle, M., Peng, S., Geiger, A., 2021. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Ost, J., Laradji, I., Newell, A., Bahat, Y., Heide, F., 2022. Neural point light fields, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F., 2021a. Neural scene graphs for dynamic scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F., 2021b. Neural scene graphs for dynamic scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Ouyang, B., Raviv, D., 2021. Occlusion guided self-supervised scene flow estimation on 3d point clouds. arXiv preprint arXiv:2104.04724 .

Pais, G.D., Ramalingam, S., Govindu, V.M., Nascimento, J.C., Chellappa, R., Miraldo, P., 2020. 3DRegNet: A deep neural network for 3d point registration, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S., 2019. DeepSDF: Learning continuous signed distance functions for shape representation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R., 2021a. Nerfies: Deformable neural radiance fields, in: IEEE International Conference on Computer Vision (ICCV).

Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M., 2021b. HyperNeRF: A higher-dimensional representation for topologically varying neural radiance fields. ACM Transactions on Graphics 40.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2021. Pytorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems (NeurIPS) .

Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A., 2020. Convolutional occupancy networks, in: European Conference on Computer Vision (ECCV).

Pfister, H., Zwicker, M., Van Baar, J., Gross, M., 2000. Surfels: Surface elements as rendering primitives, in: Computer Graphics and Interactive Techniques.

Piergiovanni, A., Casser, V., Ryoo, M.S., Angelova, A., 2021. 4d-net for learned multi-modal alignment. arXiv preprint arXiv:2109.01066 .

Pomerleau, F., Krüsi, P., Colas, F., Furgale, P., Siegwart, R., 2014. Long-term 3d map maintenance in dynamic environments, in: IEEE International Conference on Robotics and Automation (ICRA).

Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F., 2021a. D-NeRF: Neural radiance fields for dynamic scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F., 2021b. D-nerf: Neural radiance fields for dynamic scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Puy, G., Boulch, A., Marlet, R., 2020. FLOT: Scene flow on point clouds guided by optimal transport, in: European Conference on Computer Vision (ECCV).

Qadri, M., Kaess, M., Gkioulekas, I., 2022. Neural implicit surface reconstruction using imaging sonar. arXiv preprint arXiv:2209.08221 .

Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Qi, C.R., Zhou, Y., Najibi, M., Sun, P., Vo, K., Deng, B., Anguelov, D., 2021. Offboard 3d object detection from point cloud sequences, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Rasshofer, R.H., Spies, M., Spies, H., 2011. Influences of weather phenomena on automotive laser radar systems. Advances in Radio Science 9, 49–60.

Rematas, K., Liu, A., Srinivasan, P.P., Barron, J.T., Tagliasacchi, A., Funkhouser, T., Ferrari, V., 2021. Urban radiance fields. arXiv preprint arXiv:2111.14643 .

Rematas, K., Liu, A., Srinivasan, P.P., Barron, J.T., Tagliasacchi, A., Funkhouser, T., Ferrari, V., 2022. Urban radiance fields, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Rempe, D., Birdal, T., Zhao, Y., Gojcic, Z., Sridhar, S., Guibas, L.J., 2020. CaSPR: Learning canonical spatiotemporal point cloud representations.

Revaud, J., Weinzaepfel, P., De Souza, C., Pion, N., Csurka, G., Cabon, Y., Humenberger, M., 2019. R2D2: Repeatable and reliable detector and descriptor. arXiv preprint arXiv:1906.06195 .

Roessle, B., Barron, J.T., Mildenhall, B., Srinivasan, P.P., Nießner, M., 2021. Dense depth priors for neural radiance fields from sparse input views. arXiv preprint arXiv:2112.03288 .

Rudnev, V., Elgharib, M., Theobalt, C., Golyanik, V., 2022. EventNeRF: Neural radiance fields from a single colour event camera. arXiv preprint arXiv:2206.11896 .

Rusu, R.B., Blodow, N., Beetz, M., 2009. Fast point feature histograms (FPFH) for 3D registration, in: IEEE International Conference on Robotics and Automation (ICRA).

Rusu, R.B., Blodow, N., Marton, Z.C., Beetz, M., 2008. Aligning point cloud views using persistent feature histograms, in: IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS).

Sara Fridovich-Keil and Giacomo Meanti, Warburg, F.R., Recht, B., Kanazawa, A., 2023. K-planes: Explicit radiance fields in space, time, and appearance, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Sarlin, P.E., DeTone, D., Malisiewicz, T., Rabinovich, A., 2020. Superglue: Learning feature matching with graph neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Schauer, J., Nüchter, A., 2018. The peopleremover—removing dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid. IEEE RA-L 3, 1679–1686.

Schönberger, J.L., Frahm, J.M., 2016. Structure-from-motion revisited, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Shah, S., Dey, D., Lovett, C., Kapoor, A., 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles, in: International Conference on Field and Service Robotics, Springer. pp. 621–635.

Shih, Y.C., Liao, W.H., Lin, W.C., Wong, S.K., Wang, C.C., 2022. Reconstruction and synthesis of lidar point clouds of spray. IEEE RA-L 7, 3765–3772.

Shoemake, K., 1985. Animating rotation with quaternion curves, in: Conference on Computer Graphics and Interactive Techniques, Association for Computing Machinery. p. 245–254.

Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A., 2013. Scene coordinate regression forests for camera relocalization in RGB-D images, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Sinkhorn, R., 1964. A relationship between arbitrary positive matrices and doubly stochastic matrices. The annals of mathematical statistics 35, 876–879.

Sun, D., Yang, X., Liu, M.Y., Kautz, J., 2018. PWC-net: Cnns for optical flow using pyramid, warping, and cost volume, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Sun, J., Chen, X., Wang, Q., Li, Z., Averbuch-Elor, H., Zhou, X., Snavely, N., 2022. Neural 3d reconstruction in the wild, in: ACM SIGGRAPH.

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al., 2020a. Scalability in perception for autonomous driving: Waymo open dataset, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Sun, Y., Cheng, C., Zhang, Y., Zhang, C., Zheng, L., Wang, Z., Wei, Y., 2020b. Circle loss: A unified perspective of pair similarity optimization, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Tagliasacchi, A., Mildenhall, B., 2022. Volume rendering digest for nerf. arXiv preprint arXiv:2209.02417 .

Tang, H., Liu, Z., Li, X., Lin, Y., Han, S., 2022. TorchSparse: Efficient Point Cloud Inference Engine, in: MLSys, pp. 302–315.

Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., Han, S., 2020. Searching efficient 3d architectures with sparse point-voxel convolution, in: European Conference on Computer Vision (ECCV).

Tang, J., 2022. Torch-ngp: a PyTorch implementation of instant-ngp. Https://github.com/ashawkey/torch-ngp.

Tao, T., Gao, L., Wang, G., Chen, P., Hao, D., Liang, X., Salzmann, M., Yu, K., 2023. Lidar-nerf: Novel lidar view synthesis via neural radiance fields. arXiv preprint arXiv:2304.10406 .

Teed, Z., Deng, J., 2021. RAFT-3D: Scene flow using rigid-motion embeddings, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J., 2019. KPconv: Flexible and deformable convolution for point clouds, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Tombari, F., Salti, S., Di Stefano, L., 2010a. Unique shape context for 3D data description, in: ACM Workshop on 3D Object Retrieval.

Tombari, F., Salti, S., Di Stefano, L., 2010b. Unique signatures of histograms for local surface description, in: European Conference on Computer Vision (ECCV).

Turki, H., Zhang, J.Y., Ferroni, F., Ramanan, D., 2023. Suds: Scalable urban dynamic scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Ulyanov, D., Vedaldi, A., Lempitsky, V., 2016. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 .

Valentin, J., Dai, A., Nießner, M., Kohli, P., Torr, P., Izadi, S., Keskin, C., 2016. Learning to navigate the energy landscape, in: 3DV.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: Advances in Neural Information Processing Systems (NeurIPS).

Vedula, S., Baker, S., Rander, P., Collins, R., Kanade, T., 1999. Three-dimensional scene flow, in: IEEE International Conference on Computer Vision (ICCV).

Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P., 2022. Ref-nerf: Structured view-dependent appearance for neural radiance fields, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Vogel, C., Schindler, K., Roth, S., 2011. 3d scene flow estimation with a rigid motion prior, in: IEEE International Conference on Computer Vision (ICCV).

Vogel, C., Schindler, K., Roth, S., 2013. Piecewise rigid scene flow, in: IEEE International Conference on Computer Vision (ICCV).

Vogel, C., Schindler, K., Roth, S., 2015. 3d scene flow estimation with a piecewise rigid scene model. International Journal of Computer Vision 115, 1–28.

Waechter, M., Moehrle, N., Goesele, M., 2014. Let there be color! large-scale texturing of 3d reconstructions, in: European Conference on Computer Vision (ECCV).

Wagner, W., Ullrich, A., Ducic, V., Melzer, T., Studnicka, N., 2006. Gaussian decomposition and calibration of a novel small-footprint full-waveform digitising airborne laser scanner. ISPRS Journal of Photogrammetry and Remote Sensing 60, 100–112.

Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J., 2019a. Normalized object coordinate space for category-level 6d object pose and size estimation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W., 2021. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 .

Wang, Y., Han, Q., Habermann, M., Daniilidis, K., Theobalt, C., Liu, L., 2023a. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction, in: IEEE International Conference on Computer Vision (ICCV).

Wang, Y., Skorokhodov, I., Wonka, P., 2022. Hf-neus: Improved surface reconstruction using high-frequency details, in: Advances in Neural Information Processing Systems (NeurIPS).

Wang, Y., Solomon, J.M., 2019a. Deep closest point: Learning representations for point cloud registration, in: IEEE International Conference on Computer Vision (ICCV).

Wang, Y., Solomon, J.M., 2019b. PRNet: Self-supervised learning for partial-to-partial registration, in: Advances in Neural Information Processing Systems (NeurIPS).

Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M., 2019b. Dynamic graph CNN for learning on point clouds. ACM TOG 38.

Wang, Z., Shen, T., Gao, J., Huang, S., Munkberg, J., Hasselgren, J., Gojcic, Z., Chen, W., Fidler, S., 2023b. Neural fields meet explicit geometric representations for inverse rendering of urban scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Wedel, A., Rabe, C., Vaudrey, T., Brox, T., Franke, U., Cremers, D., 2008. Efficient dense scene flow from sparse or dense stereo data, in: European Conference on Computer Vision (ECCV).

Weng, X., Wang, J., Held, D., Kitani, K., 2020. 3d multi-object tracking: A baseline and new evaluation metrics, in: IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS).

Wiles, O., Ehrhardt, S., Zisserman, A., 2020. D2D: Learning to find good correspondences for image matching and manipulation. arXiv preprint arXiv:2007.08480 .

Williams, F., 2022. Point cloud utils. Https://www.github.com/fwilliams/point-cloud-utils.

Winiwarter, L., Pena, A.M.E., Weiser, H., Anders, K., Sánchez, J.M., Searle, M., Höfle, B., 2022. Virtual laser scanning with HELIOS++: A novel take on ray tracing-based simulation of topographic full-waveform 3d laser scanning. Remote Sensing of Environment 269, 112772.

Wu, P., Chen, S., Metaxas, D.N., 2020a. MotionNet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Wu, T., Wang, J., Pan, X., Xu, X., Theobalt, C., Liu, Z., Lin, D., 2022a. Voxurf: Voxel-based efficient and accurate neural surface reconstruction. arXiv preprint arXiv:2208.12697 .

Wu, T., Zhong, F., Tagliasacchi, A., Cole, F., Oztireli, C., 2022b. Dˆ 2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. Advances in Neural Information Processing Systems (NeurIPS) .

Wu, W., Wang, Z.Y., Li, Z., Liu, W., Fuxin, L., 2020b. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation, in: European Conference on Computer Vision (ECCV).

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Xiao, J., Owens, A., Torralba, A., 2013. Sun3d: A database of big spaces reconstructed using sfm and object labels, in: IEEE International Conference on Computer Vision (ICCV).

Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., Sridhar, S., 2022. Neural fields in visual computing and beyond, in: Computer Graphics Forum, Wiley Online Library. pp. 641–676.

Yang, B., Bai, M., Liang, M., Zeng, W., Urtasun, R., 2021a. Auto4d: Learning to label 4d objects from sequential point clouds. arXiv preprint arXiv:2101.06586 .

Yang, J., Ivanovic, B., Litany, O., Weng, X., Kim, S.W., Li, B., Che, T., Xu, D., Fidler, S., Pavone, M., Wang, Y., 2023a. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. arXiv preprint arXiv:2311.02077 .

Yang, Z., Chen, Y., Wang, J., Manivasagam, S., Ma, W.C., Yang, A.J., Urtasun, R., 2023b. Unisim: A neural closed-loop sensor simulator, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Yang, Z., Chen, Y., Wang, J., Manivasagam, S., Ma, W.C., Yang, A.J., Urtasun, R., 2023c. Unisim: A neural closed-loop sensor simulator, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Yang, Z., Pan, J.Z., Luo, L., Zhou, X., Grauman, K., Huang, Q., 2019. Extreme relative pose estimation for rgb-d scans via scene completion, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Yang, Z., Yan, S., Huang, Q., 2020. Extreme relative pose network under hybrid representations, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Yang, Z., Zhou, Y., Chen, Z., Ngiam, J., 2021b. 3D-MAN: 3d multi-frame attention network for object detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Yariv, L., Gu, J., Kasten, Y., Lipman, Y., 2021. Volume rendering of neural implicit surfaces. Advances in Neural Information Processing Systems (NeurIPS) .

Yew, Z.J., Lee, G.H., 2018. 3dfeat-net: Weakly supervised local 3d features for point cloud registration, in: European Conference on Computer Vision (ECCV), Springer. pp. 630–646.

Yew, Z.J., Lee, G.H., 2020. RPM-Net: Robust point matching using learned features, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Yu, A., Fridovich-Keil, S., Tancik, M., Chen, Q., Recht, B., Kanazawa, A., 2021. Plenoxels: Radiance fields without neural networks. arXiv preprint arXiv:2112.05131 .

Yu, Z., Peng, S., Niemeyer, M., Sattler, T., Geiger, A., 2022. MonoSDF: Exploring monocular geometric cues for neural implicit surface reconstruction. Advances in Neural Information Processing Systems (NeurIPS) .

Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M., 2018. PCN: Point completion network, in: IEEE International conference on 3D vision (3DV).

Yuan, W., Lv, Z., Schmidt, T., Lovegrove, S., 2021. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., Funkhouser, T., 2017. 3DMatch: learning local geometric descriptors from RGB-D reconstructions, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Zhang, J., Zhang, F., Kuang, S., Zhang, L., 2023. Nerf-lidar: Generating realistic lidar point clouds with neural radiance fields. arXiv preprint arXiv:2304.14811 .

Zhang, K., Riegler, G., Snavely, N., Koltun, V., 2020. NeRF++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 .

Zhou, Q.Y., Park, J., Koltun, V., 2018. Open3D: A modern library for 3D data processing. arXiv preprint arXiv:1801.09847 .

Zuo, X., Yang, N., Merrill, N., Xu, B., Leutenegger, S., 2023. Incremental dense reconstruction from monocular video with guided sparse feature volume fusion. IEEE Robotics and Automation Letters .

Zyrianov, V., Zhu, X., Wang, S., 2022. Learning to generate realistic LiDAR point clouds, in: European Conference on Computer Vision (ECCV), Springer.

B List of Publications

1. **S. Huang**, M. Usvyatsov, K. Schindler. ”Indoor Scene Recognition in 3D.” IROS, 2020.
2. **S. Huang***, Z. Gojcic*, M. Usvyatsov, A. Wieser, K. Schindler. ”PREDATOR: Registration of 3D Point Clouds with Low Overlap.” CVPR, 2021. (Oral)
3. **S. Huang**, Z. Gojcic, J. Huang, A. Wieser, K. Schindler ”Dynamic 3D Scene Analysis by Point Cloud Accumulation.” ECCV, 2022.
4. **S. Huang**, Z. Gojcic, Z. Wang, F. William, Y. Kasten, S. Fidler, K. Schindler, O. Litany ”Neural LiDAR Fields for Novel View Synthesis.” ICCV, 2023.
5. H. Wu, X. Zuo, S. Leutenegger, Or Litany, K. Schindle, **S. Huang** ”Dynamic LiDAR Re-simulation using Compositional Neural Fields.” CVPR, 2024.
6. L. Zhu, **S. Huang**, K. Schindler, I. Armeni. ”Living Scenes: Multi-object Relocalization and Reconstruction in Changing 3D Environments.” CVPR, 2024.
7. B. Ke, A. Obukhov, **S. Huang**, N. Metzger, R. Daudt, K. Schindler. ”Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation.” CVPR, 2024.
8. Y. Jia, L. Hoyer, **S. Huang**, T. Wang, L. Gool, K. Schindler, A. Obukhov. ”DGInStyle: DomainGeneralizable Semantic Segmentation with Image Diffusion Models and Stylized Semantic Control.” CVPR, 2024.
9. Z. Wang, T. Shen, J. Gao, **S. Huang**, J. Munkberg, J. Hasselgren, Z. Gojcic, W. Chen, S. Fidler ”Neural Fields meet Explicit Geometric Representations for Inverse Rendering of Urban Scenes.” CVPR, 2023.
10. L. Zhu, Y. Jia, **S. Huang**, N. Meyer, A. Wieser, K. Schindler, J. Aaron ”DeFlow: Self-supervised 3D Motion Estimation of Debris Flow.” CVPR Workshop, 2023. (Best Paper Award)
11. C. Stucker, B. Ke, Y. Yue, **S. Huang**, I. Armeni, K. Schindler ”ImpliCity: City Modeling from Satellite Images with Deep Implicit Occupancy Fields.” ISPRS Congress, 2022. (Best Young Author Award)
12. T. Sun, Y. Hao, **S. Huang**, S. Savarese, K. Schindler, M. Pollefeys, I. Armeni ”Nothing stands still: A spatiotemporal benchmark on 3d point cloud registration under large geometric and temporal change.” arxiv, 2023

C | Curriculum Vitae

Personal data

Name	**Shengyu Huang**
Date of birth	**17.11.1995**
Nationality	**China**

Education

Oct. 2020 - Nov. 2024	**ETH Zurich, Switzerland** PhD Student
Sep. 2018 - Aug. 2020	**ETH Zurich, Switzerland** M.Sc. in Geomatik
Sep. 2014 - Aug. 2018	**Tongji University, China** BSc. in Surveying and Mapping Engineering

Professional Experience

Oc. 2020 - Nov. 2024	**ETH Zurich, Switzerland** Research and Teaching Assistant
Jul 2023 - Dec 2023	**Google, Munich, Germany** Student Researcher
Apr 2022 - Dec 2022	**NVIDIA, Zurich, Switzerland** Research Scientist Intern