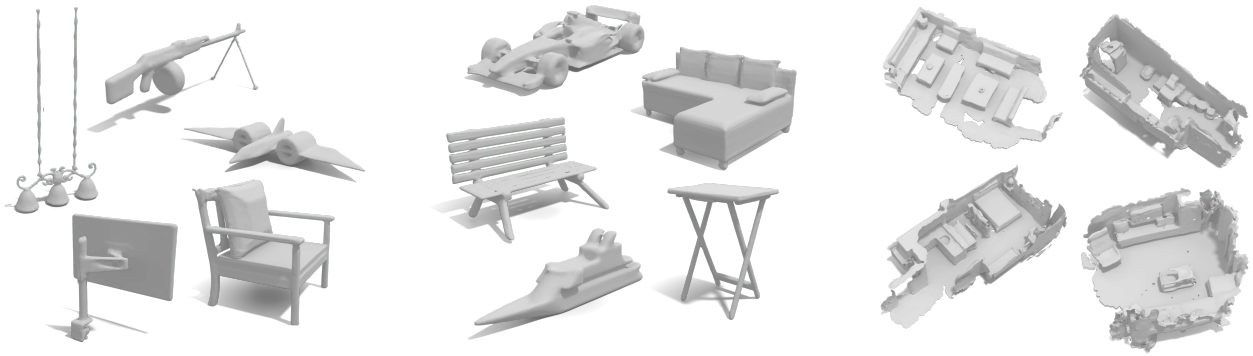
Neural Fields as Learnable Kernels for 3D Reconstruction

Francis Williams*[1,2], Zan Gojcic*[1,3], Sameh Khamis[1], Denis Zorin[2],
Joan Bruna[2], Sanja Fidler[1,4,5] Or Litany[1]

[1]NVIDIA [2]New York University [3]ETH Zürich [4]University of Toronto [5]Vector Institute



In-category reconstruction **Out-of-category reconstruction** **Generalization to scanned scenes**

Figure 1: Trained on synthetic shapes, NKF can reconstruct objects in and out of the training distribution, and scanned scenes.

Abstract

We present Neural Kernel Fields: a novel method for reconstructing implicit 3D shapes based on a learned kernel ridge regression. Our technique achieves state-of-the-art results when reconstructing 3D objects and large scenes from sparse oriented points, and can reconstruct shape categories outside the training set with almost no drop in accuracy. The core insight of our approach is that kernel methods are extremely effective for reconstructing shapes when the chosen kernel has an appropriate inductive bias. We thus factor the problem of shape reconstruction into two parts: (1) a backbone neural network which learns kernel parameters from data, and (2) a kernel ridge regression that fits the input points on-the-fly by solving a simple positive definite linear system using the learned kernel. As a result of this factorization, our reconstruction gains the benefits of data-driven methods under sparse point density while maintaining interpolatory behavior, which converges to the ground truth shape as input sampling density increases. Our experiments demonstrate a strong generalization capability to objects outside the train-set category and scanned scenes. Source code and pretrained models are available at https://nv-tlabs.github.io/nkf.

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 surface for that shape. Surface reconstruction from point clouds is a well studied topic in computer vision and graphics, with applications in robotics, entertainment, and manufacturing. Techniques for surface reconstruction broadly fall into two types: implicit methods which aim to recover a volumetric function whose zero level-set encodes the surface, and explicit methods which directly recover a triangle mesh from the input points. While implicit approaches can adapt to arbitrary topologies, the requirement to store a dense volumetric field led many past works to favor explicit approaches [44, 19]. More recently, implicit approaches have regained popularity due to a number of works demonstrating that neural networks are compact and effective at encoding signed-distance [45, 57] and occupancy fields [41, 48]. These works pair neural field[1] representations with modern advances in point cloud processing architectures to produce powerful reconstruction techniques. Current state-of-the-art shape reconstructions methods can be categorized along three axes (Fig. 3):

(1) ***Feed-forward vs. test-time optimization***: Feed-forward methods leverage shape priors to directly predict a surface from input points. While these methods are fast, they are not strictly constrained by their input and thus may perform a task more akin to retrieval than reconstruction (see [59]

[1]A *neural field* refers to the parameterization of a continuous function of spatial coordinates using a neural network. In this work we focus on scalar functions mapping coordinates to real numbers.

*Denotes equal contribution.

and Fig. 2, top). This results in decreased generalization performance on out-of-distribution shapes and input point densities. In contrast, test-time optimization via latent space traversal allows adaptation to the input, but is slow and can converge to poor local minima (See e.g. [16] and Fig. 2, bottom).

(2) ***Whether or not to leverage data priors***: Data-free methods recover the surface by minimizing the residuals between the reconstructed surface and input points, leveraging a predetermined prior to control the behavior away from the input points (e.g. a smooth space of functions [35, 64] or, emergent regularization arising from neural architectures [63, 25]). Such fixed priors are, however, difficult to tailor to specific tasks, like completion of partial shapes (Fig. 2, middle). Data-driven approaches, on the other hand, can learn task-specific priors to predict shapes that resemble a given dataset.

(3) ***Which scale to process and represent data***. Local-scale methods [33, 4] use the idea that complex structures can be reduced to a collection of simpler geometric primitives. These methods learn local models which are used to reconstruct a surface in patches. While this approach can generalize better, patch-size plays a critical role and must be carefully tuned per object (Fig. 2, bottom). Furthermore, without any notion of global context, these methods are unable to complete larger missing regions, leaving a fundamental gap in their generalization performance.

Based on these axes and the motivating examples in Fig. 2, we identify the need for a method that can learn good priors from a simple collection of shapes to drive 3D reconstruction of both in-distribution and out-of distribution shapes and scenes. In particular, the priors learned by this method should respect the input points, performing reconstruction rather than retrieval.

We thus propose a method using a novel representation of neural fields based on learned kernels, which we call *Neural Kernel Fields* (NKFs). In brief, NKFs work by learning a positive definite kernel conditioned on an input point cloud, and then using that kernel to predict an implicit shape by solving a simple linear system (Fig. 4). Our approach provides several key benefits: First, since predicted kernels are conditioned on the input and learned from data, they enjoy the versatility of learning-based methods. Second, since NKFs leverage a kernel for shape prediction, any reconstructed surfaces respect the input points by construction. Third, unlike gradient descent-based latent space optimization, at test-time NKF kernel weights are solved in closed form via a simple convex least-squares problem, guaranteeing good minima. Finally, our kernel acts as a global aggregator of spatially local features, allowing our method to work at a wide variety of sampling densities without tuning any scale parameters. The result is a generalizable method that can be trained only on synthetic shapes to seamlessly reconstruct out of distribution shapes and large scale scenes, while being robust to changes in input point density. Compared with the baselines, our method achieves a marked improvement reconstruction detail on both in and out-of distribution shapes. We summarize our contributions as follows:

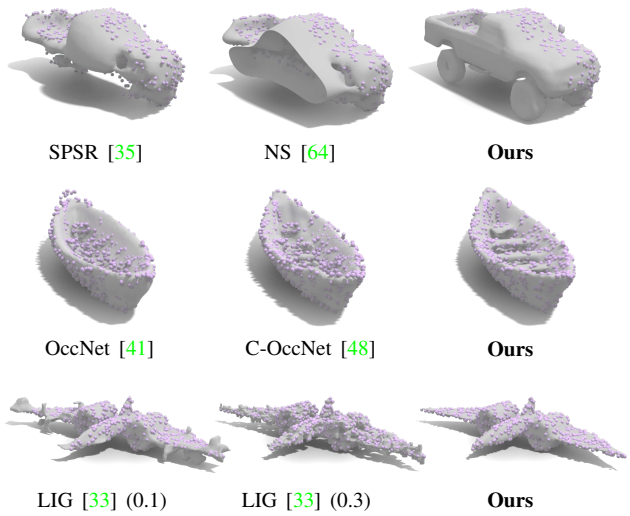


Figure 2: Comparison of our approach with methods along three Axes in Sec. 1. *Top Row:* Data free methods [35, 64] respect the input points but their simple fixed priors cannot complete the partial shape. *Middle Row:* Feed-forward methods [48, 48] learn from data, but miss the slats on the slightly out of distribution canoe. *Bottom Row:* LIG [33], a local method which performs test-time optimization, is very sensitive to the choice of patch size (0.3 left vs 0.1 middle), and gets stuck in bad local minima (bumpy artefacts).

- We introduce Neural Kernel Fields, a novel representation of neural fields for 3D reconstruction, which outputs highly detailed surfaces that respect the input points.
- Our NKF representation achieves state of the art performance on ShapeNet reconstruction (Section 4.1).
- We show state-of-the-art generalization performance on out-of-distribution shapes (Section 4.3), scenes (Section 4.4) and point densities (Section 4.5)

2. Related Work

Figure 3 visualizes existing implicit 3D shape reconstruction methods along the three axes defined in Section 1. Our Neural Kernel Field approach lies at the center of the diagram since it (1) uses a simple *convex* test time optimization, (2) leverages priors learned from data, and (3) learns local features on a spatial grid, but aggregates these globally during fitting.

We now highlight several works that are particularly relevant to our approach: Learned kernels were investigated in [66, 32, 46] and used for tasks such as few-shot transfer

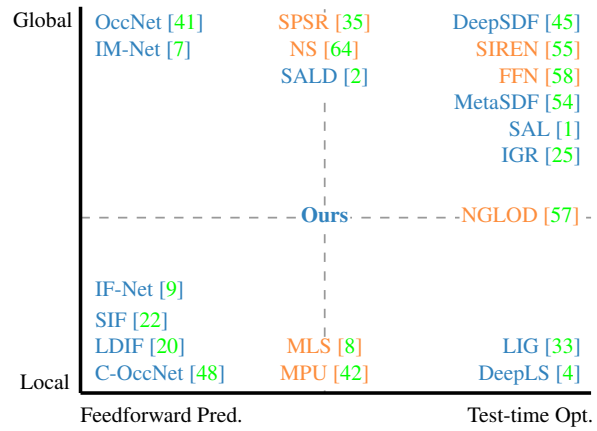


Figure 3: Taxonomy of design choices for methods which reconstruct implicit shapes from point clouds. The x and y axes correspond to axes 1 and 3 discussed in Section 1 respectively. Color corresponds to axis 2: **blue** methods use learned priors, and **orange** methods do *not*.

learning and classification of images. Neural Splines [64] used a kernel method derived from infinitely wide ReLU networks to reconstruct 3D surfaces from points. Convolutional Occupancy Networks [48] proposes a convolutional architecture that maps 3D points to features. We use a similar feature network for our Neural Kernel Field architecture. LIG [21] addresses the need for reconstruction methods that can generalize. MetaSDF [54] meta-learns a network which can be rapidly trained to predict SDFs. Neural Kernel Fields can also be viewed as a form of meta-learning since they predict a kernel machine from data. Shape as Points [47] is a concurrent work relevant to our method. It solves a linear system to reconstruct a surface after a learned upsampling phase. Unlike our method, however, Shape as Points relies on the inductive bias of Poisson reconstruction to output a surface rather than learning an inductive bias from data.

Beyond methods based on implicit surfaces, other shape reconstruction techniques exist which leverage different output representations. These representations include dense point clouds [51, 40, 73, 49, 50, 72, 56, 69, 70, 17, 36], polygonal meshes [30, 6, 19, 29, 24, 62, 12, 38, 27, 53], manifold atlases [63, 15, 26, 18, 3], and voxel grids [10, 60, 28, 67, 61, 23]. While our method focuses on shape reconstruction from points, past work has used neural fields to perform a variety of 3D tasks such as shape compression [57, 64], shape prediction from images [41, 37], voxel grid upsampling [48, 41], reconstruction from rotated inputs [14] and articulated poses [13, 71], and video to 3D [68, 39].

3. Method

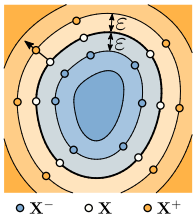
Our approach predicts an implicit surface from an oriented point cloud using a learned kernel. Neural Splines [64] also solves a 3D reconstruction problem using a fixed kernel (not learned from data), and is thus related to our approach. To introduce the reader to kernel methods for 3D reconstruction, we begin by giving an overview of Neural Splines. We then show how these kernel methods can be extended into Neural Kernel Fields capable of leveraging priors from data.

3.1. Review of Neural Splines

Given a point set $X = \{\boldsymbol{x}_i \in \mathbb{R}^3\}_{i=1}^S$ with corresponding normals $N = \{\boldsymbol{n}_i \in \mathbb{R}^3\}_{i=1}^S$, [64] seeks an implicit field $f : \mathbb{R}^3 \to \mathbb{R}$ which represents the underlying surface from which X and N were sampled. Namely, it should zero out on the set of input points and its gradient should equal the normal direction. More formally, the implicit field should minimize

$$L(f) = \sum_{i=1}^{S} |f(\boldsymbol{x}_i)|^2 + \|\nabla f(\boldsymbol{x}_i) - \boldsymbol{n}_i\|^2 \tag{1}$$

The gradient part of (1) can be approximated with a finite difference method, by augmenting the points X with $X^+ = \{\boldsymbol{x}_i^+ = \boldsymbol{x}_i + \epsilon \boldsymbol{n}_i\}_{i=1}^S$ and $X^- = \{\boldsymbol{x}_i^- = \boldsymbol{x}_i - \epsilon \boldsymbol{n}_i\}_{i=1}^S$ (see inset figure) and minimizing the simpler loss:



$$L(f) = \sum_{i=1}^{S} |f(\boldsymbol{x}_i)|^2 + |f(\boldsymbol{x}_i^+) - \epsilon|^2 + |f(\boldsymbol{x}_i^-) + \epsilon|^2 \tag{2}$$

Let $X' = X \cup X^+ \cup X^-$ denote the union of the augmented points. To minimize (2), we represent f as a weighted sum of kernel basis functions centered at the points X':

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{x}' \in X'} \alpha_i K_{\text{NS}}(\boldsymbol{x}, \boldsymbol{x}') \tag{3}$$

which is linear in the coefficients $\boldsymbol{\alpha} = \begin{bmatrix}\alpha_1 & \dots & \alpha_{3S}\end{bmatrix}^T$. These coefficients can thus be recovered by solving the linear system

$$(\boldsymbol{G} + \lambda \boldsymbol{I})\boldsymbol{\alpha} = \boldsymbol{y} \tag{4}$$

where $\boldsymbol{G} \in \mathbb{R}^{3S \times 3S}$ is the augmented Gram matrix over the points X' (*i.e.* $\boldsymbol{G}_{ij} = K_{\text{NS}}(\boldsymbol{x}'_i, \boldsymbol{x}'_j)\ \forall \boldsymbol{x}'_i, \boldsymbol{x}'_j \in X'$), $\lambda > 0$ is an optional regularizer which can be used to filter noise, and $\boldsymbol{y}$ is a vector such that

$$y_j = \begin{cases} 0 & \text{if } \boldsymbol{x}'_j \in X \\ +\epsilon & \text{if } \boldsymbol{x}'_j \in X^+ \\ -\epsilon & \text{if } \boldsymbol{x}'_j \in X^- \end{cases} \tag{5}$$

The kernel function K_{NS} is the closed form expression for an infinitely wide shallow ReLU network. It depends on the inner product between the inputs expressed in homogeneous coordinates. i.e. $K_{\text{NS}}(x, y) = K_{\text{NS}}(\langle x, y \rangle + 1)$. See the appendix for the exact equation and more details.

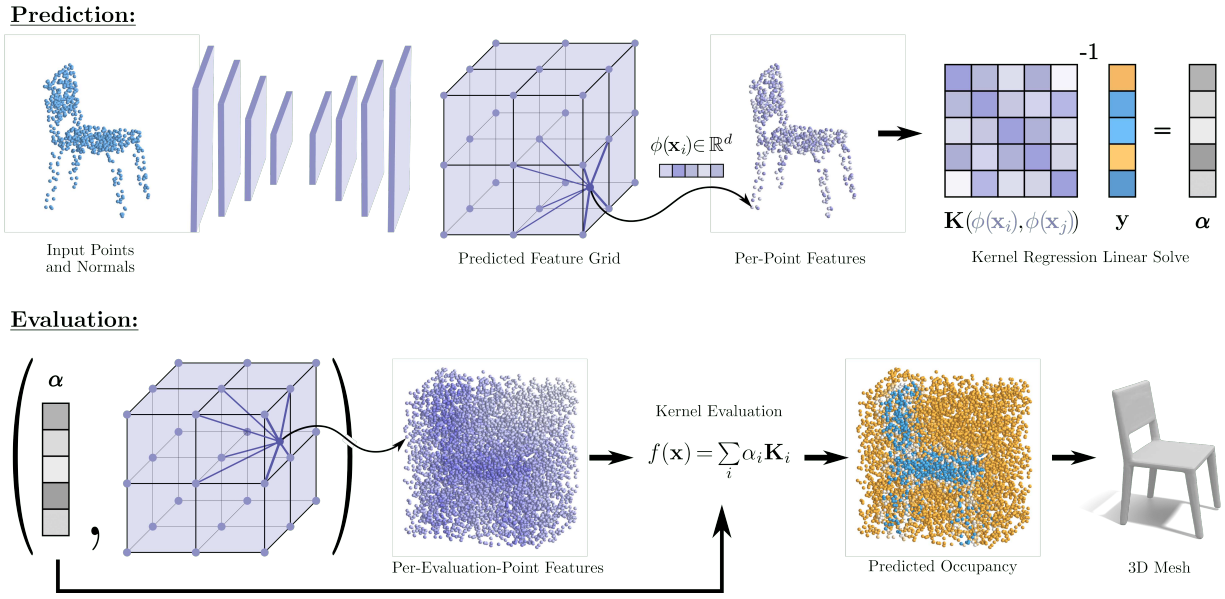


Figure 4: Our method works in two stages: (1) *prediction* (Top row) where we predict an implicit function from an input point cloud, and (2) *evaluation* (Bottom row) where we evaluate the implicit function. Our predicted implicit consists of a feature function ϕ which lifts points in the volume to features in $\mathbb{R}^d$, and a set of coefficients $\boldsymbol{\alpha}$, which are used to encode the function as a linear combination of basis functions centered at the input points.

3.2. Inductive Bias of Neural Splines

The kernel formulation in Neural Splines makes explicit the notion of *inductive bias*, *i.e.* the behavior of solutions away from the input points. To see this, we observe that solutions to the linear system (4) are solutions to the following constrained optimization problem:

$$\text{minimize } \|f\|_K = \boldsymbol{\alpha}^T \boldsymbol{G} \boldsymbol{\alpha} \tag{6}$$

$$\text{subject to } f(\boldsymbol{x}'_j) = y_j \qquad \boldsymbol{x}'_j \in X' \tag{7}$$

Here the norm $\|f\|_K$ being minimized defines the inductive bias of the kernel method, *i.e.* it governs the behavior of the function away from the constraints. The constraints $f(\boldsymbol{x}_j) = y_j$ guarantee that any solution to the above optimization problem interpolates the input data up to a bound defined by the regularizer λ.

For Neural Splines, the kernel norm favors smooth functions: It is proportional to curvature ($\|f\|_K \approx \|f''\|$) for 1D curves [65] and to the Radon transform of the Laplacian ($\|f\|_K \approx \|\mathcal{R}\{\Delta^2 f\}\|$) for 3D implicit surfaces [43, 64]. While an inductive bias favoring smoothness is good for reconstructing shapes with dense samples, it is too weak a prior in more challenging cases such as when the input points are very sparse or only cover part of a shape. For example, Fig. 2 (top) shows that Neural Splines is incapable of completing a partial point cloud of a truck. To this end, NKFs use a *data dependent kernel*, which learns an appropriate inductive bias conditioned on the input. By solving a linear system such as (4) using this kernel, we guarantee that output shapes respect their input points. We now describe NKFs in detail.

3.3. Neural Kernel Fields

Our model accepts the same inputs as Neural Splines described above in Section 3.1: *i.e.* We are given a set of points X and normals N sampled from the surface of an unknown shape, which we subsequently expand into an augmented point cloud X' with $2S$ points and corresponding labels $\boldsymbol{y} \in \mathbb{R}^{2S}$. We remark that our method only uses the inside and outside augmented points, *i.e.* $X' = X^+ \cup X^-$. For brevity, we denote the inputs to our model as $\mathcal{X} = (X', \boldsymbol{y})$. We now describe our architecture in four steps: (1) how to define our data dependent kernel, (2) how to use that kernel to predict an implicit function, (3) how to train our model, and (4) how to add filtering for noisy inputs. Figure 4 shows our NKF architecture pictorially.

Data Dependent Kernel To learn a kernel from data, we first augment input points $\boldsymbol{x}'_i \in X'$ with a feature $\phi(\boldsymbol{x}'_i | \mathcal{X}, \theta) \in \mathbb{R}^d$ where ϕ is a neural network with parameters θ conditioned on the inputs $\mathcal{X}$. Using these learned per-point features, we the define data-dependent kernel as:

$$K_{(\mathcal{X},\theta)}(\boldsymbol{x}, \boldsymbol{z}) = K_{\text{NS}}([\boldsymbol{x} : \phi(\boldsymbol{x}|\mathcal{X}, \theta)], [\boldsymbol{z} : \phi(\boldsymbol{z}|\mathcal{X}, \theta)]) \tag{8}$$

where $[\boldsymbol{a} : \boldsymbol{b}]$ is the concatenation of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, and K_{NS} is the Neural Spline kernel function. The architecture of the network ϕ follows an approach similar to Convolutional Occupancy Networks [48]: We discretize the volume around the input point cloud into a $M \times M \times M$ grid, and use a PointNet within each grid cell containing input points to extract a feature in that cell (empty cells have a zero fea-

ture). We then feed these features into a fully convolutional 3D U-Net, which produces an $M \times M \times M \times d$ grid of output features. To extract features per point, we trilinearly interpolate the output grid using the sampled points.

Predicting an Implicit Function To predict an implicit function, we find coefficients α_j for each input point $\boldsymbol{x}'_j \in X'$ by solving the $2S \times 2S$ positive definite linear system

$$\boldsymbol{\alpha} = [\alpha_j]_{j=1}^{2S} = (\boldsymbol{G}(\mathcal{X}, \theta) + \lambda \boldsymbol{I})^{-1} \boldsymbol{y} \tag{9}$$

where $\boldsymbol{G}(\mathcal{X}, \theta)$ is the gram matrix $\boldsymbol{G}(\mathcal{X}, \theta)_{ij} = K_{(\mathcal{X},\theta)}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and λ is a user supplied regularization parameter. To evaluate the predicted function at a new point $\boldsymbol{x}$, we compute the following equation using the coefficients $\boldsymbol{\alpha}$:

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{x}'_j \in X'} \alpha_j K_{(\mathcal{X},\theta)}(\boldsymbol{x}, \boldsymbol{x}'_j). \tag{10}$$

Training the Model To supervise our model during training, we use a dataset of shapes. Each shape consists of the augmented input points and labels $\mathcal{X} = (X', \boldsymbol{y})$, a dense set of points and occupancy labels ($X_{\text{vol}} = \{\boldsymbol{x}_i^{\text{vol}} \in \mathbb{R}^3\}, Y_{\text{vol}} = \{y_i^{\text{vol}} \in \mathbb{R}\}$) in the volume surrounding the shape, and a dense set of points $X_{\text{surf}} = \{\boldsymbol{x}_i^{\text{surf}} \in \mathbb{R}^3\}$ on the surface of the shape. We remark that the dense points on the surface and in the volume are only needed as supervision during training. The occupancy labels Y_{vol} denote whether a volume point lies inside or outside a shape and are defined as:

$$y_i^{\text{vol}} = \begin{cases} 1 & \text{if } \boldsymbol{x}_i^{\text{vol}} \text{ is inside the shape} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

We then train the network ϕ used to define the kernel (8) by first predicting an implicit function using the inputs $\mathcal{X}$ and then evaluating it at the dense volume X_{vol} and surface X_{surf} points to compute the loss:

$$L(f) = \sum_{i=1}^{|X_{\text{vol}}|} \text{BCE}(f(\boldsymbol{x}_i^{\text{vol}}), y_i^{\text{vol}}) + \lambda_{L1} \sum_{i=1}^{|X_{\text{surf}}|} |f(\boldsymbol{x}_i^{\text{surf}})| \tag{12}$$

The first term in (12) encourages the predicted function to have the correct occupancy, while the second term encourages the surface to agree with the ground truth shape. We backpropagate gradients through this loss to update the weights of the network ϕ, and thus learn the data dependent kernel.

Learning to Denoise We can optionally predict per-input point weights to make our solutions more robust to noise. We predict these via a fully connected network $w_j = \rho(\phi(\boldsymbol{x}_j; \mathcal{X}, \theta); \theta_w) \in \mathbb{R}$ mapping per-point input features to weights. Instead of Eq. 9, we then solve the *weighted* ridge regression problem:

$$\boldsymbol{\alpha} = (\boldsymbol{W}\boldsymbol{G}\boldsymbol{W} + \lambda \boldsymbol{I})^{-1} \boldsymbol{W} \boldsymbol{y} \tag{13}$$

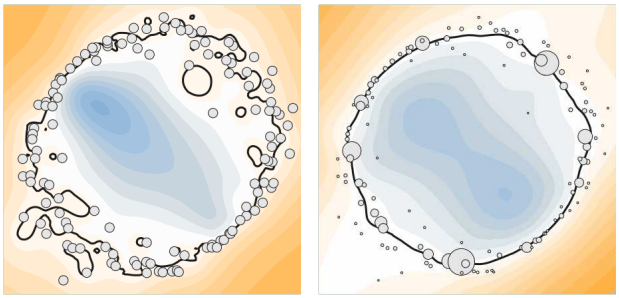


Figure 5: Unweighted (left) versus weighted (right) kernel ridge regression. Both reconstructions use the same noisy input points and regularization value. The right reconstruction, which uses per-point weights (visualized as the size of the points) can filter out the contribution of noisy points and produce a more accurate reconstruction.

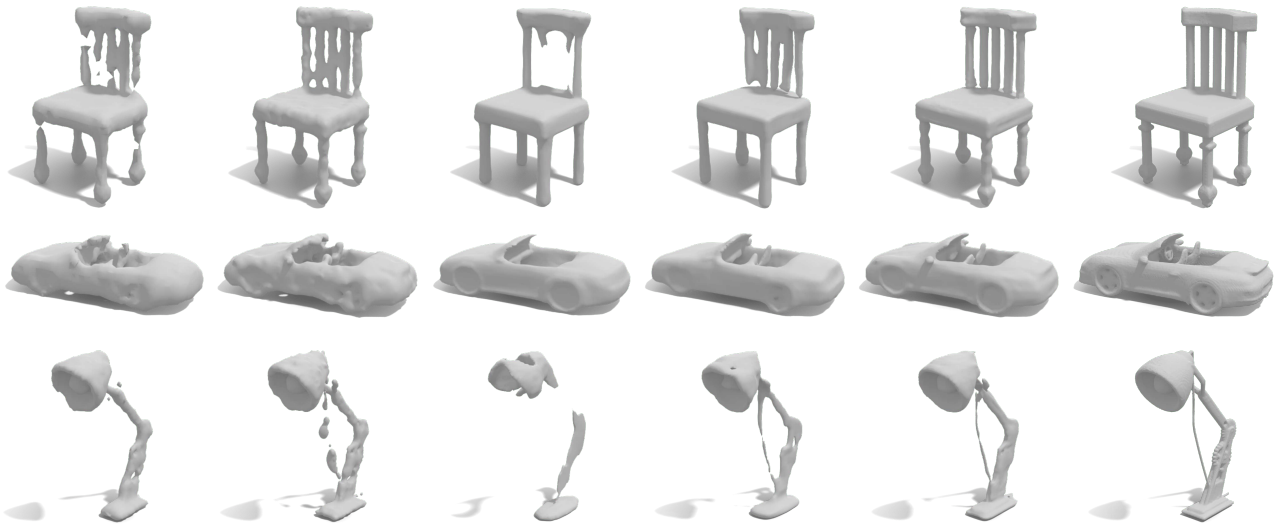
where $\boldsymbol{W} = \text{diag}(w_1, \ldots w_s)$ is a diagonal matrix of per-input-point weights. Figure 5 shows the effect of weighted versus unweighted ridge regression in the presence of noise on a toy example.

4. Experiments

We first evaluate the effectiveness of Neural Kernel Fields on the tasks of single object reconstruction (Section 4.1) and partial object completion (Section 4.2) using the ShapeNet [5] dataset. Next, we highlight NKF's ability to generalize by evaluating the tasks of out-of-category shape generlization (Section 4.3), generlization to full scenes (Section 4.4), and generlization to different sampling densities (Section 4.5). Finally, in Section 4.6, we ablate the design choices for our backbone architecture.

Baselines: For ShapeNet reconstruction, we compare our method to OccNet [41], Conv-OccNet [48], SPSR [35], and Neural Splines [64]. On the task of completion, we compare against Conv-OccNet [48]. For out-of-distribution shape reconstruction, we compare with OccNet [41], Conv-OccNet [48], LIG [33], and Neural Splines [64], while on the task of full scene reconstruction we use Conv-OccNet [48], SPSR [35], and NS [64] as baselines. Combined, these methods cover a broad spectrum of 3D shape reconstruction approaches and represent SoTA in their respective categories depicted in Fig. 3.

Metrics: We use 3 metrics for quantitative evaluation: *Intersection over Union (IoU)* is computed by sampling a set of 100k points in the volume around a watertight shape and computing the IoU of the set of inside points for the predicted and ground truth shapes. IoU indicates how well the predicted shape agrees with the ground truth both near and away from the surface. *L2 Chamfer Distance* is evaluated by sampling 100k points on the predicted and ground truth surfaces (extracted as meshes using marching cubes), then computing the average shortest distance between all pairs



SPSR [35] NS [64] OccNet [41] C-OccNet [48] **Ours** Ground truth

Figure 6: **Single object reconstruction on ShapeNet [5].** NKF recovers fine details like the lamp's cord and car's side mirror.

	Noise free						Noise std. = 0.0025						Noise std. = 0.005					
	IoU ↑		Chamfer ↓		Normal C. ↑		IoU ↑		Chamfer ↓		Normal C. ↑		IoU ↑		Chamfer ↓		Normal C. ↑	
	mean	std.	mean	std.	mean	std.	mean	std.	mean	std.	mean	std.	mean	std.	mean	std.	mean	std.
SPSR [35]	0.772	0.162	0.122	0.069	0.847	0.061	0.759	0.163	0.125	0.066	0.847	0.060	0.735	0.169	0.133	0.067	0.843	0.060
OccNet [41]	0.773	0.162	0.068	0.048	0.902	0.073	0.771	0.164	0.069	0.051	0.903	0.072	0.699	0.172	0.192	0.137	0.888	0.074
C-OccNet [48]	0.810	0.116	0.051	0.018	0.922	0.052	0.820	0.112	0.049	0.019	0.924	0.051	0.866	0.089	0.080	0.040	0.937	0.044
C-OccNet* [48]	0.823	0.105	0.048	0.016	0.928	0.048	0.847	0.094	0.043	0.015	0.932	0.046	0.863	0.088	0.078	0.031	0.937	0.045
NS [64]	0.864	0.151	0.051	0.071	0.926	0.059	0.831	0.147	0.054	0.064	0.919	0.057	0.791	0.155	0.121	0.167	0.900	0.055
Ours	**0.949**	0.053	**0.024**	0.010	**0.954**	0.042	**0.914**	0.061	**0.028**	0.010	**0.947**	0.043	**0.883**	0.074	**0.066**	0.018	**0.939**	0.041

Table 1: **Single object reconstruction on ShapeNet [5]**. NKF consistently outperforms strong baselines on standard metrics: IoU, Chamfer distance, and Normal Consistency, across all 13 categories.

of points. Chamfer distance measures how accurately each method reconstructs the surface of the input shape. *Normal Correlation* is computed as the average dot product between the normals at pairs of nearest points on the ground truth and predicted shapes and evaluates how well each method does at preserving the surface direction. We use the same 100k samples as for Chamfer distance to compute this metric.

4.1. Single Object Reconstruction on ShapeNet

We evaluate NKF's performance against strong baselines in reconstructing objects from 13 categories of the ShapeNet dataset. As input to all methods we use 1000 randomly sampled surface points to which we add Gaussian noise of different magnitudes. For learning based methods (Conv-Occnet, OccNet, Ours), we train a single model across all 13 categories per noise level. Since both NKF and Neural Splines utilize pairs of points spread along the normals, we train a version of Conv-OccNet with (C-OccNet*) and without (C-OccNet) these points. Table 1 shows that NKF achieves large improvements across all metrics, reaching near 95% IoU on noise-free reconstruction. Figure 6, which shows reconstructions at the middle noise level, clearly demonstrates how NKF recovers fine details like the cars' side-view mirror, the cord on the lamp, and the bulges on the chair legs. In the supplemental, we provide per-category results, additional figures, and ablations on different numbers of input points.

4.2. Shape Completion on ShapeNet

Albeit using input points as anchors, thanks to the global support of the kernel, NKF can learn to recover an entire shape from partial input. To demonstrate that, we sample a point cloud from up to 50 % of a shape surface along one of the principal axes, and supervise NKF to predict the full shape. We train a separate model per shape category for each of 13 ShapeNet categories. Table 2 presents quantitative results across all categories for this task. NKF achieves on-par Chamfer and Normal correlation as C-OccNet with substantially better IoU. The top row of Fig. 2 shows an example of completing a truck shape from very partial input. Note how NKF learned to leverage shape symmetry to faithfully recover unobserved regions like the wheels. The appendix shows per-category quantitative and qualitative results.

	IoU ↑		Chamfer ↓		Normal C. ↑	
	mean	std.	mean	std.	mean	std.
C-OccNet [48]	0.770	0.152	**0.075**	0.068	**0.909**	0.059
Ours	**0.819**	0.171	0.077	0.091	0.907	0.067

Table 2: **Object completion from partial point clouds.**

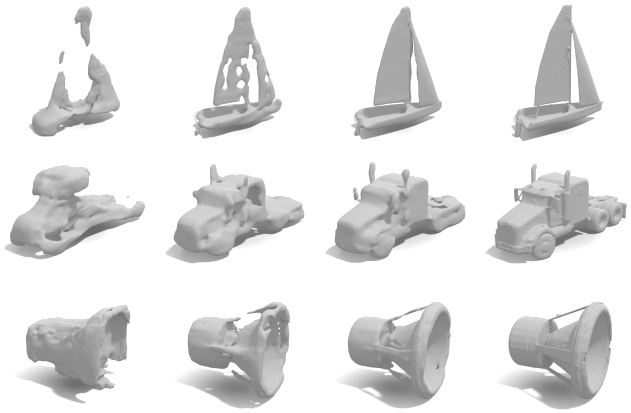


Figure 7: **Out-of-category generalization.** Reconstructed object from categories unseen during training.

4.3. Out of Category Generalization

Generalization to categories beyond the train set is key to making learnable methods useful in the wild. To evaluate NKF on this task we train all methods on 6 of the ShapeNet categories (*airplane, lamp, display, rifle, chair, cabinet*) and evaluate on the other 7 (*bench, car, loudspeaker, sofa, table, telephone, watercraft*). Table 3 presents quantitative statistics for this task using the standard metrics. NKF greatly outperforms both learned and non-learned baselines. Furthermore, we note in brackets the decrease in performance compared to the model trained on all categories. NKF, with a minimal 1.1% drop in IoU, aligns with data-free methods thanks to its test-time adaptation ability. We point out that LIG only provides models pretrained on all categories, which sets an upper bound on its generalization performance. The distinct differences between NKF and baselines are readily apparent in Figure 7.

4.4. Scene Reconstruction on ScanNet

Next, we extend beyond single objects and evaluate NKFs on ScanNet scenes. For this experiment, we followed the setup in [48] and trained our model on synthetic scenes consisting of random ShapeNet object placements. We found the synthetic floors and walls, added by [48] to the training set, harmed performance and, hence, trained our method without them. We report C-OccNet's results with and without walls for completeness. According to Table 4, for 10K input points, NKF achieves an average Chamfer distance of about half of the next best method. Figure 8 shows a comparison to baselines on 2 reconstructed rooms. Now how our method better captures small details such as the stepladder and shelf.

	IoU ↑	Chamfer ↓	Normal C. ↑
OccNet [41]	0.603 (-20.4%)	0.134 (0.070)	0.829 (-8.3%)
C-OccNet [48]	0.734 (-9.5%)	0.074 (0.023)	0.895 (-2.9%)
C-OccNet* [48]	0.785 (-4.9%)	0.064 (0.013)	0.911 (-1.7%)
LIG [33]	0.518 (N.A.)	0.112 (N.A.)	0.536 (N.A.)
NS [64]	0.869 (0.0%)	0.049 (0.000)	0.924 (0.0%)
Ours	**0.938 (-1.1%)**	**0.028 (0.003)**	**0.939 (-1.0%)**

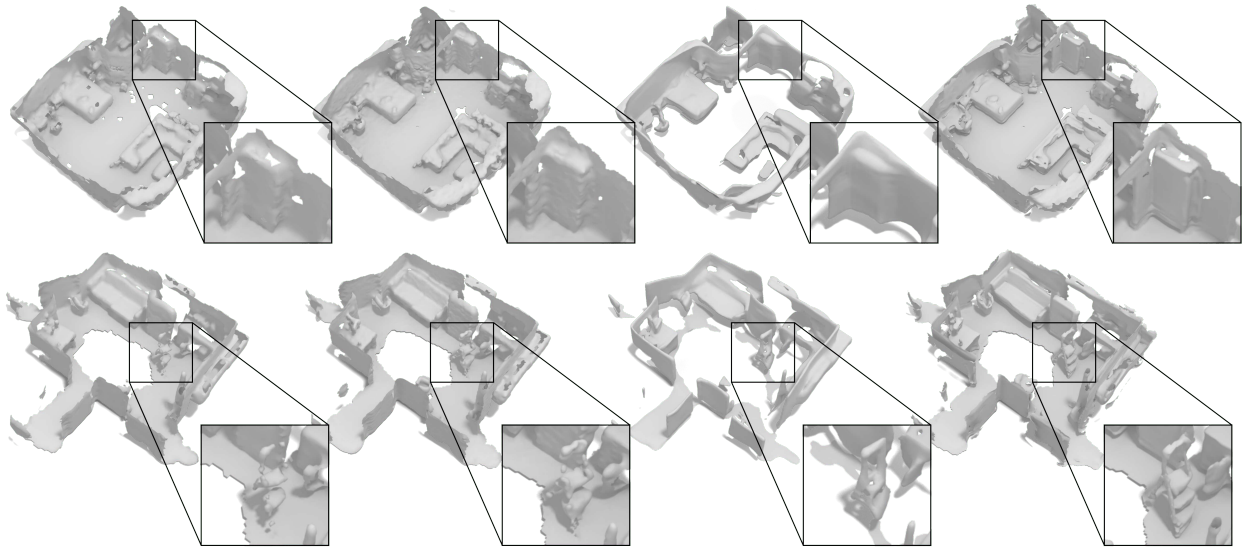
Table 3: **Generalization capacity of object-level 3D reconstruction from sparse points clouds.** We train all models using 6 ShapeNet categories (*airplane, lamp, display, rifle, chair, cabinet*) and evaluate them on the remaining 7 (*bench, car, loudspeaker, sofa, table, telephone, watercraft*). The numbers in the brackets denote the difference in performance with the model trained on all categories.

	Chamfer ↓	Normal C. ↑
C-OccNet (w. walls) [48]	0.133	0.779
C-OccNet (w.o. walls) [48]	0.074	0.843
SPSR [35]	0.060	0.871
NS [64]	0.060	**0.876**
Ours	**0.032**	0.873

Table 4: **Scene-level 3D reconstruction from sparse point clouds on ScanNet [11].** All methods use 10 000 input points for each scene.

4.5. Point Density Generalization

In real-world applications, point density may differ between train and test times. A good data-driven prior should compensate for lack of data (i.e. sparse inputs) without hindering data-rich settings (i.e. dense inputs). Therefore, we evaluate the response of NKF and various baseline methods to changes in input sampling density. We trained each method on 1000 input points and evaluated it on varying numbers of input samples (between 250 and 3000). To report the upper-bound performance of each method, we train additional models on each density value. Figure 9 shows the mean IoU of each method versus the number of input points. Curves with labels ending in "-1k" were trained on 1000 points, and otherwise, were trained and tested on the same number of points. OccNet shows no response to increased sampling density (even at train time). Although C-OccNet marginally improves when trained on denser data, it does not improve when evaluated with more points than it was trained on. The performance of Neural Splines improves for denser inputs, but is poor on sparse inputs as expected from



SPSR [35] NS [64] C-OccNet [48] **Ours**

Figure 8: **ScanNet reconstruction**. Trained on ShapeNet objects, NKF gracefully scales to real world scanned scenes.

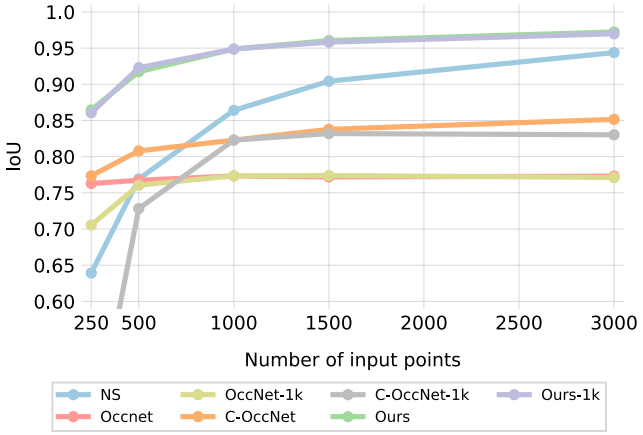


Figure 9: **ShapeNet IoU vs. number of input points.** Curves ending in "-1k" correspond to methods trained on 1000 points, and other methods were trained and evaluated on the same number of points. Our method performs well in the sparse and dense regimes and does not decay when trained and tested on different point densities.

data-free methods. Finally, our method works well in sparse settings and improves with increasing density. Moreover, it does not degrade if trained and tested on different sampling densities (the gray and green curves are nearly identical).

4.6. Ablations

We conduct an ablation study of our design choices on the task of shape reconstruction on ShapeNet. We experiment with using different per-point feature dimensions and whether to include the surface L_1 loss, $L(f)_{L1}$. Table 5 summarizes the results.

	feature dimension			
	8	16	32	64
without $L(f)_{L1}$	0.939	0.941	0.942	0.942
with $L(f)_{L1}$	0.945	0.947	**0.949**	**0.949**

Table 5: **Ablation study** (Section 4.1). NKFs benefits from the L1 surface loss and work well even with small feature dimensions. Values in the table are mean IoU on the test set.

5. Conclusion and Limitations

We presented a novel method for reconstructing and completing 3D shapes from sparse point clouds. Our method outperforms the state-of-the-art on object reconstruction and completion as well as scene reconstruction, while demonstrating strong generalization capability (both with respect to shape categories and input sampling density). While our method pushes the boundary on many fronts, it still has several limitations which we plan to address in future work: First, our current kernel implementation requires a dense linear solve, which limits the number of evaluation points to around 12k on a V100 GPU. State-of-the-art Kernel solvers in the literature (e.g. [52]) have scaled up to millions of points by leveraging techniques such as Nyström sampling. We plan to investigate how to leverage these approaches to handle larger inputs. Furthermore, we would like to investigate kernels with spatial decay to sparsify our linear system and scale our method to very large inputs. A second limitation is the requirement of oriented points. While these are usually available from sensors, they can be noisy. Thus, in the future we would like to incorporate normal prediction into our method so it can operate on unoriented point clouds.

References

[1] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. 3

[2] Matan Atzmon and Yaron Lipman. Sald: Sign agnostic learning with derivatives. *arXiv preprint arXiv:2006.05400*, 2020. 3

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

[4] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer, 2020. 2, 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. 5, 6, 14, 15, 16

[6] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 45–54, 2020. 3

[7] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 3

[8] Zhi-Quan Cheng, Yanzhen Wang, Bao Li, Kai Xu, Gang Dang, and Shiyao Jin. A survey of methods for moving least squares surfaces. In *VG/PBG@ SIGGRAPH*, pages 9–23, 2008. 3

[9] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020. 3

[10] 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*, pages 628–644. Springer, 2016. 3

[11] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 7

[12] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 31–44, 2020. 3

[13] Boyang Deng, John P Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Nasa neural articulated shape approximation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 612–628. Springer, 2020. 3

[14] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas Guibas. Vector neurons: A general framework for so (3)-equivariant networks. *arXiv preprint arXiv:2104.12229*, 2021. 3

[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. 3

[16] Shivam Duggal, Zihao Wang, Wei-Chiu Ma, Sivabalan Manivasagam, Justin Liang, Shenlong Wang, and Raquel Urtasun. Secrets of 3d implicit object shape reconstruction in the wild. *arXiv preprint arXiv:2101.06860*, 2021. 2

[17] 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*, pages 605–613, 2017. 3

[18] Matheus Gadelha, Rui Wang, and Subhransu Maji. Deep manifold prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1107–1116, 2021. 3

[19] 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. 1, 3

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

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

[22] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7154–7164, 2019. 3

[23] 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*, pages 484–499. Springer, 2016. 3

[24] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9785–9795, 2019. 3

[25] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020. 2, 3

[26] 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*, pages 216–224, 2018. 3

[27] Oshri Halimi, Ido Imanuel, Or Litany, Giovanni Trappolini, Emanuele Rodolà, Leonidas Guibas, and Ron Kimmel. The whole is greater than the sum of its nonrigid parts. *arXiv preprint arXiv:2001.09650*, 2020. 3

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

[29] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn. *ACM Transactions on Graphics*, 38(4):1–12, Jul 2019. 3

[30] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *arXiv preprint arXiv:2005.11084*, 2020. 3

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. 12

[32] 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. 2

[33] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6001–6010, 2020. 2, 3, 5, 7

[34] W Kahan. Computing cross-products and rotations. 12

[35] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. 2, 3, 5, 6, 7, 8, 15

[36] 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. 3

[37] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. Sdf-srn: Learning signed distance 3d object reconstruction from static images. *arXiv preprint arXiv:2010.10505*, 2020. 3

[38] 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. 3

[39] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2019–2028, 2020. 3

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

[41] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 5, 6, 7, 15

[42] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics (TOG)*, 22(3):463–470, 2003. 3

[43] Greg Ongie, Rebecca Willett, Daniel Soudry, and Nathan Srebro. A function space view of bounded norm infinite width relu nets: The multivariate case. *arXiv preprint arXiv:1910.01635*, 2019. 4

[44] 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*, pages 9964–9973, 2019. 1

[45] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 3

[46] 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. 2

[47] 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. 3

[48] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 3, 4, 5, 6, 7, 8, 15, 16

[49] 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*, pages 652–660, 2017. 3

[50] 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. 3

[51] 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. 3

[52] Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. Falkon: An optimal large scale kernel method. *arXiv preprint arXiv:1705.10958*, 2017. 8

[53] 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 *NeurIPS*, 2021. 3

[54] Vincent Sitzmann, Eric R Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. *arXiv preprint arXiv:2006.09662*, 2020. 3

[55] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020. 3

[56] 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. 3

[57] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021. 1, 3

[58] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Neural Information Processing Systems (NeurIPS)*, 2020. 3

[59] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2019. 1

[60] Shubham Tulsiani, Or Litany, Charles R Qi, He Wang, and Leonidas J Guibas. Object-centric multi-view aggregation. *arXiv preprint arXiv:2007.10300*, 2020. 3

[61] 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*, pages 2626–2634, 2017. 3

[62] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020. 3

[63] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. 2, 3

[64] Francis Williams, Matthew Trager, Joan Bruna, and Denis Zorin. Neural splines: Fitting 3d surfaces with infinitely-wide neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3, 4, 5, 6, 7, 8, 12, 15

[65] Francis Williams, Matthew Trager, Claudio Silva, Daniele Panozzo, Denis Zorin, and Joan Bruna. Gradient dynamics of shallow univariate relu networks. *arXiv preprint arXiv:1906.07842*, 2019. 4

[66] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016. 2

[67] 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. 3

[68] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *arXiv preprint arXiv:2003.09852*, 2020. 3

[69] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 386–402, 2018. 3

[70] 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*, pages 2790–2799, 2018. 3

[71] Ge Zhang, Or Litany, Srinath Sridhar, and Leonidas Guibas. Strobenet: Category-level multiview reconstruction of articulated objects. *arXiv preprint arXiv:2105.08016*, 2021. 3

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

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

	Pretrain on Chairs			Pretrain on All		
	IoU ↑	Chamfer ↓	Normal C. ↑	IoU ↑	Chamfer ↓	Normal C. ↑
airplane	0.922	0.021	0.945	0.951	0.016	0.962
bench	0.898	0.024	0.936	0.908	0.022	0.940
cabinet	0.938	0.043	0.947	0.968	0.028	0.962
car	0.913	0.037	0.882	0.937	0.030	0.913
chair	0.946	0.026	0.962	0.943	0.027	0.960
display	0.967	0.028	0.971	0.976	0.023	0.978
lamp	0.895	0.040	0.928	0.920	0.024	0.940
loudspeaker	0.931	0.059	0.935	0.965	0.033	0.952
rifle	0.889	0.115	0.937	0.957	0.012	0.970
sofa	0.971	0.025	0.967	0.974	0.024	0.969
table	0.939	0.028	0.964	0.951	0.025	0.969
telephone	0.985	0.018	0.986	0.988	0.017	0.988
watercraft	0.936	0.039	0.934	0.955	0.019	0.950
mean	0.929	0.036	0.939	0.949	0.024	0.954

Table 6: Comparison between model trained only on chairs (left column) to model trained on all categories.

A. Neural Spline Kernel Equation

The Neural Spline [64] kernel is defined as the limiting kernel for an infinitely wide ReLU network with either Gaussian or Uniform initialization (using Kaiming-He [31] initialization). In our implementation we use the Gaussian initialized version which has the following closed form solution:

$$K_{\text{NS}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{\|\tilde{\boldsymbol{x}}\|\|\tilde{\boldsymbol{x}}'\|}{\pi}\big(\sin\theta + 2(\pi - \theta)\cos\theta\big) \qquad \theta = \measuredangle(\boldsymbol{x}, \boldsymbol{x}') \tag{14}$$

where $\tilde{\boldsymbol{x}} = (\boldsymbol{x}, 1), \tilde{\boldsymbol{x}}' = (\boldsymbol{x}', 1)$ are the vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ expressed in homogeneous coordinates and $\theta = \measuredangle(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}')$ is the angle between the input vectors in homogeneous coordinates. In practice we compute the angle using the formula from Kahan [34]:

$$\theta = 2\arctan\Big(\frac{\Big\| \|\tilde{\boldsymbol{x}}'\|\tilde{\boldsymbol{x}} - \|\tilde{\boldsymbol{x}}\|\tilde{\boldsymbol{x}}' \Big\|}{\Big\| \|\tilde{\boldsymbol{x}}'\|\tilde{\boldsymbol{x}} + \|\tilde{\boldsymbol{x}}\|\tilde{\boldsymbol{x}}' \Big\|}\Big), \tag{15}$$

which is numerically stable, especially with small angles.

B. The Effect of Noise Filtering in 3D

Figure 10 shows the effect of weighting (Section 3.3) to filter noise in the input points. The left column shows our reconstruction without these learned weights, the middle column shows the effect of adding weighting, while the right column shows the ground truth surface. Notice how the weighted model is smoother and does not interpolate the input noise.

C. More Extreme Generalization

Table 6 and Figure 11 compare our reconstruction results using a model trained *only on chairs* to reconstruct the other 12 ShapeNet categories (airplane, bench, cabinet, car, display, lamp, loudspeaker, rifle, sofa, table, telephone, watercraft) against a model trained on all categories. The experimental setup is identical to Section 4.3 (1000 input points) except the model is trained only on chairs. Note how the performance of model trained only on chairs only drops slightly compared to the model trained on all categories.

D. Inference Timings

Our method uses a convex test-time optimization to perform inference of 3D shapes. We report the timing of each part of our method for the ShapeNet reconstruction (Section 4.1) and ScanNet reconstruction (Section 4.4) experiments in Table 7.

Ours (w/o weights) | Ours (weighted) | Ground truth

Figure 10: **The effect of noise filtering versus regularization.** The left column shows reconstructions using our method without any noise filtering and 0.1 regularization in the kernel ridge regression. The middle column shows these same models reconstructed with additional noise filtering (Section 3.3). Note how the regularized model still has bumps caused by the noisy input points while these are smoothed out by the filtering module.

	ShapeNet	ScanNet
Encoder	12.9ms	229.8ms
Decoder	0.3ms	0.42ms
Solve	30.3ms	3142ms
Eval	193.5ms	13254ms

Table 7: Timings on ShapeNet (1k input points and 2.1 million evaluation points) and ScanNet (10k input points and 16.9 million eval points).

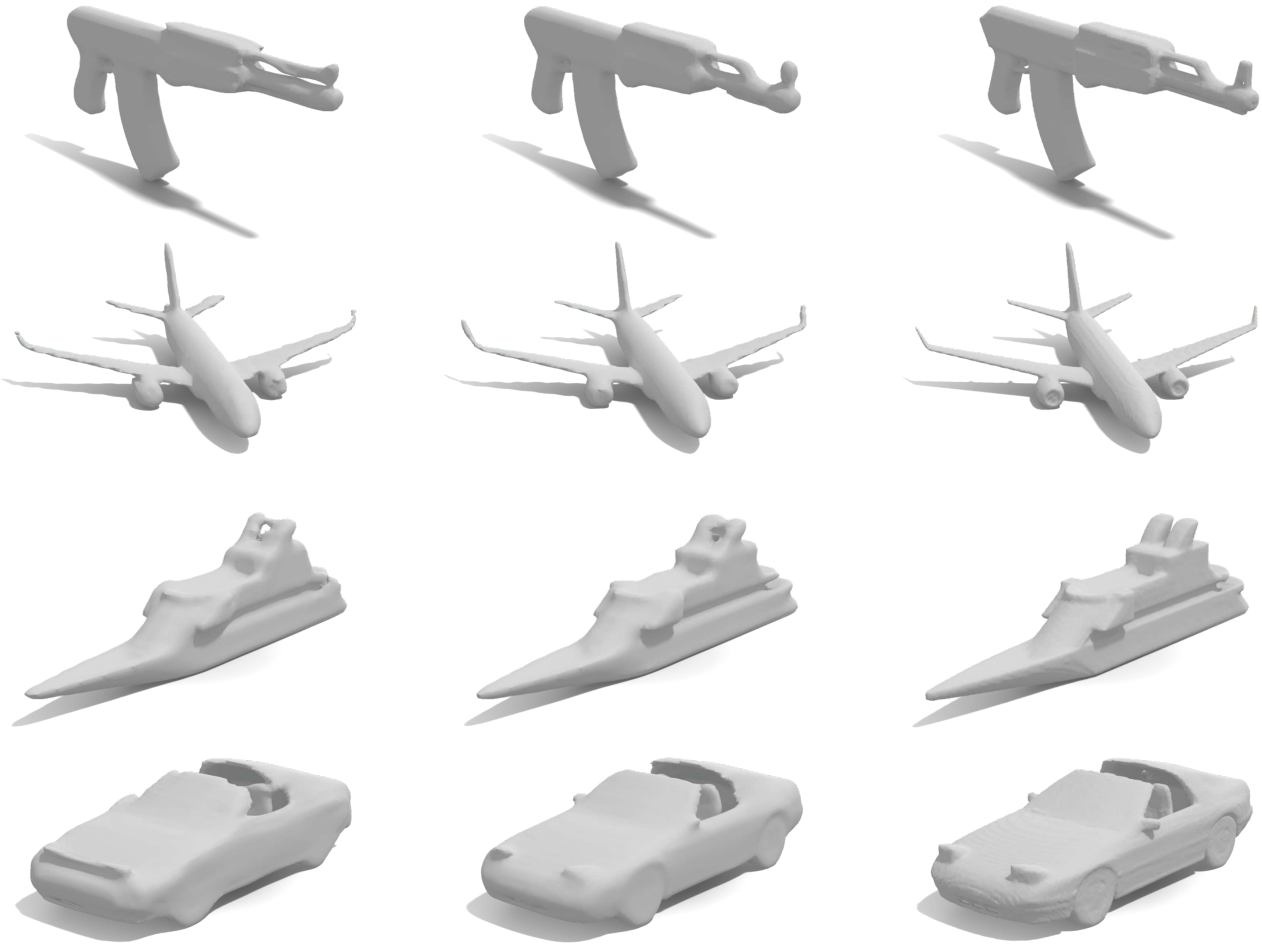
With 1000 input points for ShapeNet, we evaluated on a grid of size 128^3 (2.1M points), and with 8000 input points for ScanNet, we evaluated on a grid of size 256^3 (16.78M points. We implemented the kernel evaluation as a single monolithic CUDA kernel and report the timings on a Quadro GV100 GPU.

E. Additional ShapeNet Reconstruction Figures

Figure 12 shows additional reconstruction comparisons (with 0.0025) noise as described in Section 4.1.

F. Additional ShapeNet Generalization Figures

Figure 13 shows additional reconstructions for the out-of-category reconstruction experiment described in Section 4.3.



Ours (chairs) Ours (all) Ground truth

Figure 11: **Out of category generalization on ShapeNet [5].** Our model trained only on *chairs* (left) can seamlessly generalize to other 12 ShapeNet categories, achiving only sligthly worse performance than the model trained on all categories (middle).

G. Additional Completion Figures

Figure 13 shows additional completion comparisons for the experiment described in Section 4.2.

H. Per-Category ShapeNet Results

Tables 8 and 9 report the per-category reconstruction and completion results respectively for the experiments described in Sections 4.1 and 4.2.

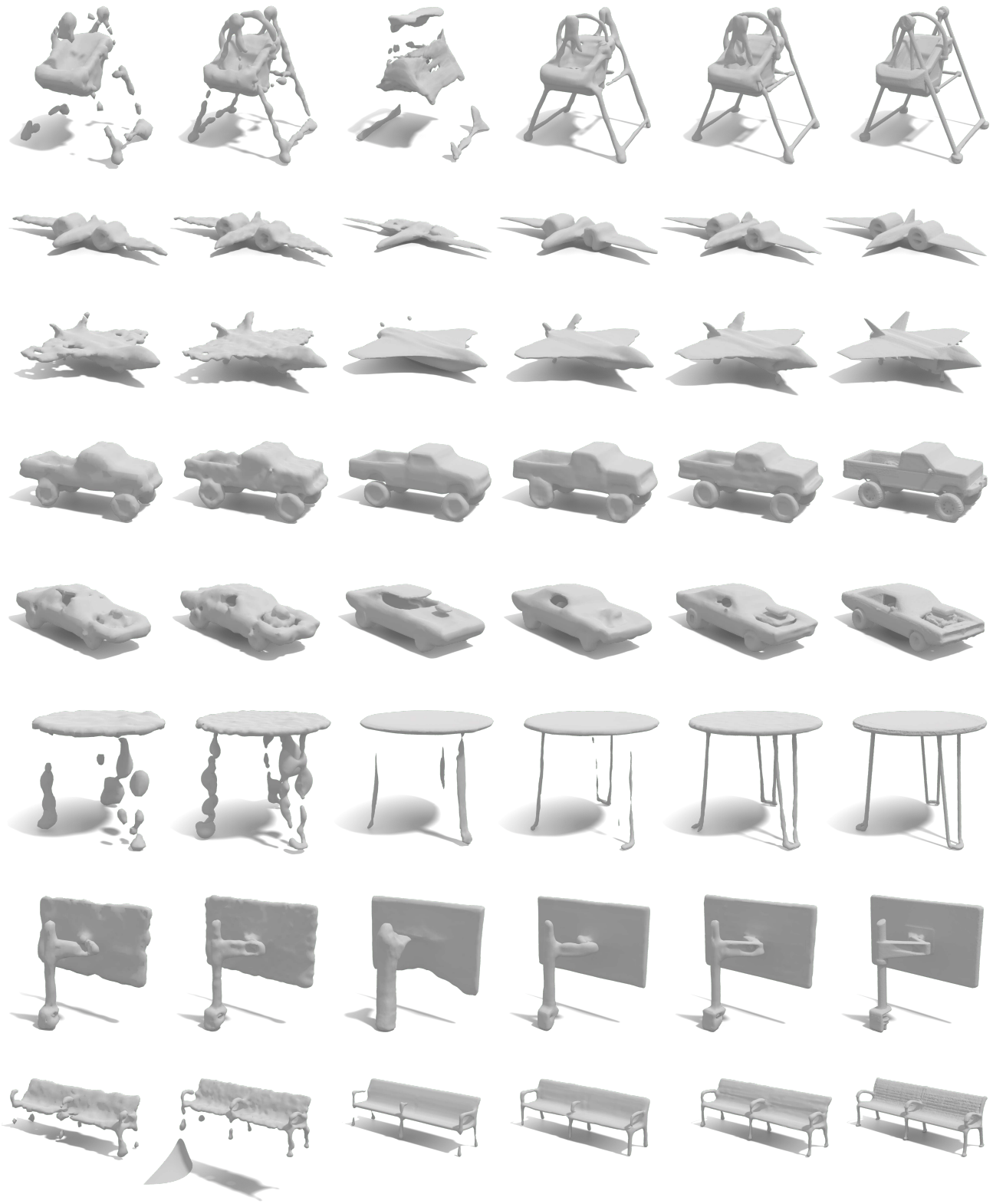
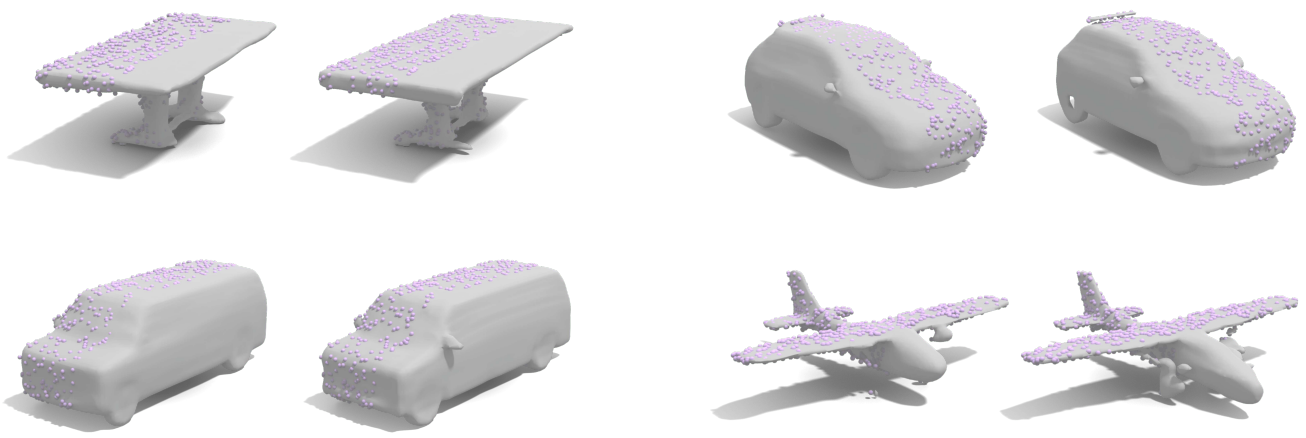


Figure 12: **ShapeNet [5] Reconstruction**. Reconstructions of models from the ShapeNet test set given 1000 input points and normals with Gaussian noise.



C-OccNet [48] **Ours** C-OccNet [48] **Ours**

Figure 13: **Shape completion on ShapeNet [5].**

Noise-Free

	IoU ↑				Chamfer ↓				Normal C. ↑			
	OccNet	C-OccNet*	NS	Ours	OccNet	C-OccNet*	NS	Ours	OccNet	C-OccNet*	NS	Ours
airplane	0.752	0.811	0.775	**0.951**	0.054	0.036	0.103	**0.016**	0.900	0.927	0.898	**0.962**
bench	0.713	0.723	0.768	**0.908**	0.052	0.045	0.065	**0.022**	0.889	0.900	0.901	**0.940**
cabinet	0.869	0.898	0.921	**0.968**	0.060	0.049	0.041	**0.028**	0.931	0.950	0.939	**0.962**
car	0.841	0.873	0.911	**0.937**	0.069	0.051	0.037	**0.030**	0.896	0.898	0.903	**0.913**
chair	0.740	0.811	0.858	**0.943**	0.076	0.051	0.045	**0.027**	0.896	0.933	0.933	**0.960**
display	0.825	0.854	0.938	**0.976**	0.062	0.048	0.030	**0.023**	0.932	0.960	0.964	**0.978**
lamp	0.550	0.751	0.834	**0.920**	0.144	0.058	0.047	**0.024**	0.819	0.902	0.915	**0.940**
loudspeaker	0.833	0.892	0.938	**0.965**	0.090	0.059	0.041	**0.033**	0.910	0.938	0.945	**0.952**
rifle	0.678	0.757	0.936	**0.957**	0.057	0.038	0.021	**0.012**	0.860	0.915	0.960	**0.970**
sofa	0.876	0.893	0.927	**0.974**	0.055	0.047	0.041	**0.024**	0.939	0.952	0.949	**0.969**
table	0.768	0.785	0.801	**0.951**	0.059	0.048	0.065	**0.025**	0.923	0.948	0.926	**0.969**
telephone	0.915	0.904	0.969	**0.988**	0.035	0.035	0.021	**0.017**	0.973	0.979	0.983	**0.988**
watercraft	0.737	0.825	0.894	**0.955**	0.083	0.046	0.044	**0.019**	0.870	0.909	0.930	**0.950**
mean	0.773	0.823	0.864	**0.949**	0.068	0.048	0.051	**0.024**	0.902	0.928	0.926	**0.954**

0.0025 Noise

	IoU ↑				Chamfer ↓				Normal C. ↑			
	OccNet	C-OccNet*	NS	Ours	OccNet	C-OccNet*	NS	Ours	OccNet	C-OccNet*	NS	Ours
airplane	0.739	0.825	0.729	**0.905**	0.057	0.034	0.103	**0.020**	0.904	0.928	0.888	**0.953**
bench	0.713	0.758	0.723	**0.867**	0.053	0.040	0.068	**0.025**	0.889	0.906	0.892	**0.935**
cabinet	0.871	0.916	0.905	**0.952**	0.061	0.044	0.045	**0.031**	0.933	0.953	0.934	**0.959**
car	0.839	0.877	0.892	**0.921**	0.068	0.052	0.041	**0.033**	0.895	0.902	0.896	**0.911**
chair	0.740	0.837	0.825	**0.912**	0.077	0.045	0.050	**0.030**	0.896	0.937	0.926	**0.956**
display	0.818	0.890	0.902	**0.953**	0.063	0.039	0.036	**0.026**	0.932	0.963	0.958	**0.975**
lamp	0.547	0.774	0.784	**0.880**	0.153	0.050	0.053	**0.026**	0.824	0.907	0.906	**0.936**
loudspeaker	0.829	0.910	0.922	**0.952**	0.091	0.052	0.046	**0.035**	0.912	0.943	0.940	**0.952**
rifle	0.678	0.783	0.860	**0.904**	0.058	0.033	0.023	**0.016**	0.865	0.919	0.947	**0.960**
sofa	0.879	0.913	0.905	**0.956**	0.055	0.041	0.047	**0.028**	0.937	0.956	0.942	**0.966**
table	0.768	0.832	0.772	**0.917**	0.059	0.040	0.065	**0.028**	0.924	0.953	0.922	**0.966**
telephone	0.909	0.931	0.932	**0.969**	0.036	0.029	0.027	**0.020**	0.973	0.980	0.975	**0.986**
watercraft	0.732	0.843	0.857	**0.926**	0.086	0.041	0.050	**0.022**	0.874	0.913	0.918	**0.945**
mean	0.771	0.847	0.831	**0.919**	0.069	0.043	0.054	**0.027**	0.903	0.932	0.919	**0.945**

0.005 Noise

	IoU ↑				Chamfer ↓				Normal C. ↑			
	OccNet	C-OccNet*	NS	Ours	OccNet	C-OccNet*	NS	Ours	OccNet	C-OccNet*	NS	Ours
airplane	0.675	0.839	0.758	**0.852**	0.155	0.062	0.098	**0.053**	0.890	0.933	0.886	**0.937**
bench	0.589	0.779	0.673	**0.813**	0.160	0.073	0.161	**0.062**	0.860	0.911	0.876	**0.922**
cabinet	0.802	0.928	0.881	**0.936**	0.181	0.078	0.105	**0.070**	0.914	0.958	0.920	**0.952**
car	0.804	0.888	0.869	**0.899**	0.182	0.095	0.095	**0.077**	0.891	0.905	0.879	**0.902**
chair	0.652	0.859	0.779	**0.876**	0.217	0.081	0.119	**0.071**	0.884	0.944	0.910	**0.946**
display	0.742	0.914	0.858	**0.924**	0.170	0.067	0.091	**0.061**	0.922	0.968	0.940	**0.967**
lamp	0.478	0.796	0.701	**0.827**	0.421	0.099	0.171	**0.065**	0.802	0.914	0.868	**0.921**
loudspeaker	0.785	0.924	0.900	**0.937**	0.236	0.091	0.108	**0.080**	0.899	0.947	0.925	**0.946**
rifle	0.600	0.807	0.774	**0.850**	0.151	0.060	0.068	**0.045**	0.832	0.925	0.906	**0.943**
sofa	0.818	0.929	0.889	**0.936**	0.159	0.072	0.095	**0.065**	0.925	0.961	0.931	**0.957**
table	0.663	0.859	0.704	**0.873**	0.168	0.072	0.167	**0.066**	0.906	0.957	0.898	**0.956**
telephone	0.847	0.944	0.892	**0.945**	0.107	0.050	0.072	**0.049**	0.966	0.982	0.958	**0.980**
watercraft	0.695	0.863	0.808	**0.890**	0.216	0.074	0.147	**0.056**	0.861	0.921	0.890	**0.931**
mean	0.699	0.863	0.791	**0.883**	0.192	0.078	0.121	**0.066**	0.888	0.937	0.900	**0.939**

Table 8: Per-category ShapeNet reconstruction results corresponding to the experiment described in Section 4.1.

	IoU ↑		Chamfer ↓		Normal C. ↑		F-Score ↑	
	C-OccNet*	Ours	C-OccNet*	Ours	C-OccNet	Ours	C-OccNet*	Ours
airplane	0.800	0.844	0.048	0.054	0.926	0.919	0.921	0.916
bench	0.615	0.705	0.082	0.086	0.868	0.872	0.808	0.853
cabinet	0.834	0.881	0.079	0.067	0.924	0.918	0.784	0.872
car	0.862	0.891	0.059	0.047	0.899	0.899	0.859	0.912
chair	0.731	0.790	0.092	0.091	0.906	0.910	0.805	0.854
display	0.768	0.850	0.088	0.079	0.921	0.925	0.774	0.876
lamp	0.620	0.685	0.138	0.159	0.864	0.866	0.751	0.797
loudspeaker	0.808	0.851	0.101	0.105	0.904	0.902	0.701	0.814
rifle	0.746	0.809	0.045	0.051	0.899	0.904	0.915	0.907
sofa	0.837	0.864	0.073	0.075	0.923	0.916	0.823	0.866
table	0.730	0.777	0.075	0.089	0.925	0.911	0.851	0.863
telephone	0.886	0.906	0.046	0.048	0.964	0.958	0.920	0.922
watercraft	0.761	0.830	0.067	0.061	0.887	0.912	0.829	0.884
mean	0.770	0.819	0.075	0.077	0.909	0.907	0.837	0.875

Table 9: Per category completion results corresponding to the experiment described in Section 4.2.