Neural Kernel Surface Reconstruction

Jiahui Huang[1] Zan Gojcic[1] Matan Atzmon[1] Or Litany[1] Sanja Fidler[1,2,3] Francis Williams[1]

[1]NVIDIA [2]University of Toronto [3]Vector Institute

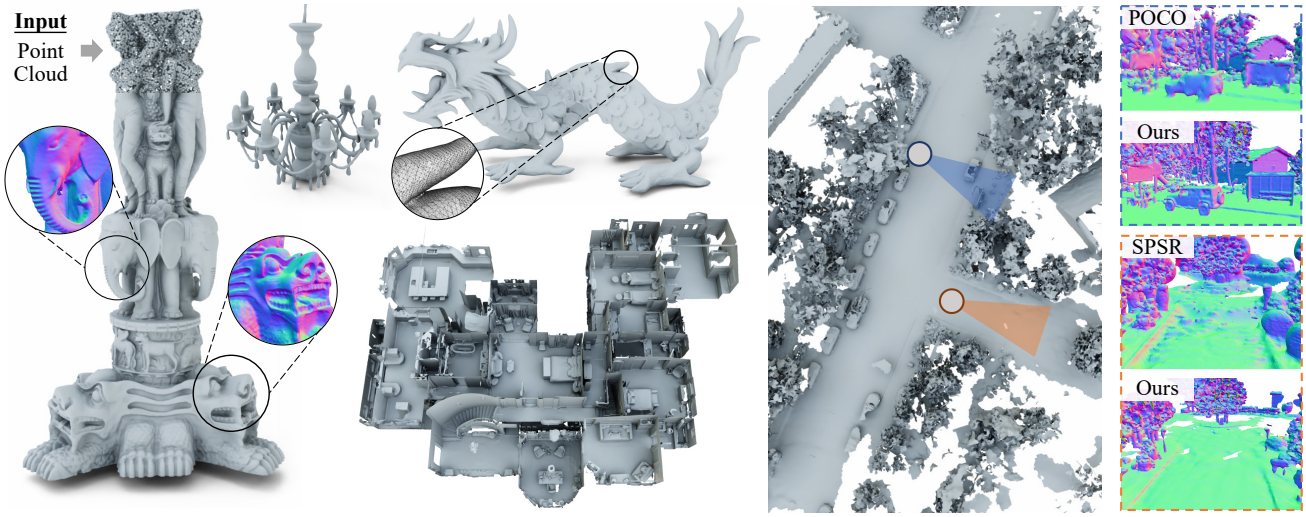


Figure 1: We present Neural Kernel Surface Reconstruction (**NKSR**) for recovering a 3D surface from an input point cloud. Trained directly from dense points, our method reaches state-of-the-art reconstruction quality and scalability. NKSR is also highly generalizable: All the meshes in this figure are reconstructed using a single trained model.

Abstract

We present a novel method for reconstructing a 3D implicit surface from a large-scale, sparse, and noisy point cloud. Our approach builds upon the recently introduced Neural Kernel Fields (NKF) [58] representation. It enjoys similar generalization capabilities to NKF, while simultaneously addressing its main limitations: (a) We can scale to large scenes through compactly supported kernel functions, which enable the use of memory-efficient sparse linear solvers. (b) We are robust to noise, through a gradient fitting solve. (c) We minimize training requirements, enabling us to learn from any dataset of dense oriented points, and even mix training data consisting of objects and scenes at different scales. Our method is capable of reconstructing millions of points in a few seconds, and handling very large scenes in an out-of-core fashion. We achieve state-of-the-art results on reconstruction benchmarks consisting of single objects (ShapeNet [5], ABC [33]), indoor scenes (ScanNet [11], Matterport3D [4]), and outdoor scenes (CARLA [16], Waymo [49]).

1. Introduction

The goal of 3D reconstruction is to recover geometry from partial measurements of a shape. In this work, we aim to map a sparse set of oriented points sampled from the surface of a shape to a 3D implicit field. This is a challenging inverse problem since point clouds acquired from real-world sensors are often very large (millions or billions of points), vary in sampling density, and are corrupted with sensor noise. Furthermore, since surfaces are continuous but points are discrete, there are many valid solutions which can explain a given input. To address these issues, past approaches aim to recover surfaces that agree with the input points while satisfying some prior everywhere else in space. Classical methods use an explicit prior (e.g. smoothness), while more recent learning-based approaches promote a likely reconstruction under a data-driven prior.

There are, however, key limitations to both types of techniques that inhibit their application in practical situations. Since classical methods are fast, scalable, and able to handle diverse inputs, they have become an industry standard (e.g. [32, 61]). Yet, they suffer from quality issues in the presence of high noise or sparse inputs, often failing to reconstruct even simple geometry such as a ground plane (see the ground in Fig. 1). On the other hand, learning-based approaches were shown to handle large noise [42], and sparse inputs [39, 3], yet they often struggle to generalize to out-of-distribution shapes and sampling densities as was highlighted in [58]. These generalization issues can be attributed to the

fact that current learning-based methods struggle to exploit large and diverse amounts of data for training. One cause of this is that a single forward pass can take minutes for even moderately sized inputs (*e.g.* [3]), limiting training to collections consisting of small point clouds. Furthermore, many existing methods rely on a preprocessing step to extract supervision in the form of occupancy or signed distance function [43, 38, 40, 3, 58]. In practice, this preprocessing step hinders the ability to easily use diverse datasets for training since most shape datasets (including synthetic ones such as the popular ShapeNet [5]) consist of non-watertight shapes, open surfaces, or contain ghost geometry from which extracting supervision is hard.

Recently, [58] proposed Neural Kernel Fields (NKF), a new paradigm to address the problem of generalization in 3D reconstruction. NKF learns a data-dependent kernel, and predicts a continuous occupancy field as a linear combination of this kernel supported on the input points. The key insights of NKF are that a kernel explicitly encodes inductive bias, and that solving a kernel linear interpolation problem at test time always produces solutions that adhere to the inputs. Thus, by training on diverse shapes, NKF can learn a good inductive bias for the general 3D reconstruction problem rather than for a specific dataset. While NKF achieves impressive generalization results, it suffers from two major limitations that restrict its practical application. First, since it uses a globally supported kernel, it requires solving a dense linear system and cannot reconstruct inputs with more than ten thousand input points. Second, it degrades poorly in the presence of noise due to its interpolation of exact positional occupancy constraints.

In this work, we build upon the excellent generalization capability of NKF and tackle its main limitations to achieve a *practical* learning-based reconstruction method that is scalable, fast, and robust to noise. Like NKF, our work leverages the idea of a learned kernel for generalization, but we (1) develop a novel, gradient-based kernel formulation which is robust to noise, and (2) use an explicit voxel hierarchy structure and compactly supported kernels to make our interpolation problem sparse, multi-scale, and capable of handling large inputs while still producing high fidelity outputs. The result is a learning-based yet out-of-the-box reconstruction method that can be applied to point clouds in the wild. In particular, it enjoys all of the following properties:

- It can generalize to out-of-distribution inputs, producing high-fidelity reconstructions, even in the presence of sparsity and noise.
- It can be trained on the union of diverse datasets while only requiring dense oriented points as supervision, unlocking a new level of training data scale.
- It can reconstruct point clouds consisting of millions of points in seconds, and scale to extremely large inputs in an out-of-core fashion.

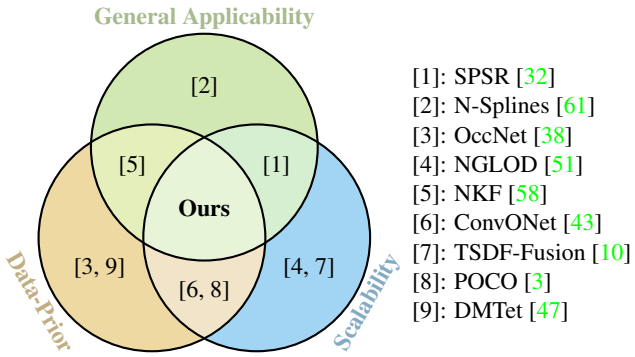


Figure 2: **Comparison to related works.**

We illustrate other methods in the context of these points visually in Fig. 2.

2. Related Work

We now give a brief overview of prior works that are relevant to our approach. Learned kernels were investigated in [62, 29, 41] for tasks such as few-shot transfer learning and classification of images. In the context of 3D reconstruction, Chu *et al*. [9] encoded the inductive bias intrinsically in a 3D CNN structure without training data. NKF [58] proposed a novel *data-dependent* kernel, which improved upon the non-learned kernel method derived from infinitely wide ReLU networks in Neural Splines [61]. Comparably, we use a data-dependent kernel but restrict its spatial support to increase computational efficiency and use a gradient-based fitting formulation to increase noise robustness. Mapping 3D points to a feature grid via a convolutional architecture was proposed in ConvONet [43] and CIRCLE [6] for predicting an occupancy field. POCO [3] improved the quality and performance of ConvONet by using a transformer architecture instead of convolutions. Both methods, however, take a long time to reconstruct even a small scene. Differently, our feature mapping is made efficient through a hierarchical sparse data structure. Non-dense data structures were studied in ASR [54] and DOGNN [56] which proposed octree-based convolutional architectures for reconstructing large scenes. Generalization to novel scenes was addressed by LIG [21, 31] using local patches which have smaller variability, but are very sensitive to the choice of patch size and relies on test-time optimization with unknown convergence properties. Similarly, our kernel weights are fitted to the scene at prediction but the fitting is done via a linear solver in a form of meta-learning [48]. Shape as Points [42] learns to upsample the input points followed by differentiable Poisson reconstruction, and this idea is further extended by NGSolver [30] to incorporate learnable basis functions. However, the representation power of the surface is still bounded by the chosen family of basis functions where non-trivial integrations have to be applied. Beyond meth-

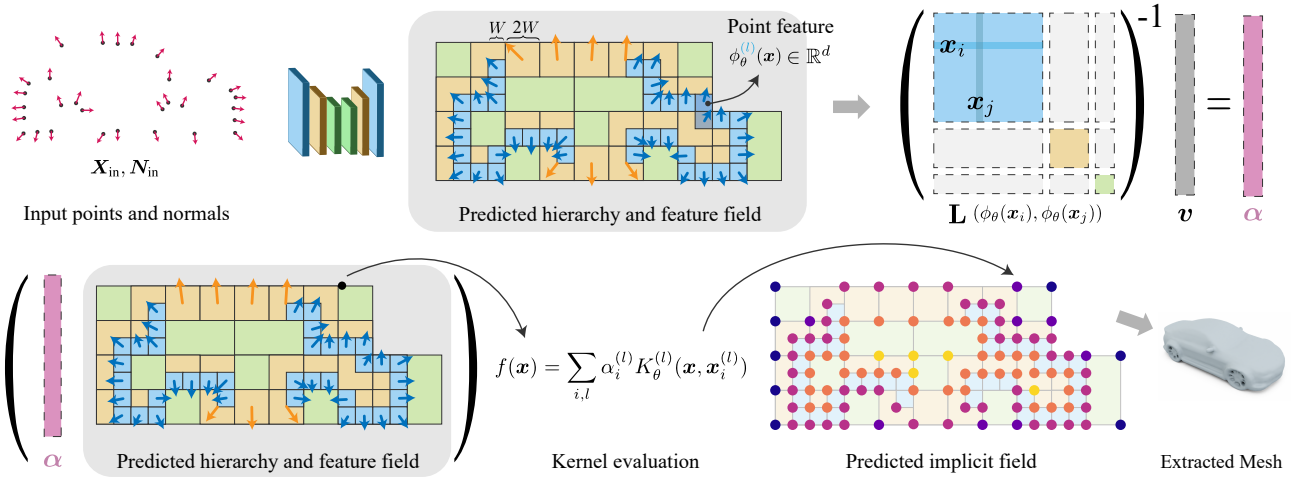


Figure 3: **Pipeline.** Our method accepts an oriented point cloud and predicts a sparse hierarchy of voxel grids containing features as well as normals in each voxel. We then construct a sparse linear system and solve for a set of per-voxel coefficients $\boldsymbol{\alpha}$. The linear system corresponds to the gram matrix arising from a kernel which depends on the predicted features, illustrated as $\mathbf{L}$ and $\boldsymbol{v}$ above but mathematically defined in Eq (4). To extract the predicted surface, we evaluate the function values at the voxel corners using a linear combination of the learned kernel basis functions, followed by dual marching cubes.

ods based on implicit surfaces, other shape reconstruction techniques exist which leverage different output representations. These representations include dense point clouds [46, 37, 69, 44, 45, 68, 50, 64, 65, 18, 34], polygonal meshes [28, 7, 20, 27, 23, 59, 12, 35, 25, 47], manifold atlases [60, 15, 24, 19, 2], and voxel grids [8, 52, 26, 63, 53, 22]. While our method uses a neural field for reconstruction, past work has used neural fields to perform a variety of point cloud tasks such as shape compression [51, 61], voxel grid upsampling [43, 38], reconstruction from rotated inputs [14, 1] and articulated poses [13, 66].

3. Method

Our method predicts a 3D surface given a point cloud with normals. We encode this predicted surface as the zero level set of a Neural Kernel Field, *i.e.* an implicit function represented as a weighted sum of learned, positive-definite basis functions which are conditioned on the input, and whose weights are computed using a linear optimization in the forward pass. These basis functions are supported on a sparse voxel hierarchy which we predict from the input point cloud using a sparse convolutional network, and depend on interpolated features at each voxel corner. In the following sections, we describe the key steps of our method, and summarize it pictorially in Fig. 3. We additionally provide rigorous derivations for each step in the Appendix.

3.1. Predicting a 3D Shape from Points

Given points and normals, the forward pass of our model predicts an implicit surface as a weighted sum of learned kernels in two steps: First, we feed the input to a sparse convolutional network that predicts a voxel hierarchy with features at each corner (Fig. 4). These features define a collection of learned basis functions, which are centered at each voxel in the hierarchy. Second, we find a set of weights for these basis functions by solving a linear system that encourages the predicted implicit field to have a zero value near the input points, and to have gradients which agree with the input normals. Optionally, we can also predict a geometric mask, which outputs where in space to extract the final surface, trimming away spurious geometry.

Predicting a Sparse Voxel Hierarchy. Given input points $\boldsymbol{X}_{\text{in}} = \{\boldsymbol{x}_j^{\text{in}} \in \mathbb{R}^3\}_{j=1}^{n_{\text{in}}}$, input normals $\boldsymbol{N}_{\text{in}} = \{\boldsymbol{n}_j^{\text{in}} \in \mathbb{R}^3\}_{j=1}^{n_{\text{in}}}$, and a voxel size W, we first predict a hierarchy of L voxel grids using a convolutional backbone digesting the point cloud with concatenated normal $[\boldsymbol{x}_j^{\text{in}}, \boldsymbol{n}_j^{\text{in}}] \in \mathbb{R}^6$ for each point. Each of the predicted voxel grid has $n^{(1)}, \ldots, n^{(L)}$ voxels with widths $W, 2W, \ldots 2^L W$ respectively and any voxel with center $\boldsymbol{x}_i^{(l-1)}$ at level $l-1$ is contained within some voxel with center $\boldsymbol{x}_j^{(l)}$ at level l. The design of such a backbone network is inspired by [57] and is described in detail in the Appendix. We additionally predict features $\boldsymbol{z}_i^{(l)} \in \mathbb{R}^d$ and normals $\boldsymbol{n}_i^{(l)} \in \mathbb{R}^3$ for each voxel in the hierarchy. The features $\boldsymbol{z}_i^{(l)}$ are employed to predict a feature field $\phi_\theta^{(l)}(\boldsymbol{x} | \boldsymbol{X}_{\text{in}}, \boldsymbol{N}_{\text{in}})$ which lifts the coordinates $\boldsymbol{x} \in \mathbb{R}^3$ to d-dimenensional vectors via Bézier interpolation followed by an MLP. Fig. 4 shows a 2D illustration of our predicted hierarchy and features.

Sparse Neural Kernel Field Hierarchy. We encode our reconstructed shape as the zero level set of a 3D implicit field $f_\theta : \mathbb{R}^3 \to \mathbb{R}$ defined as a hierarchical *Neural Kernel*

Field, *i.e.*, a weighted combination of *positive definite kernels* which are conditioned on the inputs and centered at the midpoints $\boldsymbol{x}_i^{(l)} \in \mathbb{R}^3$ of voxels in the predicted hierarchy:

$$f_\theta(\boldsymbol{x}|\boldsymbol{X}_{\text{in}}, \boldsymbol{N}_{\text{in}}) = \sum_{i,l} \alpha_i^{(l)} K_\theta^{(l)}(\boldsymbol{x}, \boldsymbol{x}_i^{(l)}|\boldsymbol{X}_{\text{in}}, \boldsymbol{N}_{\text{in}}). \quad (1)$$

Here, $\alpha_i^{(l)} \in \mathbb{R}$ are scalar coefficients at the i^{th} voxel at level $l = 1, \dots L$ in the hierarchy, and $K_\theta^{(l)}$ is the predicted kernel for the l^{th} level defined as

$$\begin{aligned} K_\theta^{(l)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi_\theta^{(l)}(\boldsymbol{x}; \boldsymbol{X}_{\text{in}}, \boldsymbol{N}_{\text{in}}), \\ \phi_\theta^{(l)}(\boldsymbol{x}'; \boldsymbol{X}_{\text{in}}, \boldsymbol{N}_{\text{in}}) \rangle \cdot K_{\text{b}}^{(l)}(\boldsymbol{x}, \boldsymbol{x}'), \end{aligned} \quad (2)$$

where $\langle \cdot, \cdot \rangle$ is the dot product, $\phi_\theta^{(l)} : \mathbb{R}^3 \to \mathbb{R}^d$ is the feature field extracted from the hierarchy, and $K_b^{(l)} : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$ is the *Bézier Kernel*, which decays to zero in a one-voxel (at level-l) neighborhood around its origin (See Appendix for definition).

Computing a 3D implicit surface from points. Given our predicted voxel hierarchy, learned kernels $K_\theta^{(l)}$, and predicted normals $\boldsymbol{n}_j^{(l)}$, we compute an implicit surface by finding optimal coefficients $\boldsymbol{\alpha}^* = \{\{\alpha_i^{(l)}\}_{l=1}^{L}\}_{i=1}^{n^{(l)}}$ for the kernel field (1). We find these coefficients by exactly minimizing the following loss in the forward pass of our model (omitting the conditioning of f_θ on $\boldsymbol{X}_{\text{in}}, \boldsymbol{N}_{\text{in}}$ for brevity):

$$\begin{aligned} \boldsymbol{\alpha}^* = \underset{\alpha_i^{(l)}}{\text{argmin}} \sum_{l=1}^{L'} \sum_{i=1}^{n^{(l)}} \|\nabla_{\boldsymbol{x}} f_\theta(\boldsymbol{x}_i^{(l)}) - \boldsymbol{n}_i^{(l)}\|_2^2 + \\ \sum_{j=1}^{n_{\text{in}}} |f_\theta(\boldsymbol{x}_j^{\text{in}})|^2, \end{aligned} \quad (3)$$

where $L' \leq L$ is a hyper-parameter for the hierarchy. By minimizing this loss, we want our Neural Kernel Field f_θ to have a gradient which agrees with the normals at the voxel centers (hence regions around the surface), and to have a scalar value near zero at all the input points $\boldsymbol{X}_{\text{in}}$. Since f_θ is linear in the parameters $\alpha_i^{(l)}$, we find the optimal coefficients $\boldsymbol{\alpha}^*$ by solving the linear system:

$$(\mathbf{Q}^\top \mathbf{Q} + \mathbf{G}^\top \mathbf{G})\boldsymbol{\alpha} = \mathbf{Q}^\top \boldsymbol{n}, \quad (4)$$

where $\boldsymbol{n}$ are the predicted normal vectors $\boldsymbol{n}_i^{(l)}$ stacked into a single vector, $\boldsymbol{\alpha}$ is the vector of coefficients $\alpha_i^{(l)}$, and

$$\begin{aligned} \mathbf{G} &= \begin{bmatrix} \mathbf{G}^{(1)} & \dots & \mathbf{G}^{(L)} \end{bmatrix}, \\ \mathbf{Q} &= \begin{bmatrix} \mathbf{Q}^{(1)} & \dots & \mathbf{Q}^{(L)} \end{bmatrix}, \end{aligned} \quad (5)$$

$$\mathbf{G}_{i,j}^{(l)} = K_\theta(\boldsymbol{x}_i^{\text{in}}, \boldsymbol{x}_j^{(l)}), \quad \mathbf{Q}_{i,j}^{(l)} = \partial_{\boldsymbol{x}_i^{(l')}} K_\theta(\boldsymbol{x}_i^{(l')}, \boldsymbol{x}_j^{(l)}) \quad (6)$$

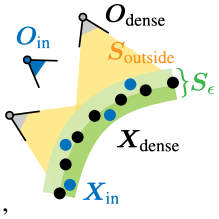
are the gram matrix and partial derivatives of the gram matrix at the voxel centers where normals are defined, respectively.

We remark that the linear system (4) is sparse due to modulation with the compactly supported $K_b^{(l)}$, and positive definite by construction since it is a Gram matrix. As a result, (4) can be solved very efficiently on a GPU.

Masking module. The predicted Neural Kernel Field f_θ is defined on the entire voxel hierarchy, however at coarse levels far from the surface, it may contain unwanted geometry. To discard such geometry away from the predicted surface, we add an additional branch to our backbone as $\varphi : \mathbb{R}^3 \to \{0, 1\}$ which determines if a point $\boldsymbol{x}$ should be trimmed ($\varphi(\boldsymbol{x}) = 0$) or kept ($\varphi(\boldsymbol{x}) = 1$). The branch originates from the immediate features of the backbone network and consists of a few linear layers with ReLU activations followed by a sigmoid. When we extract the final surface, we only consider vertices in regions where $\varphi(\boldsymbol{x}) > 0.5$.

3.2. Supervision

To train our model, we require pairs ($\boldsymbol{X}_{\text{in}} = \{\boldsymbol{x}_i \in \mathbb{R}^3\}, \boldsymbol{O}_{\text{in}} = \{\boldsymbol{o}_i \in \mathbb{R}^3\}$) and ($\boldsymbol{X}_{\text{dense}} = \{\boldsymbol{x}_j\}, \boldsymbol{O}_{\text{dense}} = \{\boldsymbol{o}_j\}$). Here $\boldsymbol{X}_{\text{in}}$ and $\boldsymbol{X}_{\text{dense}}$ are noisy input points and dense supervision points respectively, and $\boldsymbol{O}_{\text{in}}$ and $\boldsymbol{O}_{\text{dense}}$ are sensor origin for each input and supervision point (*i.e.* a position in 3D space from which each point was acquired). We additionally compute input and supervision normals $\boldsymbol{N}_{\text{in}}, \boldsymbol{N}_{\text{dense}}$ by fitting planes to points in a local neighborhood and orienting the normals to align with the directions from points to sensors. We remark that our training requirements impose no restrictions on the shapes being trained on. For example, one could use a single LiDAR frame as input and an accumulated LiDAR scan of a scene as supervision, alongside a noisy scan of a synthetic object as input and a dense noiseless scan of the same object as supervision.



In order to define the loss terms used to supervise our model, we first define two regions of space around the dense points $\boldsymbol{X}_{\text{dense}}$:

- $\boldsymbol{S}_\epsilon$: points which are ϵ distance or less from $\boldsymbol{X}_{\text{dense}}$ *i.e.* $\{\boldsymbol{x} | \min_{\boldsymbol{x}_j \in \boldsymbol{X}_{\text{dense}}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 < \epsilon\}$,
- $\boldsymbol{S}_{\text{outside}}$: points which lie within the region enclosing points in $\boldsymbol{X}_{\text{dense}}$ and their sensor origin in $\boldsymbol{O}_{\text{dense}}$.

Then, given a predicted Neural Kernel Field $f_\theta(\boldsymbol{x})$, we backpropagate through the following loss functions:

- $\mathcal{L}_{\text{surf}}(f) = \mathbb{E}_{\boldsymbol{x} \in \boldsymbol{X}_{\text{dense}}}[\|f(\boldsymbol{x})\|_1]$;
- $\mathcal{L}_{\text{tsdf}}(f) = \mathbb{E}_{\boldsymbol{x} \in \boldsymbol{S}_\epsilon}\left[\|f(\boldsymbol{x}) - \text{tsdf}(\boldsymbol{x}, \boldsymbol{X}_{\text{dense}})\|_1\right]$ where $\text{tsdf}(\boldsymbol{x}, \boldsymbol{X}_{\text{dense}})$ is the ground-truth truncated signed distance computed from $\boldsymbol{X}_{\text{dense}}$ using nearest neighbors;

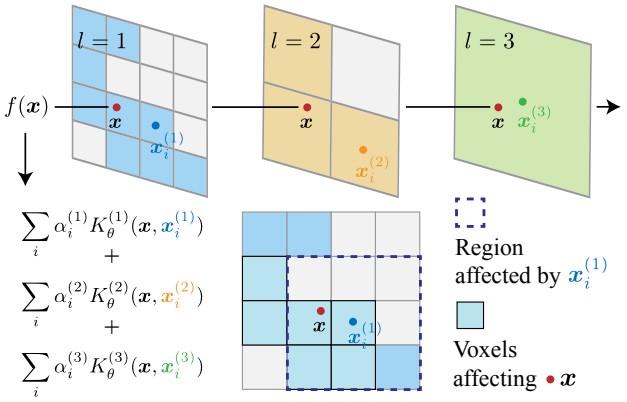


Figure 4: **Our implicit field** $f(\boldsymbol{x})$ is represented as a sum of kernel basis functions on a sparse voxel hierarchy. Each voxel with center $\boldsymbol{x}_i^{(l)}$ contributes one kernel basis function $K_\theta^{(i)}(\boldsymbol{x}, \boldsymbol{x}_i^{(l)})$ with support in the one-ring around $\boldsymbol{x}_i^{(l)}$.

- $\mathcal{L}_{\text{normal}}(f) = \mathbb{E}_{\boldsymbol{n} \in \boldsymbol{N}_{\text{in}}} \left[1 - \left\langle \frac{\nabla_{\boldsymbol{x}} f(\boldsymbol{x})}{\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\|_2}, \boldsymbol{n} \right\rangle \right]$;
- $\mathcal{L}_{\text{outside}}(f) = \mathbb{E}_{\boldsymbol{x} \in \boldsymbol{S}_{\text{outside}}} e^{-\beta \|f(\boldsymbol{x})\|_1}$, where $\beta = 0.1$;
- $\mathcal{L}_{\text{min-surf}}(f) = \mathbb{E}_{\boldsymbol{x} \in \boldsymbol{S}_\epsilon} \left[\frac{\eta \pi^{-1}}{\eta^2 + f(\boldsymbol{x})^2} \right]$, where $\eta = 0.5$.

Here $\mathcal{L}_{\text{surf}}$ ensures that the implicit function is zero near the ground truth surface, $\mathcal{L}_{\text{tsdf}}$ ensures that the implicit field undergoes a sign change near the surface, $\mathcal{L}_{\text{normal}}$ ensures the gradient of the predicted implicit agrees with the dense normals, $\mathcal{L}_{\text{outside}}$ ensures there is no geometry far away from the surface, and $\mathcal{L}_{\text{min-surf}}$ acts as a regularizer encouraging the predicted implicit surface to have minimal area [67].

We additionally compute structure prediction and masking losses which we describe in the Appendix. We train our model in an end-to-end fashion using gradient descent by back-propagating through the sum of all these loss functions.

4. Experiments

Overview. In this section we demonstrate that NKSR fulfills the three main desired properties of practical surface reconstruction method as analyzed in § 1: (1) *Accuracy* (§ 4.1), by training and testing on object-level datasets [5, 33, 70] with varying noise settings. (2) *Scalability* (§ 4.2), by evaluating on large-scale outdoor driving dataset [16]. (3) *Generalizability* (§ 4.3), where we train on object-level/outdoor datasets and test on room-level datasets [11, 4] as well as scans with very low densities. Notably, to encourage the practical usage of NKSR, we present a *kitchen-sink-model* (denoted as 'Ours - ✓') trained on the union of various datasets [5, 33, 16, 4] and report its performance whenever applicable. While this model slightly underperforms domain-specific models, it still outperforms most baseline methods and can be used on a wide variety of inputs as shown in Fig. 1 and Fig. 9. We hope the kitchen-sink-model enables end-users to use NKSR in a plug-and-play manner.

[1]This blue checkmark model will be made publicly available, for free.

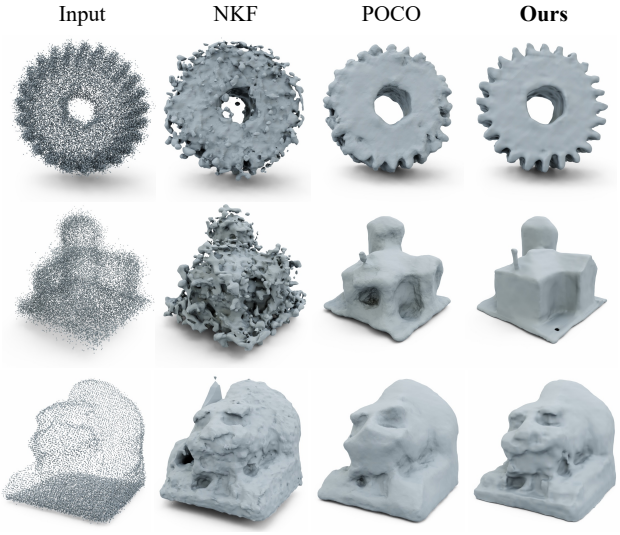


Figure 5: **ABC/Thingi10K [33, 70] visualization.**

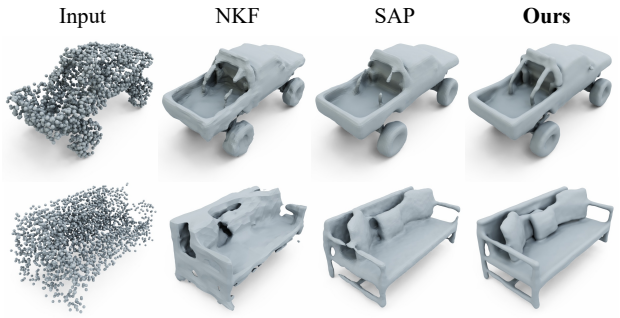


Figure 6: **ShapeNet [5] visualization.** The two shapes are with $\sigma = 0.005$ and $\sigma = 0.025$ Gaussian noise respectively.

Implementation Details. Our pipeline is fully accelerated using PyTorch and CUDA. The operations on our sparse hierarchy including convolution, neighborhood querying and interpolations are based on a customized tree structure that is highly efficient and scalable. Our sparse linear solver uses a Jacobi-preconditioned conjugate gradient method and works jointly with the sparse hierarchy for fast inference. Unless otherwise specified, our experiments are run on a single V100 GPU with 8 CPU cores. Hyperparameter details are given in the Appendix.

4.1. Accuracy: Object-level Reconstruction

Settings. We follow two common evaluation settings from the literature. One is the manifold ShapeNet [5] dataset prepared by [38]. The dataset contains man-made geome-

Table 1: **ABC/Thingi10K [33, 70] comparison**. d_C is multiplied by 10^3. σ is the Gaussian noise added to the sensor depth and L is the largest box length. 10 scans are used to accumulate the point cloud unless specified.

	ABC (100 shapes)						Thingi10K (100 shapes)									
	$\sigma = 0$		$\sigma \in [0, 0.05L]$		$\sigma = 0.05L$		$\sigma = 0$		$\sigma = 0.01L$		$\sigma = 0.05L$		5 scans		30 scans	
	$d_C\downarrow$	F-S.↑	$d_C\downarrow$	F-S.↑	$d_C\downarrow$	F-S.↑	$d_C\downarrow$	F-S.↑	$d_C\downarrow$	F-S.↑	$d_C\downarrow$	F-S.↑	$d_C\downarrow$	F-S.↑	$d_C\downarrow$	F-S.↑
SPSR [32]	7.02	87.5	11.2	72.8	18.8	47.9	4.23	91.9	5.44	90.3	16.5	52.5	12.4	77.6	3.07	96.7
POCO [3]	5.34	88.3	8.23	75.7	12.0	58.9	4.42	92.5	5.10	89.7	11.2	58.8	6.95	84.4	3.69	95.0
SAP [42]	6.83	85.0	8.00	79.5	10.4	68.7	4.30	92.7	4.54	91.8	7.82	**74.7**	6.73	84.7	3.95	93.8
NKF [58]	6.10	88.1	13.8	62.3	24.0	35.1	3.48	94.2	4.78	90.8	24.7	34.3	7.05	84.8	4.36	93.2
NGSolver [30]	3.92	92.7	6.35	83.1	9.68	66.4	2.96	95.9	3.51	95.0	8.70	69.1	5.65	89.2	2.80	97.1
Ours	**3.68**	**93.2**	**6.00**	**85.4**	**8.70**	**73.2**	**2.36**	**97.3**	**3.19**	**95.9**	**7.66**	**74.7**	**5.10**	**89.9**	**2.48**	**98.0**
Ours - ✓	4.10	92.2	6.44	83.6	9.97	68.1	2.92	96.3	3.34	95.6	8.55	72.7	5.60	89.1	2.54	97.7

Table 2: **ShapeNet [5] comparison**. 'N.' denotes whether normals $\boldsymbol{N}_{\text{in}}$ are used as input. d_C is multiplied by 10^3.

		1000 Pts. $\sigma = 0.0$		3000 Pts. $\sigma = 0.005$		3000 Pts. $\sigma = 0.025$	
	N.	$d_C\downarrow$	IoU↑	$d_C\downarrow$	IoU↑	$d_C\downarrow$	IoU↑
ConvONet [43]	-	6.07	82.3	4.35	88.0	7.31	78.7
IMLSNet [36]	-	3.15	91.2	3.03	91.3	6.58	76.0
SAP [42]	-	3.44	90.8	3.30	91.1	5.34	**82.9**
POCO [3]	-	3.03	92.7	2.93	92.2	5.82	81.7
NGSolver [30]	-	2.91	91.9	2.90	91.8	5.06	82.8
Ours	-	**2.64**	**93.4**	**2.71**	**92.6**	**4.96**	**82.9**
SPSR [32]	✓	6.26	81.4	3.84	88.5	10.7	66.8
SAP [42]	✓	3.21	92.1	3.16	92.3	4.44	87.1
NKF [58]	✓	2.65	94.7	3.17	91.2	11.7	67.0
NGSolver [30]	✓	2.47	95.0	2.51	94.1	3.93	87.5
Ours	✓	**2.34**	**95.6**	**2.45**	**94.3**	**3.87**	**87.6**

tries from 13 categories, with >30K shapes for training and >8K shapes for testing. Gaussian noise of different standard deviations (denoted as σ) is added to the randomly-subsampled points from the ground truth as input. As many existing learning-based baselines do not need point normals $\boldsymbol{N}_{\text{in}}$ as input, we present a variant of our model that does not take $\boldsymbol{N}_{\text{in}}$ as extra input channels for a fair comparison. The other setting is from [17] where a random subset of ∼5K shapes from ABC [33] is picked for training and testing, and an additional 100 shapes from Thingi10K [70] is used for testing generalization. The input is acquired by simulating ToF sensors with different levels of noise and densities. For the metrics we use the standard Chamfer distance (d_C), F-score (F-S.), normal consistency (N.C.), and intersection-over-union ratio (IoU) as benchmarks.

Results. The comparisons are quantitatively shown in Tab. 1 and Tab. 2, and selectively visualized in Fig. 5 and Fig. 6. Our model reaches state-of-the-art performance on all the datasets. Our baseline, NKF, works well on the noise-free setting but inelegantly degrades with higher noise due to its over-reliance on the raw input normals. On the other hand, SAP and NGSolver are more robust under noise, but the fitting tightness as reflected by d_C is higher than ours due to the lack of representation power. Our performance gain is mainly based on the gradient-based energy fitting formulation backed up by the natural inductive biases emerging from the learned kernel.

Table 3: **CARLA [16] comparison.** d_C is the average of Acc. and Comp. (Unit is cm. The smaller the better.)

	Original			Novel			Time
	Acc.	Comp.	F-S.↑	Acc.	Comp.	F-S.↑	(sec.)
TSDF-Fusion [55]	8.2	8.0	80.2	8.6	6.6	80.7	**0.5**
POCO [3]	10.5	3.6	90.1	9.1	2.9	92.4	420
SPSR [32]	10.3	16.4	86.5	9.9	12.8	88.3	30
Ours	**5.6**	**2.2**	**93.9**	**3.6**	**2.1**	**96.0**	2.6
Ours - ✓	4.1	3.0	94.0	3.6	2.4	96.0	2.6

4.2. Scalability: Outdoor Driving Scenes

Dataset. The applicability of NKSR to large-scale datasets is demonstrated using the synthetic CARLA [16] dataset due to the lack of large-scale real-world datasets with ground-truth geometries. To generate such a dataset, we manually pick 3 towns and simulate 10 random drives using the engine. We call these drives the 'Original' subset. An additional town along with its 3 drives is used only during evaluation to test generalization, which we denote as the 'Novel' subset. For $(\boldsymbol{X}_{\text{in}}, \boldsymbol{O}_{\text{in}})$, we use a sparse 32-beam LiDAR with 0-5cm ray distance noise and 0-3° pose noise. For $(\boldsymbol{X}_{\text{dense}}, \boldsymbol{O}_{\text{dense}})$, we employ a noise-free highly-dense 256-beam LiDAR for ground-truth supervision. The accumulated LiDAR points are cropped into $51.2 \times 51.2\text{m}^2$ chunks for the ease of benchmarking. Please find more details and visualizations in the Appendix.

Results. We compare our results to TSDF-Fusion, SPSR and the learning-based POCO. While for the latter two baselines we use the same voxel sizes $W = 10$cm as ours, for TSDF-Fusion we find it necessary to increase W to 30cm to reach decent surface completeness. The results are shown in Tab. 3 and visualized in Fig. 7. We compute single-sided

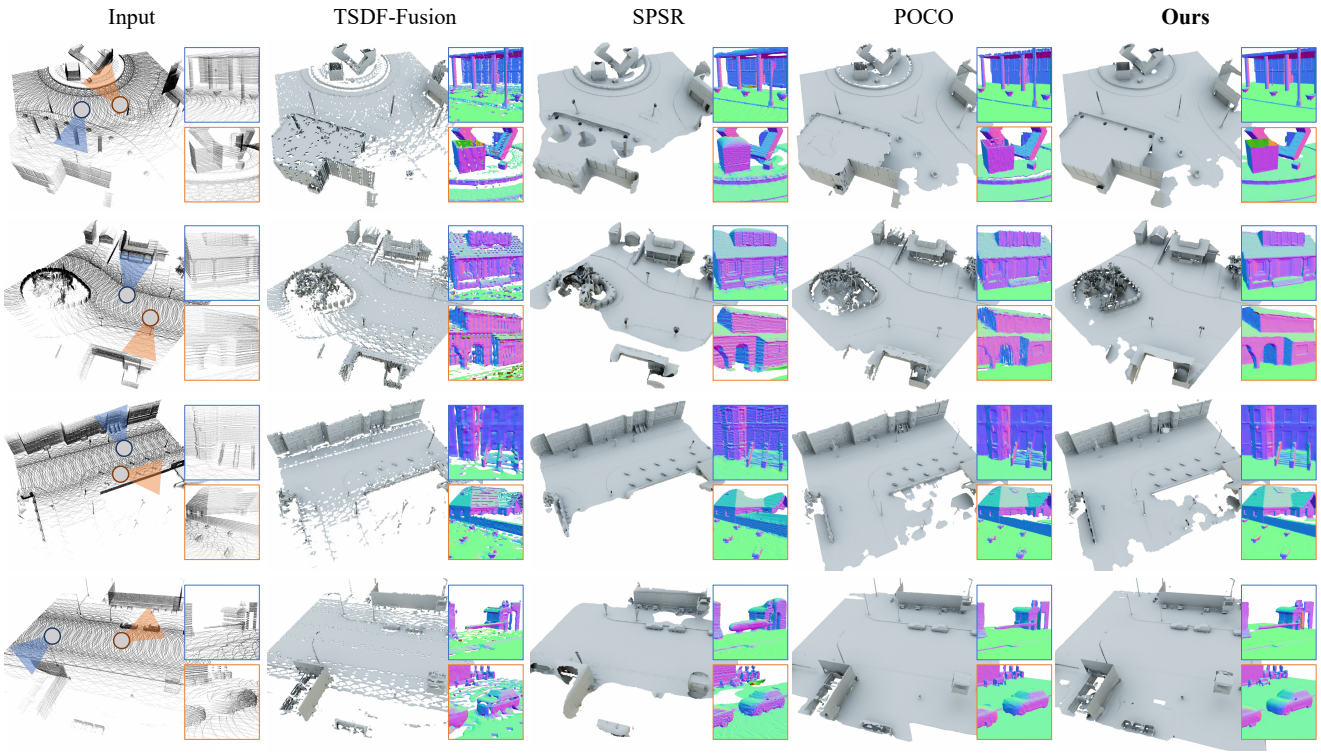


Figure 7: **CARLA [16] visualization.** The insets show zoom-ins captured by the cameras shown in the leftmost column with the corresponding color. The upper 2 rows are from the 'Novel' subset and the lower 2 rows are from the 'Original' subset.

Table 4: **Room-level dataset [11, 4] comparison**. d_C is multiplied by 10^3.

	Training Set	ScanNet			Matterport3D		
		$d_C\downarrow$	F-S.↑	N.C.↑	$d_C\downarrow$	F-S.↑	N.C.↑
SPSR [32]	-	7.04	84.3	87.2	10.4	87.0	92.3
LIG [31]	Shape Net	6.19	83.8	83.7	5.13	90.1	90.1
POCO [3]		6.21	77.4	87.1	5.14	84.9	93.7
NKF [58]		6.50	80.9	84.2	6.48	84.2	90.4
DOGNN [56]		4.93	85.9	85.7	4.85	89.3	92.4
Ours		**2.68**	**97.7**	**90.5**	**3.19**	**96.8**	**95.2**
POCO [3]	Synth. Rooms	5.96	82.5	82.0	6.52	80.3	85.9
NKF [58]		9.15	66.5	83.4	9.87	69.3	86.2
Ours		**5.38**	**86.6**	86.4	**5.01**	**90.5**	**91.8**
Ours	CARLA	3.20	95.9	89.1	3.08	98.1	95.0
Ours -	Mixed	3.72	93.6	89.1	3.17	97.4	95.5

Chamfer distance that reflects reconstruction accuracy (Acc.) and completeness (Comp.). We additionally report average running times for each method on the datasets. The mean/min/max number of input points in this setting are 490k/290k/820k. Compared to ours, SPSR is quite sensitive to the noise and sparsity in the input, leaving bumpy and incomplete geometries. Although POCO could reach a similar completeness value, the fitted surfaces fail to faithfully respect the input. The long running time (161x slower than ours) also prohibits POCO from practical use.

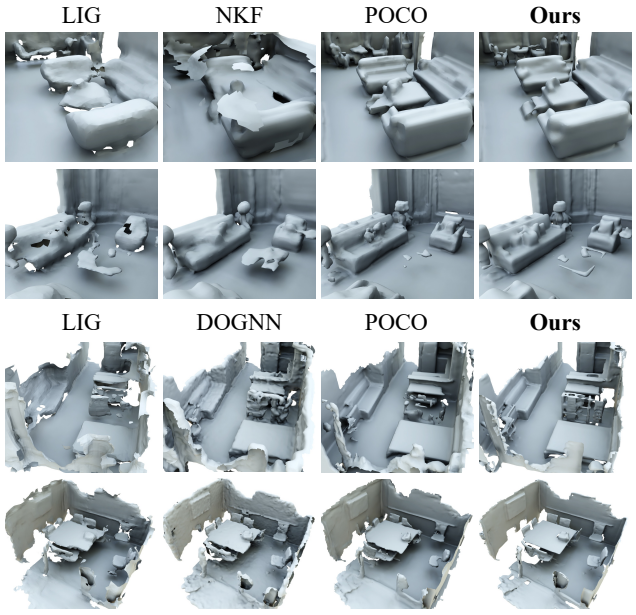


Figure 8: **Room-level datasets [4, 11] visualization.** All the models are trained only with ShapeNet.

4.3. Generalization across Domains and Densities

Across domains. We compare the generalizability of our method with others by directly applying the models trained on ShapeNet and Synthetic Room dataset (Synth. Rooms) [43] to room-level datasets, *i.e.*, ScanNet [11] and

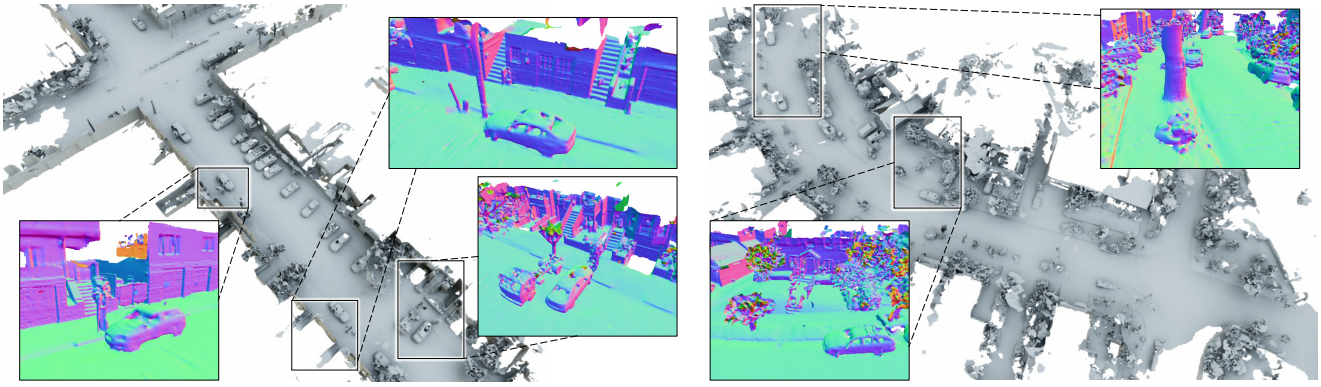


Figure 9: **Application to Waymo [49] dataset.** We run our kitchen-sink-model in an out-of-core manner (see Appendix for implementation details) to scale to very large scenes consisting of 10M / 11M (left / right) points, taking only 20s / 35s.

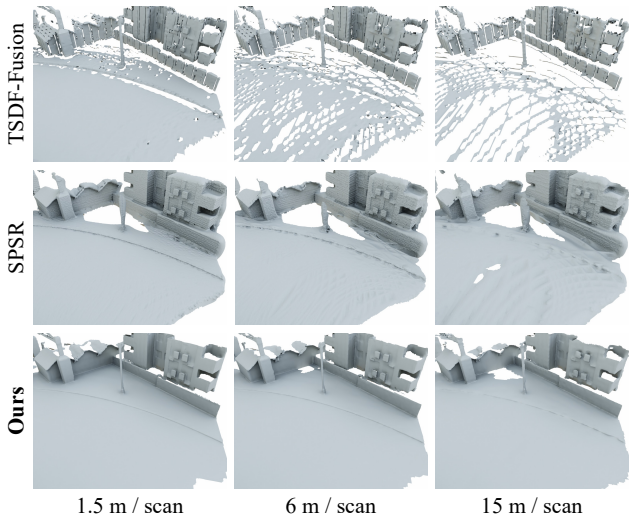
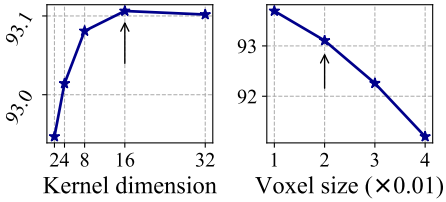


Figure 10: **Generalization to different densities.**

Table 5: **Performance comparison using different input densities.** Here the F-Score ↑ metric is shown.

Meters / scan	1.5	3	6	9	12	15
TSDF-Fusion [32]	80.0	80.7	78.4	74.8	70.6	67.8
SPSR [32]	88.2	88.3	87.9	86.4	83.3	79.5
Ours	**96.1**	**96.0**	**95.4**	**94.1**	**92.6**	**92.0**



	IoU
w/o Hier.	92.1 **(-1.0)**
w/o Grad.	91.5 **(-1.6)**

Figure 11: **Ablation study.** IoU metric is shown. The back arrows indicate the setting we use to obtain Tab. 2.

the test split of Matterport3D [4]. For both datasets we sample 10K points as input, and normalize the scale to roughly match the training set. As shown by the comparisons in Fig. 8 and Tab. 4, the generalization of our method is significantly better than the baselines, with ShapeNet training set reaching the highest accuracy possibly due to its diversity and similar geometric distributions.

Across densities. We test the robustness of our model under sparse input by keeping only one scan of LiDAR frame within a fixed driving distance in our CARLA dataset ('Novel' subset). The results are shown in Fig. 10 and Tab. 5. At the level of extreme sparsity our method is still able to reconstruct complete geometry (*e.g.* the ground) while the baselines start to degrade.

4.4. Ablation Study

We run our method with different feature dimensions d for kernel computation, as well as different voxel sizes W, and the results are shown in Fig. 11. While increasing the feature dimension helps reach a slightly better performance, the influence of voxel sizes is more prominent. We try to remove the hierarchies from the linear solver by setting $\{\alpha_i^{(l)} \mid l > 0\}$ to 0 ('w/o Hier.'), or remove the gradient-based matrices $\mathbf{Q}^\top \mathbf{Q}$ ('w/o Grad.'). Both of the settings lead to a degraded performance, showing the effectiveness of our design choices.

5. Conclusion

In this paper we present NKSR, an accurate and scalable surface reconstruction algorithm using the neural kernel field representation. We show by extensive experiments that our method reaches state-of-the-art quality and efficiency, while enjoying good generalization to unseen data. We believe our method further pushes the boundary of the field of 3D reconstruction and makes deep-learning-based surface reconstruction more practical for general use. For future work we will try further improving the reconstruction quality using more expressive kernel models, as well as reducing memory footprint to allow for even larger-scale reconstructions.

References

[1] Matan Atzmon, Koki Nagano, Sanja Fidler, Sameh Khamis, and Yaron Lipman. Frame averaging for equivariant shape space learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 631–641, 2022.

[2] Abhishek Badki, Orazio Gallo, Jan Kautz, and Pradeep Sen. Meshlet priors for 3d mesh reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2849–2858, 2020.

[3] Alexandre Boulch and Renaud Marlet. Poco: Point convolution for surface reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6302–6314, 2022.

[4] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2017.

[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.

[6] Hao-Xiang Chen, Jiahui Huang, Tai-Jiang Mu, and Shi-Min Hu. Circle: Convolutional implicit reconstruction and completion for large-scale indoor scene. In *European Conference on Computer Vision (ECCV)*, pages 506–522. Springer, 2022.

[7] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 45–54, 2020.

[8] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European Conference on Computer Vision (ECCV)*, pages 628–644. Springer, 2016.

[9] Lei Chu, Hao Pan, and Wenping Wang. Unsupervised shape completion via deep prior in the neural tangent kernel perspective. *ACM Transactions on Graphics (TOG)*, 40(3), 2021.

[10] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[11] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[12] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 31–44, 2020.

[13] Boyang Deng, John P Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Nasa neural articulated shape approximation. In *European Conference on Computer Vision (ECCV)*, pages 612–628. Springer, 2020.

[14] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 12200–12209, 2021.

[15] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Learning elementary structures for 3d shape generation and matching. *arXiv preprint arXiv:1908.04725*, 2019.

[16] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[17] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J Mitra, and Michael Wimmer. Points2surf learning implicit surfaces from point clouds. In *European Conference on Computer Vision (ECCV)*, pages 108–124. Springer, 2020.

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

[19] Matheus Gadelha, Rui Wang, and Subhransu Maji. Deep manifold prior. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1107–1116, 2021.

[20] Jun Gao, Wenzheng Chen, Tommy Xiang, Clement Fuji Tsang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. *arXiv preprint arXiv:2011.01437*, 2020.

[21] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4857–4866, 2020.

[22] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision (ECCV)*, pages 484–499. Springer, 2016.

[23] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 9785–9795, 2019.

[24] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 216–224, 2018.

[25] Oshri Halimi, Ido Imanuel, Or Litany, Giovanni Trappolini, Emanuele Rodolà, Leonidas Guibas, and Ron Kimmel. Towards precise completion of deformable shapes. In *European Conference on Computer Vision (ECCV)*, pages 359–377. Springer, 2020.

[26] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 412–420. IEEE, 2017.

[27] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn. *ACM Transactions on Graphics (TOG)*, 38(4), Jul 2019.

[28] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *arXiv preprint arXiv:2005.11084*, 2020.

[29] Geoffrey E Hinton and Russ R Salakhutdinov. Using deep belief nets to learn covariance kernels for gaussian processes. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008.

[30] Jiahui Huang, Hao-Xiang Chen, and Shi-Min Hu. A neural galerkin solver for accurate surface reconstruction. *ACM Transactions on Graphics (TOG)*, 41(6), 2022.

[31] Jiahui Huang, Shi-Sheng Huang, Haoxuan Song, and Shi-Min Hu. Di-fusion: Online implicit 3d reconstruction with deep priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8932–8941, 2021.

[32] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3), 2013.

[33] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[34] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[35] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[36] Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. Deep implicit moving least-squares functions for 3d reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1788–1797, 2021.

[37] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2837–2845, 2021.

[38] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[39] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. Deep mesh reconstruction from single rgb images via topology modification networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 9964–9973, 2019.

[40] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[41] Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. *arXiv preprint arXiv:1910.05199*, 2019.

[42] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, Andreas Geiger, et al. Shape as points: A differentiable poisson solver. *arXiv preprint arXiv:2106.03452*, 2021.

[43] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020.

[44] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[45] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[46] Davis Rempe, Tolga Birdal, Yongheng Zhao, Zan Gojcic, Srinath Sridhar, and Leonidas J Guibas. Caspr: Learning canonical spatiotemporal point cloud representations. *arXiv preprint arXiv:2008.02792*, 2020.

[47] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *Advances in Neural Information Processing Systems*, 2021.

[48] Vincent Sitzmann, Eric R Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. *arXiv preprint arXiv:2006.09662*, 2020.

[49] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[50] Weiwei Sun, Andrea Tagliasacchi, Boyang Deng, Sara Sabour, Soroosh Yazdani, Geoffrey Hinton, and Kwang Moo Yi. Canonical capsules: Unsupervised capsules in canonical pose. *arXiv preprint arXiv:2012.04718*, 2020.

[51] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11358–11367, 2021.

[52] Shubham Tulsiani, Or Litany, Charles R Qi, He Wang, and Leonidas J Guibas. Object-centric multi-view aggregation. *arXiv preprint arXiv:2007.10300*, 2020.

[53] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2626–2634, 2017.

[54] Benjamin Ummenhofer and Vladlen Koltun. Adaptive surface reconstruction with multiscale convolutional kernels. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5651–5660, October 2021.

[55] Ignacio Vizzo, Tiziano Guadagnino, Jens Behley, and Cyrill Stachniss. Vdbfusion: Flexible and efficient tsdf integration of range sensor data. *Sensors*, 22(3), 2022.

[56] Peng-Shuai Wang, Yang Liu, and Xin Tong. Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Transactions on Graphics (TOG)*, 41(4), 2022.

[57] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. Adaptive o-cnn: A patch-based deep representation of 3d shapes. *ACM Transactions on Graphics (TOG)*, 37(6), 2018.

[58] Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural fields as learnable kernels for 3d reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18500–18510, 2022.

[59] Francis Williams, Jerome Parent-Levesque, Derek Nowrouzezahrai, Daniele Panozzo, Kwang Moo Yi, and Andrea Tagliasacchi. Voronoinet: General functional approximators with local support. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[60] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun 2019.

[61] Francis Williams, Matthew Trager, Joan Bruna, and Denis Zorin. Neural splines: Fitting 3d surfaces with infinitely-wide neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[62] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[63] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, William T Freeman, and Joshua B Tenenbaum. MarrNet: 3D Shape Reconstruction via 2.5D Sketches. In *Advances in Neural Information Processing Systems*, 2017.

[64] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *European Conference on Computer Vision (ECCV)*, pages 386–402, 2018.

[65] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2790–2799, 2018.

[66] Ge Zhang, Or Litany, Srinath Sridhar, and Leonidas Guibas. Strobenet: Category-level multiview reconstruction of articulated objects. *arXiv preprint arXiv:2105.08016*, 2021.

[67] Jingyang Zhang, Yao Yao, Shiwei Li, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Critical regularizations for neural surface reconstruction in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6270–6279, 2022.

[68] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3d point capsule networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1009–1018, 2019.

[69] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. *arXiv preprint arXiv:2104.03670*, 2021.

[70] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.