NoKSR: Kernel-Free Neural Surface Reconstruction via Point Cloud Serialization

Zhen (Colin) Li[1*] Weiwei Sun[2,3*†] Shrisudhan Govindarajan[1]
Shaobo Xia[4] Daniel Rebain[2] Kwang Moo Yi[2] Andrea Tagliasacchi[1,5,6]

[1]Simon Fraser University [2]University of British Columbia [3]Amazon
[4]Changsha University of Science and Technology [5]University of Toronto [6]Google DeepMind

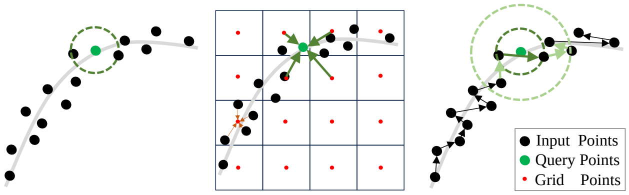
theialab.github.io/noksr

Abstract

We present a novel approach to large-scale point cloud surface reconstruction by developing an efficient framework that converts an irregular point cloud into a signed distance field (SDF). Our backbone builds upon recent transformer-based architectures (i.e. PointTransformerV3), that serializes the point cloud into a locality-preserving sequence of tokens. We efficiently predict the SDF value at a point by aggregating nearby tokens, where fast approximate neighbors can be retrieved thanks to the serialization. We serialize the point cloud at different levels/scales, and non-linearly aggregate a feature to predict the SDF value. We show that aggregating across multiple scales is critical to overcome the approximations introduced by the serialization (i.e. false negatives in the neighborhood). Our frameworks sets the new state-of-the-art in terms of accuracy and efficiency (better or similar performance with half the latency of the best prior method, coupled with a simpler implementation), particularly on outdoor datasets where sparse-grid methods have shown limited performance.

1. Introduction

Reconstructing the surface sampled by a point cloud is a fundamental problem with many applications in robotics [41], autonomous driving [7], and virtual reality [17, 53]. We tackle this task by predicting the *signed distance field* (SDF) associated with a given point cloud: a function that returns the signed distance to the nearest surface for any given 3D position. Given the distance field, the surface can be extracted by finding the zero-crossings of the distance function.



		CD (10^{-2}) ↓	F-Score ↑	Latency (s) ↓
K-Nearest Neighbors	oracle	3.2	97.5	3.2
Ours (Minkowski) [11]	grid-based	3.8	96.2	1.5
Ours	serialization	3.3	97.4	1.7

Figure 1. To locally predict the SDF value that (implicitly) reconstructs the surface, the pivotal operation is to *aggregate* the information (i.e. features) of nearby points. (left) Working on the point cloud directly is difficult, as there is no simple way to implement multi-scale architectures suitable for large scale point cloud processing. (middle) State-of-the-art methods therefore opt to quantize the input point cloud to a voxel grid, and employ established sparse CNN backbones, but quantization leads to information loss. (right) By fetching approximate neighbors via serialization we can fetch the local context efficiently *and* avoid information loss. We summarize the performance of representative works on a large scale outdoor dataset (CARLA [13]), and show that our method achieves the best performance in both time efficiency (latency) and accuracy (CD and F-score); for additional details see Section 4.

State-of-the-art approaches such as NKSR [22] and NeuralUDF [28] train a point cloud backbone to predict the distance value for any position in space. Their backbones are trained on a collection of scenes, so as to capture the priors within the data that allows reconstruction to be performed even when the problem is *ill-posed* (e.g., when the point cloud is sparse and/or incomplete).

The core operation within these backbones is to predict a feature that *aggregates* the information of input points near the query, which is then decoded to an SDF value. In state-of-the-art models, these aggregation operations are realized by implementing multi-scale sparse convolutional neural networks [11, 43]. To be able to scale to large-scale point

[*] Equal contribution.
[†] Work performed while at the University of British Columbia.

clouds, these backbones require voxelizing the input point cloud, and summarizing the information of points therein: a spatial *quantization* operation that inevitably leads to information loss. This quantization operation is detrimental when real-world point clouds are used, as the non-uniformity of sampling leads to performance degradation.

Rather than relying on spatial quantization and sparse-CNNs, we build upon PointTransformerV3 [48], and aggregate information by relying on locality-preserving serialization: we serialize the input point cloud to an ordered list, so that nearby points in the list are in close Euclidean proximity; see Figure 1. The serialization transformation *does not* incur information loss due to quantization, and it offers superior computational efficiency in terms of feature aggregation compared to methods based on voxelization.

With serialization, retrieving the local neighbors to aggregate our features can result in *false negatives*: points can be close in Euclidean space, but far in their serialized index. To circumvent this problem, we retrieve features from approximate nearest neighbors *across* several serialization levels, as provided by [48], and then aggregate them with a PointNet architecture [6] to predict the signed distance function.

Compared to state-of-the-art techniques, our framework requires neither heavily engineered sparse processing backbones [28], nor differentiating through linear systems of equations [22, 47]. Nonetheless, this simple framework outperforms the state-of-the-art in *both* time efficiency and reconstruction quality on *multiple datasets* including ScanNet [12], SceneNN [20], Carla [13, 22], Synthetic-Room [36].

Given the dominance of voxel-based data structures in surface reconstruction from point clouds, we demonstrate that carefully designed *point-based* architectures can also be highly effective for this task, and we hope this will inspire *renewed* interest in the research area.

2. Related works

Explicit 3D representations such as points [1], voxels [40], meshes [15], and polygonal surfaces [33] are commonly used for visualization and reconstruction [23]. However, the discrete structures of these representations are challenging to adapt for learning-based approaches [36, 38] which rely on differentiability. As a result, implicit 3D representations via neural networks have gained popularity [35] as they can be converted to an explicit model after training by techniques like Marching Cubes [29]. This paper addresses the task of reconstructing neural implicit surface representations from point clouds.

Neural implicit surface reconstruction. Neural implicit methods utilize neural networks to model an occupancy field [31, 34] or a distance field [21, 35] for surface reconstruction. Distance fields, including signed distance fields (SDF) [24, 30, 35] and unsigned distance fields (UDF) [3, 10], are functions whose zero level set implicitly defines the object surface. A learnt SDF predicts a query point's signed distance to the nearest surface, with a negative value indicating the point is inside the surface and a positive value indicating it is outside [38]. To encode unstructured point clouds into neural fields, various network architectures have been proposed, such as MLPs [9, 31, 35], infinitely-wide-ReLU networks [46], PointNet [39, 47], 3D-UNet [45], RandLA-Net [3], sparse hierarchical networks [22], and MinkowskiNet [28]. Compared to encoding the point cloud into a global feature [14, 35], organizing point clouds into regular or irregular grids or voxels for feature learning and spatial querying preserves more details [25, 26, 36, 47, 50, 52]. For example, [38] uses a voxel octree to collect point-wise features, and retrieves the query point feature via trilinear interpolation at each tree level. Alternatively, feature interpolation can be implemented with an attention-based [45] or learning-based approach [4]. Such data-driven methods often struggle to guarantee the accuracy of learned surfaces, and are difficult to scale and generalize [21, 47]. Huang et al. [22] addresses this problem by solving complex kernel functions on hierarchical voxels. However, the quantization inherent to voxels or grids leads to information loss, and the solver increases the reconstruction time quadratically with the number of grid cells. To solve these problems, we propose a point-based framework powered by a serialization encoding for implicit surface reconstruction.

Efficient point cloud networks. Point-based networks [6, 19] achieve good performance on small datasets, but in applications to large point cloud data their message-passing strategy is not sufficiently computationally efficient. Sparse convolution networks [11, 16] based on voxelization are fast but suffer from information loss. The recently proposed OctFormer [43] and PointTransformerV3 [48] provide a superior combination of efficiency and encoding performance by leveraging a serialization-based strategy, and our method builds upon these approaches.

Point cloud serialization. Backbone networks that rely on voxelization (e.g., MinkowskiNet [11] and sub-manifold sparse U-Net [16]) suffer of high computational cost, and from information loss due to quantization. To avoid these limitations, point cloud serialization methods encode irregular point clouds into sequential structures with the use of space-filling curves. This bijective encoding scheme excels at dimension reduction, preserving topology and retaining locality, making it a promising approach to address voxelization issues [42, 44]. Chen et al. [8] leverages the Hilbert curve [18] to map voxels into an ordered sequence, enabling the use of 2D convolution and Transformers on

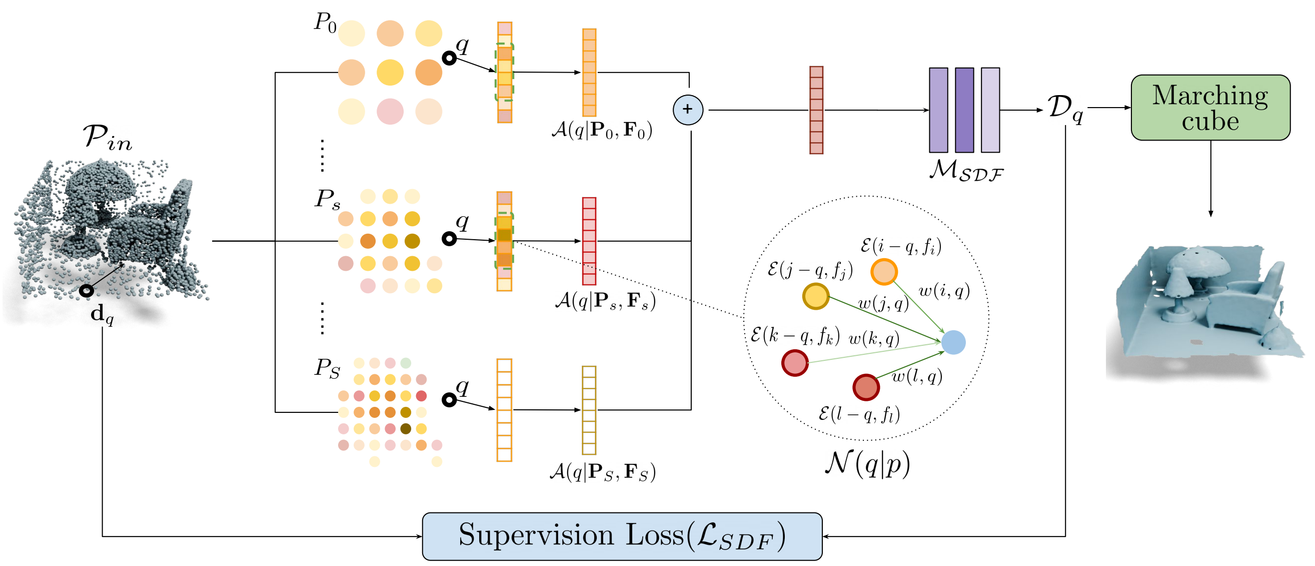


Figure 2. **Overview**: We map a sparse input point cloud with a point cloud backbone [48] into a point feature hierarchy, from which we compute the signed distance of a query. At each level, we utilize the efficient procedure defined in Section 3.2.1 to retrieve local neighborhoods of the query. We then compute per-level features with the aggregation module defined in Section 3.2. At last, we sum per-level features and convert it into the signed distance with an MLP.

3D voxels. Wang [43], by employing z-order curves [32] to sort octree nodes, achieves equal-sized point partitions and constructs an effective octree attention module for point clouds. The idea of equal-sized sorting of windows is also adopted in [27]. To mitigate the computational overhead of K-Nearest Neighbor (KNN), Wu et al. [48] integrates z-order and Hilbert curves to map 3D points into structured sequences and patches, upon which attention layers are constructed. Zhang et al. [51] introduces a Hilbert input layer for serializing 3D voxels, laying the foundation for a voxel-based state space model designed for 3D object detection. This approach eliminates the need for 1D sequence grouping and padding. In our work, we present a point feature retrieval algorithm that operates directly on points based on point cloud serialization, and demonstrate its performance for neural surface reconstruction applications.

3. Method

The overview of our method is illustrated in Figure 2. Given a point cloud $\mathbf{P}\in\mathbb{R}^{N\times 3}$ of N points in 3-dimensional space, we compute hierarchical point features $\mathbf{F}$ with S levels using a point-based transformer $\mathcal{F}$ parameterized via $\boldsymbol{\theta}_{\mathcal{F}}$ as

$$\{(\mathbf{P}_s, \mathbf{F}_s)\}_{s=1}^{S} = \mathcal{F}(\mathbf{P}; \boldsymbol{\theta}_{\mathcal{F}}) \tag{1}$$

where with the s subscript we denote the point cloud and learned features at s-th level. For a given query $\mathbf{q} \in \mathbb{R}^3$, we employ the features from the feature hierarchy to predict its distance field value $\mathbf{d}$:

$$\mathbf{d} = \mathcal{D}(\mathbf{q} \mid \{(\mathbf{P}_s, \mathbf{F}_s)\}; \boldsymbol{\theta}_{\mathcal{D}}) \tag{2}$$

where $\mathcal{D}$ has learnable parameters $\boldsymbol{\theta}_{\mathcal{D}}$.

3.1. Distance field – $\mathcal{D}$

For each query $\mathbf{q}$, we calculate a per-level feature from the feature hierarchy through an *aggregation* module $\mathcal{A}$, and then sum the features of all levels as the query's feature, which is then mapped to an SDF value by $\mathcal{M}$:

$$\mathcal{D}(\mathbf{q}) = \mathcal{M}\left(\sum_{s=1}^{S} \mathcal{A}\left(\mathbf{q}|\mathbf{P}_s, \mathbf{F}_s\right)\right) \tag{3}$$

Following Huang et al. [22], $\mathcal{M}$ is simply an MLP with a single hidden layer followed by a *tanh* activation, and its parameters are included in the set $\boldsymbol{\theta}_{\mathcal{D}}$. We now describe the aggregation module $\mathcal{A}$ in more details.

3.2. Aggregation module – $\mathcal{A}$

At the s-th level, we retrieve the local neighborhood at the query location and use a PointNet-style network to map the local point cloud into the per-level feature:

$$\mathcal{A}(\mathbf{q}|\mathbf{P}_s, \mathbf{F}_s) = \frac{\sum_{\mathbf{p}\in\mathcal{N}(\mathbf{q}|\mathbf{P}_s)} w(\mathbf{p}, \mathbf{q}) \cdot \mathcal{E}(\mathbf{p}-\mathbf{q}, \mathbf{f}_{\mathbf{p}})}{\epsilon + \sum_{\mathbf{p}\in\mathcal{N}(\mathbf{q}|\mathbf{P}_s)} w(\mathbf{p}, \mathbf{q})} \tag{4}$$

where $w(\mathbf{p}, \mathbf{q})$ is the inverse spatial distance, $\epsilon{=}1e{-}8$ avoids division by zero, $\mathcal{E}$ is a small MLP whose parameters are included in the set $\boldsymbol{\theta}_{\mathcal{D}}$, $\mathbf{f}_{\mathbf{p}}$ is the feature in $\mathbf{F}$ at $\mathbf{p}$,

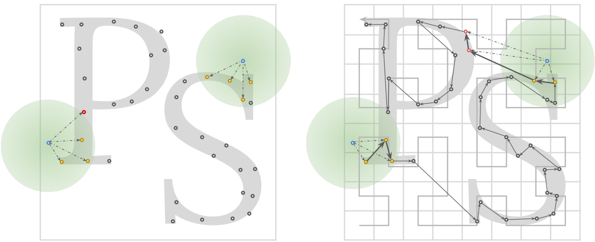


Figure 3. **Neighborhood function** – (left) retrieving a local neighborhood with K-nearest neighbor(KNN) or ball-query methods is challenging to implement efficiently on GPU hardware. (right) we propose to retrieve a neighborhood from a 1-D ordered list, by serializing points along a Hibert curve [18], and excluding the impact of points distant from the query (i.e. remove false positives).

$\mathcal{N}(\mathbf{q}|\mathbf{p})$ is a function that retrieves the local neighborhood of $\mathbf{q}$ from $\mathbf{P}_s$.

3.2.1 Neighborhood function – $\mathcal{N}$

Retrieving neighbors via k-nearest neighbor (KNN) or ball-query methods would be optimal, but these are difficult to implement efficiently on GPU hardware. As the reconstruction pipeline is sensitive to the computational cost of this operation, we choose to leverage a more efficient strategy. In particular, we implement our approximate neighborhood lookup $\mathcal{N}$ on the locality-preserving *serialization* encoding proposed by [43, 48]. A serialization encoding is a hash function ($\gamma : \mathbb{R}^3 \hookrightarrow \mathbb{Z}$) that maps a point to a integer. Given a point $point \in \mathbb{R}^3$, we calculate the integer as

$$\gamma = \phi(\lfloor p/g \rfloor)) \tag{5}$$

where $\lfloor p/g \rfloor$ is a floor function that quantizes a point with real-valued coordinates to the integral coordinates of cells in a 3-dimensional grid with size g, and ϕ is a bijective function that maps 3D coordinates $\mathbb{Z}^3$ to 1D values $\mathbb{Z}$. We define the bijective ϕ as a space filling curve, which traverses 3D space in a *locality-preserving* order. We utilize Hilbert curves [18], and to avoid collisions in the quantization of point coordinates to grid cells, we use a very fine grid resolution across all levels, as there is no cost associated with increasing the resolution of this *virtual* grid. As illustrated in Figure 3, to retrieve the local neighborhood of a query from a point cloud, we first encode the point cloud into a set of sorted integers using γ. We then apply γ to the query coordinate and search through its neighbors on the 1D line to identify close-by points in Euclidean space.

3.3. Training

To train our networks, we optimize the loss:

$$\underset{\boldsymbol{\theta}_{\mathcal{F}}, \boldsymbol{\theta}_{\mathcal{D}}, \boldsymbol{\theta}_{\mathcal{C}}}{\arg\min} \quad \lambda_{\text{SDF}} \mathcal{L}_{\text{SDF}} + \lambda_{\text{Eikonal}} \mathcal{L}_{\text{Eikonal}} + \lambda_{\text{mask}} \mathcal{L}_{\text{mask}} \tag{6}$$

where λ_{SDF} , λ_{eikonal} and λ_{mask} are the coefficients for loss terms, which we will detail below.

Signed distance function supervision – $\mathcal{L}_{\text{SDF}}$. We define $\mathcal{L}_{\text{SDF}}$ to reproduce the ground truth SDF value $\mathbf{d}_q$ at $\mathbf{q}$:

$$\mathcal{L}_{\text{SDF}} = \mathbb{E}_{\mathbf{q} \sim \mathcal{Q}} \left[||\mathbf{d}_{\mathbf{q}} - \mathcal{D}(\mathbf{q})||_1 \right]$$

where $\mathcal{Q}$ is the distribution from Huang et al. [22].

Surface regularizer – $\mathcal{L}_{\text{Eikonal}}$. We regularize the field $\mathcal{D}$ with an Eikonal loss to encourage this function to be a signed distance field away from the surface:

$$\mathcal{L}_{\text{Eikonal}} = \mathbb{E}_{\mathbf{x} \sim \mathcal{Q}} [(||\nabla_{\mathbf{x}} \mathcal{D}(\mathbf{x})\|_2 - 1)^2] \tag{7}$$

Auxiliary loss – $\mathcal{L}_{\text{mask}}$. Following [22], the classification branch $\mathcal{C}$ with learnable parameters $\boldsymbol{\theta}_{\mathcal{C}}$ classifies queries as near/far from the surface as supervised by the loss:

$$\mathcal{L}_{\text{mask}} = \mathbb{E}_{\mathbf{q} \sim \mathcal{Q}} [\mathbf{CE}(\mathbf{c}_{\mathbf{q}}, \mathcal{C}(\mathbf{q}))]$$

where **CE** is the cross entropy function, $\mathbf{c}_{\mathbf{q}}$ is ground-truth binary label calculated by thresholding the ground-truth SDF with empirically chosen values of 0.015 meters for indoor scenes, and 0.1 meters for outdoor. At inference time, the output of this classifier helps avoid reconstructing surfaces that are far from the input point cloud (i.e. not supported by input point-cloud data).

4. Results

Following NKSR [22], we evaluate our method using metrics including the standard Chamfer-L_1 Distance (CD-$L_1 \times 10^{-2}$, $\downarrow$) and F-score ($\uparrow$) with a threshold (δ=0.010). We also report additional metrics proposed in NKSR [22] including Chamfer-L_1 Distance by Completeness (Comp. $\times 10^{-2}$, $\downarrow$) and Accuracy (Acc. $\times 10^{-2}$, $\downarrow$) in the `Supplementary Material`. We evaluate our method on multiple datasets, under two settings including in-domain evaluation for accuracy estimation – training set and test set are from same dataset, and cross-domain evaluation for generalization ability estimation where training set and test set are from different datasets. Additionally, for cross-domain evaluation we use the following datasets prepared by the leading voxel-based baseline, NKSR [22], and one additional dataset from RangeUDF [3]:

- `SyntheticRoom` [36] is a synthetic dataset created from ShapeNet objects [5]. Each scene contains 2-3 objects. Following prior works [3, 10], we re-scale the synthetic rooms to roughly match real-world scale. There are 3750 scenes as training set and 995 scenes as the test set.
- `ScanNet` [12] is a real-world indoor scene dataset. We use the setting from previous work [3, 4, 36, 39] where we train on 1201 rooms and test on 312 rooms.

- CARLA [13] is a large-scale outdoor driving scene prepared by NKSR [22] using the CARLA simulator [13]. Following NSKR [22], we test on two subsets including the 'Original' subset (10 random drives simulated on 3 towns) and the 'Novel' subset (3 drives from an additional town only for testing). To avoid exploding GPU memory during training, we follow NKSR [22] to divide a large scene into patches. The resultant training set has 3757 patches.
- SceneNN [20] is a real-world indoor dataset prepared by RangeUDF [3] which we used for cross-domain evaluation. We only use its test set which consists of 20 scenes.

Evaluation pipeline. To evaluate our method, we first extract the mesh with Dual Marching Cubes [37] on the predicted SDF, and then compute the CD and F-score between 100k points sampled on the mesh, and 100k points sampled from the ground-truth dense point cloud. We use the same approach as NKSR [22] to prepare the input point clouds for training and evaluation from the ground-truth dense point clouds through downsampling. Specifically, for indoor datasets (i.e., SyntheticRoom [36], ScanNet [12] and SceneNN [20]), we uniformly sample 10K points sampled from the ground truth dense point cloud. For outdoor driving scenes (i.e., CARLA [13]), we follow the evaluation pipeline from NKSR [22]. We sample sparse input point clouds with a sparse 32-beam LiDAR with a ray distance noise of 0-5 cm and pose noise of $0 - 3°$, and obtain the ground truth from a noise-free dense 256-beam LiDAR.

Implementation details. We base our feature backbone on PointTransformerV3 [48] with 4-levels. The PointNet-style network is a 2-layered residual connection MLP, with hidden dimension of 32 and output feature dimension of 32. The grid size used in neighborhood function is 0.01 meters. Following NKSR [22], we use the similar coefficients for loss terms – i.e., λ_{SDF} is 300 and λ_{mask} is 150. However, we empirically set λ_{Eikonal} to 10 (NKSR [22] does not need this regularizer thanks to its specialized surface solver). We train our model with a batch size of 4 on either a single NVIDIA RTX A6000 ADA or an NVIDIA L40S, and a learning rate of 10^{-3}. We adopt the Adam optimizer with default parameters. We set the maximum number of epochs to 200 and employ a cosine learning rate decay starting from epoch 120.

Reconstruction latency. For both our models and NKSR, we record the reconstruction latency for all indoor scenes on a single NVIDIA RTX 3090, and for large outdoor scenes on a single NVIDIA L40s given that more GPU memory is required. We omit data loading time, and only record the average forward pass time.

4.1. In-domain evaluation

We compare against NKSR [22] (the current state-of-the-art), RangeUDF [3], SPSR [23], NDF [10], ConvOcc [36] and SA-CONet [39]. We further include a baseline that replaces our backbone with MinkowskiNet [11] (i.e., Ours (Minkowski)) to show the degraded performance due to the information loss caused by voxelization.

Quantitative results – Table 1. Across indoor and outdoor datasets, our method outperforms baselines in terms of accuracy and time efficiency. Especially in outdoor datasets, our method achieves the best surface reconstruction with the smallest latency – nearly *half* of the second best's latency. In indoor datasets, which have relatively uniform sampling patterns, we achieve accuracy on par with the previous state-of-the-art, but with significantly improved time efficiency. Note that we achieve this advantage even with KNN because, in smaller indoor point clouds, the highly engineered KNN implementation has similar time efficiency to that of our neighborhood function. We further detail our analysis on this matter in the Supplementary Material. We also note that our approximate neighborhood function is still effective, as it outperforms the directly comparable baseline MinkowskiNet [11], which shares the same structure except for the backbone and neighborhood function.

Qualitative results – Figures 4 and 5. We show that our method tends to reconstruct surfaces of the best quality among the compared methods. Especially, on the non-uniform large scale CARLA [13], our method tends to preserve more details than the previous state-of-the-art [22], which voxelizes the point cloud.

4.2. Cross-domain evaluation – Table 2

We further test the generalization ability of our method with a cross-domain evaluation. We evaluate models trained with dataset A on other a different dataset B; we denote this as A → B. As shown in Table 2, there are three cases in total. In two cases (i.e., SyntheticRoom [36] → ScanNet [12] and ScanNet [12] → SyntheticRoom [36]), our method achieves the best accuracy with the best time efficiency. In another case (ScanNet [12] → SceneNN [20]), we achieve accuracy on par with SOTA [22] with a much better time efficiency, i.e., less than a half of the latency required by the SOTA [22].

4.3. Ablation studies

Our ablations are executed on ScanNet [12], as it is a real-world dataset, and is equipped with precise ground truth surface meshes.

Impact of neighborhood size – Table 3. We analyze the impact of neighborhood size on performance. Larger neigh-

Methods		SyntheticRoom [36]			ScanNet [12]			CARLA [13] (Original)			CARLA [13] (Novel)		
	Primitive	CD (10^{-2}) ↓	F-Score ↑	Latency (s) ↓	CD (10^{-2}) ↓	F-Score ↑	Latency (s) ↓	CD (cm) ↓	F-Score ↑	Latency (s) ↓	CD (cm) ↓	F-Score ↑	Latency (s) ↓
SA-CONet [39]	Voxels	0.496	93.60	-	-	-	-	-	-	-	-	-	-
ConvOcc [36]	Voxels	0.420	96.40	-	-	-	-	-	-	-	-	-	-
NDF [10]	Voxels	0.408	95.20	-	0.385	96.40	-	-	-	-	-	-	-
RangeUDF [3]	Voxels	0.348	97.80	-	0.286	98.80	-	-	-	-	-	-	-
TSDF-Fusion [49]	-	-	-	-	-	-	-	8.1	80.2	-	7.6	80.7	-
POCO [4]	-	-	-	-	-	-	-	7.0	90.1	-	12.0	92.4	-
SPSR [23]	-	-	-	-	-	-	-	13.3	86.5	-	11.3	88.3	-
NKSR [22]	Voxels	0.346	97.41	0.40	0.246	99.51	1.54	3.9	93.9	2.0	2.9	96.0	1.8
NKSR [22] (more data)	Voxels	-	-	-	-	-	-	3.6	94.0	2.0	3.0	96.0	1.8
Ours (Minkowski) [11] (w/ KNN)	Voxels	-			0.254	99.41	0.46	3.4	97.2	1.9	2.7	98.1	2.0
Ours (Minkowski) [11]	Voxels	-			0.301	98.48	0.31	3.8	96.2	1.5	3.0	97.4	1.5
Ours (w/ KNN)	Points	0.321	98.34	0.13	0.243	99.61	0.48	3.2	97.5	3.2	2.6	98.3	3.4
Ours	Points	0.360	96.32	0.14	0.257	99.33	0.49	3.3	97.4	1.7	2.7	98.2	1.7

Table 1. **In-domain evaluation** – We show that our method achieves the best accuracy (CD and F-score) with significantly improved time efficiency (inference latency). Note we retrain NKSR [22] (numbers are underlined) for fairer comparison, as the training data for NKSR [22] is different from ours – i.e., they reported some models trained on a "mix" of datasets, which is impossible to reproduce.

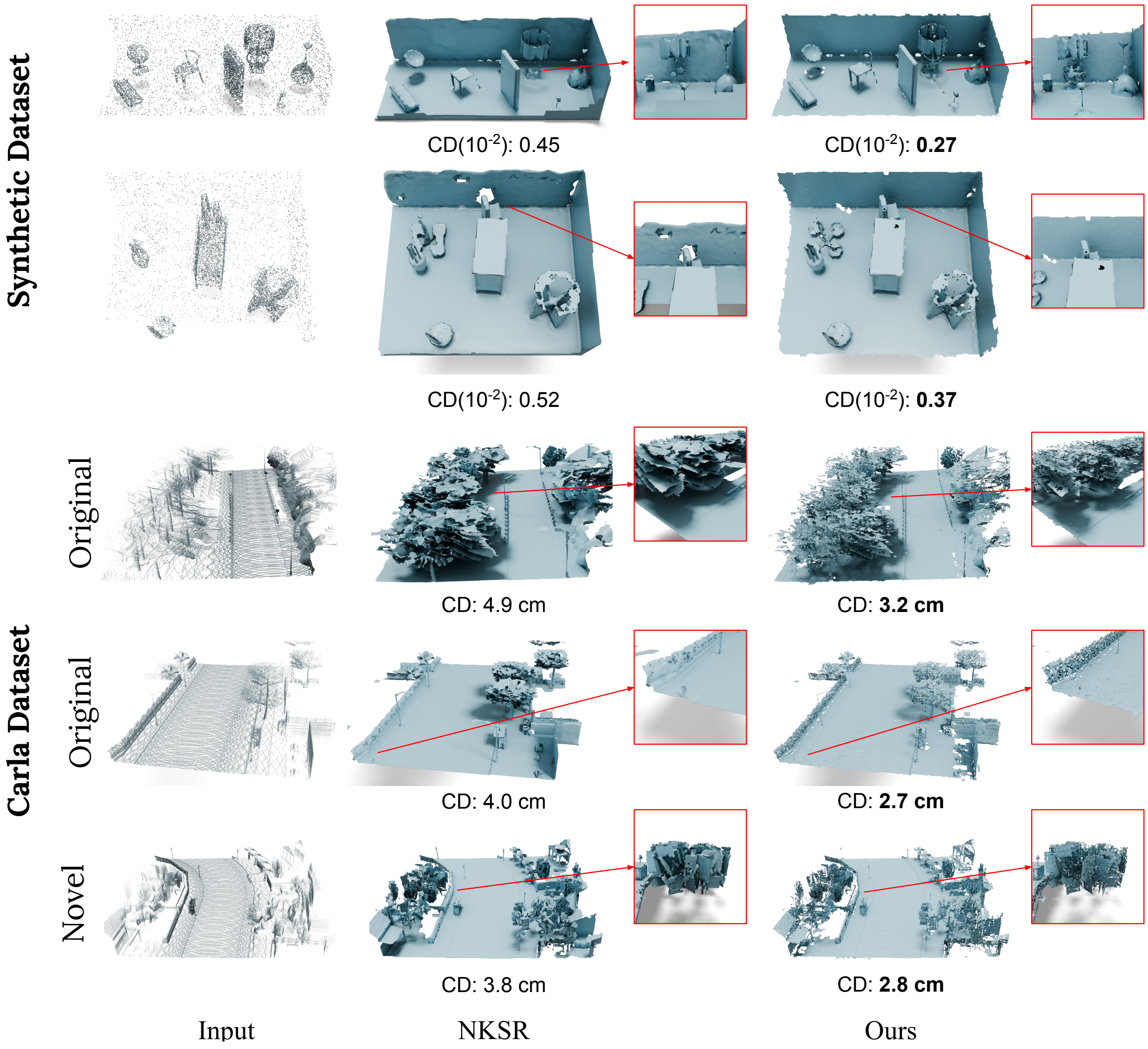


Figure 4. **Qualitative results on CARLA [13] and SyntheticRoom [36]** – our method achieves high quality surface reconstructions which preserve more details than NKSR [22] which loses information due to quantization for large and non-uniformly sampled datasets like Carla.

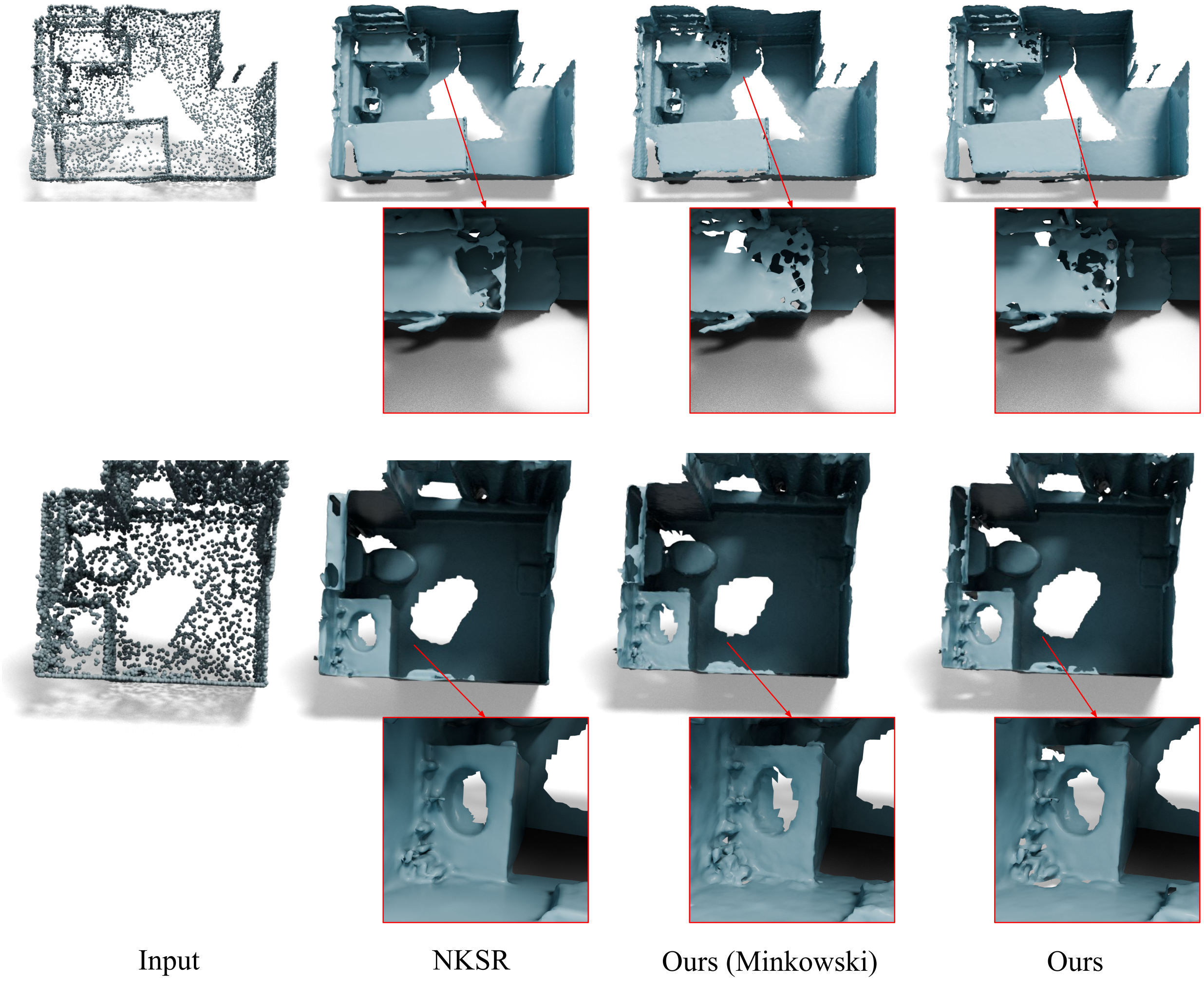


Figure 5. Qualitative results on `ScanNet` [12]: We compare our method with prior SOTA [22] and Ours (Minkowski) [11] that is more comparable as it only differs from ours in the backbone. Our method achieves reconstruction of similar quality to the SOTA. It also *significantly* outperforms Ours (Minkowski), highlighting the importance of point-based methods.

Methods	Primitive	**`SyntheticRoom`** [36] → **`ScanNet`** [12]			**`ScanNet`** [12] → **`SyntheticRoom`** [36]			**`ScanNet`** [12] → **`SceneNN`** [20]		
		CD (10^{-2}) ↓	F-Score ↑	Latency (s) ↓	CD (10^{-2}) ↓	F-Score ↑	Latency (s) ↓	CD (10^{-2}) ↓	F-Score ↑	Latency (s) ↓
SA-CONet [39]	Voxels	0.845	77.80	-	-	-	-	-	-	-
ConvOcc [36]	Voxels	0.776	83.30	-	-	-	-	-	-	-
NDF [10]	Voxels	0.452	96.00	-	0.568	88.10	-	0.425	94.80	-
RangeUDF [3]	Voxels	0.303	98.60	-	0.481	91.50	-	0.324	97.80	-
NKSR [22]	Voxels	0.329	97.37	2.02	0.351	97.41	0.46	0.268	99.18	1.95
Ours (w/ KNN)	Points	0.284	98.65	0.54	0.327	98.37	0.13	0.277	99.00	0.50

Table 2. **Cross-domain evaluation** – we achieve the best generalization ability in two cases with much better time efficiency. In the other case where we generalize from `ScanNet` [12] to `SceneNN` [20], we achieve accuracy on par with the SOTA baseline [22] with less than a half of their latency.

borhood size leads to increased computation overhead. We show that the 8-nearest neighboring points gives the best trade-off between accuracy and time efficiency. Considering a large number (e.g., 16) of neighboring points degrades performance as the the aggregation module has limited capacity to predict the precise SDF from a large local point cloud.

Impact of capacity of $\mathcal{A}$ – Table 4. We report how the capacity of the aggregation module $\mathcal{A}$ (i.e., different number of hidden layers) impacts the performance. We observe that aggregation modules of higher capacity give better performance but degraded time efficiency. However, as shown in Table 4, a very large capacity (4 layers) for $\mathcal{A}$ does not help. We show that we we use 2 layers to have a good

Neighbor Num.	CD (10^{-2}) ↓	F-score ↑	Latency (s) ↓
2	0.246	99.56	109
4	0.244	99.59	127
8	0.243	99.61	151
16	0.256	99.28	187

Table 3. **The impact of neighborhood size** – larger neighborhoods lead to increased computational cost, and we find that 8 neighbors gives the best balance of cost and quality.

Num. of hidden layers in $\mathcal{A}$	CD (10^{-2}) ↓	F-score ↑	Latency (s) ↓
2	0.257	99.33	152
4	0.256	99.32	166

Table 4. **Impact of capacity of $\mathcal{A}$** – we find that increasing the number of layers in $\mathcal{A}$ beyond 2 decreases time efficiency without substantially improving the reconstruction quality.

Num. of scales	KNN	Minkowski	Z-order	Hilbert
0	1.00	0.17	0.44	0.46
1	1.00	0.29	0.48	0.50
2	1.00	0.38	0.49	0.52
3	1.00	0.44	0.49	0.53

Table 5. **Recall rate of our Hilbert-curve based $\mathcal{N}$** – we find that the Hilbert curve consistently outperforms both the Z-order curve [32] and the one-ring neighborhood from Minkowski relative to the exact k-nearest neighbors.

trade-off between accuracy and time efficiency. We supplement Table 4 with an analysis across even more levels in the `Supplementary Material`.

Analysis of neighbors retrieved by $\mathcal{N}$ – Table 5. We now investigate the quality of the point neighborhoods retrieved by various possible implementations for $\mathcal{N}$. In particular, we are interested to experimentally study whether our serialization indeed preserves locality. To quantify this, we treat the neighborhood retrieved with KNN as the ground-truth. We report the recall rate of a local neighborhood by comparing it with this ground truth (we ignore the precision rate because we remove false positives with a distance threshold). We also report the recall rate of the one-ring neighborhood retrieved in Minkowski [11]. We show that the recall rate of our Hilbert $\mathcal{N}$ is the best across variants, and across all scales.

The impact of sampling pattern – Table 6. We report the impact of sampling pattern on performance by evaluating models on ScanNet point clouds that are uniformly or non-uniformly sampled. To non-uniformly sample the ScanNet point clouds, we first partitioned the scene into eight blocks and randomly sampled a different number of points from each block. The number of samples followed an arithmetic sequence with a common difference of 200. Finally, we padded the last block to ensure that the total number of points remained 10K.

Methods	Uniform		Non-Uniform	
NKSR [22]	0.246	480s	0.273	668s
Ours (Minkowski) [11]	0.301	97s	0.349	94s
Ours (Minkowski) [11] (w/ KNN)	0.254	145s	0.294	155s
Ours (w/ serialization)	0.257	152s	0.296	145s
Ours (w/ KNN)	**0.243**	**151s**	**0.273**	**142s**

Table 6. **The impact of sampling** – we evaluate uniform vs non-uniform sampling on ScanNet. We find that our method achieves the best accuracy (in terms of CD (10^{-2})) and good time efficiency compared to NKSR [22] for both sampling types.

We show that our method achieves better robustness to non-uniform sampling than the baselines, highlighting the importance of avoiding quantization of the point cloud for high quality surface reconstruction.

5. Conclusions

Voxel-based data structures dominate surface reconstruction for large scale point clouds due to their superior time efficiency. Despite its dominance, voxelization, which collapses multiple points within a cube into a single voxel feature, causes significant information loss, leading to degraded performance in the task of surface reconstruction. In this work, we propose an efficient *point-based* framework that allows us to use the original point cloud, hence not incurring in any information loss. The key idea is point cloud serialization, inspired by recent efficient point transformers [43, 48], coupled with a simple point-cloud architecture for feature aggregation. We show that our method outperforms prior SOTA in both accuracy and time efficiency, enabling point-based representation for surface reconstruction from large scale point clouds.

Future works. We would like to explore better strategies for combining the optimal KNN and fast approximate neighborhood function, as this strategy should ideally be adaptive to point cloud size. We would also be interested in exploring generative modeling of large scale point clouds using these methods.

Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant, NSERC Collaborative Research and Development Grant, Google DeepMind, Digital Research Alliance of Canada, the Advanced Research Computing at the University of British Columbia, and the SFU Visual Computing Research Chair program. Shaobo Xia was supported by National Natural Science Foundation of China under Grant 42201481. We would also like to thank Jiahui Huang for the valuable discussion and feedback.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*. PMLR, 2018. 2

[2] Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. Digs : Divergence guided shape implicit neural representation for unoriented point clouds, 2023. 2

[3] Wang Bing, Yu Zhengdi, Bo Yang, Qin Jie, Breckon Toby, Shao Ling, Niki Trigoni, and Andrew Markham. Rangeudf: Semantic surface reconstruction from 3d point clouds. *arXiv preprint arXiv:2204.09138*, 2022. 2, 4, 5, 6, 7, 3

[4] Alexandre Boulch and Renaud Marlet. Poco: Point convolution for surface reconstruction. In *CVPR*, 2022. 2, 4, 6, 3

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 4, 2

[6] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2

[7] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *PAMI*, 2024. 1

[8] Wanli Chen, Xinge Zhu, Guojin Chen, and Bei Yu. Efficient point cloud analysis using hilbert curve. In *ECCV*, 2022. 2

[9] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 2

[10] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *NIPS*, 2020. 2, 4, 5, 6, 7, 3

[11] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 1, 2, 5, 6, 7, 8, 3

[12] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes, 2017. 2, 4, 5, 6, 7, 1, 3

[13] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator, 2017. 1, 2, 5, 6, 3, 4

[14] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J. Mitra, and Michael Wimmer. Points2Surf: Learning implicit surfaces from point clouds. In *ECCV*, 2020. 2

[15] Rao Fu, Kai Hormann, and Pierre Alliez. Lfs-aware surface reconstruction from unoriented 3d point clouds. *arXiv preprint arXiv:2403.13924*, 2024. 2

[16] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 2

[17] Jianwei Guo, Haobo Qin, Yinchang Zhou, Xin Chen, Liangliang Nan, and Hui Huang. Fast building instance proxy reconstruction for large urban scenes. *PAMI*, 2024. 1

[18] David Hilbert and David Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes: Nebst Einer Lebensgeschichte*, 1935. 2, 4

[19] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020. 2

[20] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *International Conference on 3D Vision (3DV)*, 2016. 2, 5, 7, 3

[21] Jiahui Huang, Hao-Xiang Chen, and Shi-Min Hu. A neural galerkin solver for accurate surface reconstruction. *TOG*, 2022. 2

[22] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural kernel surface reconstruction. In *CVPR*, 2023. 1, 2, 3, 4, 5, 6, 7, 8

[23] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *TOG*, 2013. 2, 5, 6, 3

[24] Chamin Hewa Koneputugodage, Yizhak Ben-Shabat, Dylan Campbell, and Stephen Gould. Small steps and level sets: Fitting neural surface models with point guidance. In *CVPR*, 2024. 2

[25] Shengtao Li, Ge Gao, Yudong Liu, Yu-Shen Liu, and Ming Gu. Gridformer: Point-grid transformer for surface reconstruction. In *AAAI*, 2024. 2

[26] Tianyang Li, Xin Wen, Yu-Shen Liu, Hua Su, and Zhizhong Han. Learning deep implicit functions for 3d shapes with dynamic code clouds. In *CVPR*, 2022. 2

[27] Zhijian Liu, Xinyu Yang, Haotian Tang, Shang Yang, and Song Han. Flatformer: Flattened window attention for efficient point cloud transformer. In *CVPR*, 2023. 3

[28] Xiaoxiao Long, Cheng Lin, Lingjie Liu, Yuan Liu, Peng Wang, Christian Theobalt, Taku Komura, and Wenping Wang. Neuraludf: Learning unsigned distance fields for multi-view reconstruction of surfaces with arbitrary topologies. In *CVPR*, 2023. 1, 2

[29] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*. 1998. 2

[30] Baorui Ma, Junsheng Zhou, Yu-Shen Liu, and Zhizhong Han. Towards better gradient consistency for neural signed distance functions via level set alignment. In *CVPR*, 2023. 2

[31] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 2

[32] Guy M Morton. A computer oriented geodetic data base and a new technique in file sequencing. 1966. 3, 8

[33] Liangliang Nan and Peter Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *ICCV*, 2017. 2

[34] Amine Ouasfi and Adnane Boukhayma. Unsupervised occupancy learning from sparse point cloud. In *CVPR*, 2024. 2

[35] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2

[36] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks, 2020. 2, 4, 5, 6, 7, 1, 3

[37] Scott Schaefer and Joe Warren. Dual marching cubes: Primal contouring of dual grids. In *Proc. Pacific Graphics*, 2004. 5

[38] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *CVPR*, 2021. 2

[39] Jiapeng Tang, Jiabao Lei, Dan Xu, Feiying Ma, Kui Jia, and Lei Zhang. Sa-convonet: Sign-agnostic optimization of convolutional occupancy networks, 2021. 2, 4, 5, 6, 7, 3

[40] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *CVPR*, 2017. 2

[41] Wenwen Tong, Chonghao Sima, Tai Wang, Li Chen, Silei Wu, Hanming Deng, Yi Gu, Lewei Lu, Ping Luo, Dahua Lin, et al. Scene as occupancy. In *ICCV*, 2023. 1

[42] Jun Wang and Jie Shan. Space filling curve based point clouds index. In *Proceedings of the 8th International Conference on GeoComputation*, 2005. 2

[43] Peng-Shuai Wang. Octformer: Octree-based transformers for 3d point clouds. *TOG*, 2023. 1, 2, 3, 4, 8

[44] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *TOG*, 2017. 2

[45] Zhen Wang, Shijie Zhou, Jeong Joon Park, Despoina Paschalidou, Suya You, Gordon Wetzstein, Leonidas Guibas, and Achuta Kadambi. Alto: Alternating latent topologies for implicit 3d reconstruction. In *CVPR*, 2023. 2

[46] Francis Williams, Matthew Trager, Joan Bruna, and Denis Zorin. Neural splines: Fitting 3d surfaces with infinitely-wide neural networks. In *CVPR*, 2021. 2

[47] Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural fields as learnable kernels for 3d reconstruction. In *CVPR*, 2022. 2

[48] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *CVPR*, 2024. 2, 3, 4, 5, 8

[49] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017. 6, 3

[50] Biao Zhang, Matthias Nießner, and Peter Wonka. 3dilg: Irregular latent grids for 3d generative modeling. In *NIPS*, 2022. 2

[51] Guowen Zhang, Lue Fan, Chenhang He, Zhen Lei, Zhaoxiang Zhang, and Lei Zhang. Voxel mamba: Group-free state space models for point cloud based 3d object detection. *arXiv preprint arXiv:2406.10700*, 2024. 3

[52] Xingguang Zhong, Yue Pan, Cyrill Stachniss, and Jens Behley. 3d lidar mapping in dynamic environments using a 4d implicit neural representation. In *CVPR*, 2024. 2

[53] Zhiyun Zhuang, Zhiyang Zhi, Ting Han, Yiping Chen, Jun Chen, Cheng Wang, Ming Cheng, Xinchang Zhang, Nannan Qin, and Lingfei Ma. A survey of point cloud completion. *IEEE JSTARS*, 2024. 1