SurfR: Surface Reconstruction with Multi-scale Attention

Siddhant Ranade[*]
University of Utah

Gonçalo Dias Pais[*]
Instituto Superior Técnico

Ross Tyler Whitaker
University of Utah

Jacinto C. Nascimento
Instituto Superior Técnico

Pedro Miraldo
MERL

Srikumar Ramalingam
Google Research

Abstract

We propose a fast and accurate surface reconstruction algorithm for unorganized point clouds using an implicit representation. Recent learning methods are either single-object representations with small neural models that allow for high surface details but require per-object training or generalized representations that require larger models and generalize to newer shapes but lack details, and inference is slow. We propose a new implicit representation for general 3D shapes that is faster than all the baselines at their optimum resolution, with only a marginal loss in performance compared to the state-of-the-art. We achieve the best accuracy-speed trade-off using three key contributions. Many implicit methods extract features from the point cloud to classify whether a query point is inside or outside the object. First, to speed up the reconstruction, we show that this feature extraction does not need to use the query point at an early stage (lazy query). Second, we use a parallel multi-scale grid representation to develop robust features for different noise levels and input resolutions. Finally, we show that attention across scales can provide improved reconstruction results. The code will be made available.
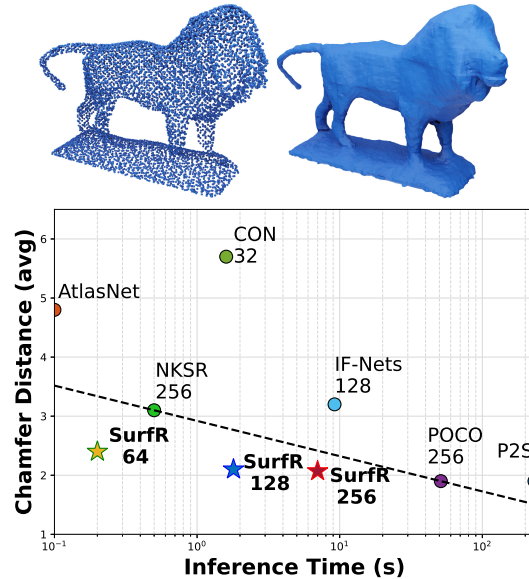
Figure 1. **SurfR result and error/time trade-off**: At the top, we show a sparse input point cloud and its reconstruction using the proposed method **SurfR**. At the bottom, we show our method and the baselines **error vs. time trade-off** comparison, with different resolutions (number below).

1. Introduction

Surface reconstruction from noisy point clouds (e.g., from structure-from-motion pipelines or 3D sensors) has been one of the essential tasks in the computer vision and graphics community [3, 22, 25, 26], with numerous applications in autonomous vehicles, medical imaging, and AR/VR technologies. The challenges arise from noisy/incomplete data, the choice of representation, complex geometry, and computational bottlenecks. While 3D point-based representations are lightweight, they lack surface information, and voxels, on the other hand, leave a huge memory footprint. We propose **SurfR**, a data-driven and efficient implicit surface reconstruction method that produces high-fidelity mod-

els, striking the right balance between speed and accuracy (see Fig. 1). Note that applications that require real-time processing or have low compute budgets will benefit immensely from this speedup, including those in robotics, augmented/virtual reality, and embedded systems, for example, in interactive mesh editing, such as [45, 53], where re-meshing is frequently re-run, or interactive SLAM [33, 42], where the user accompanied by a robot interacts with the environment to get accurate object representations for scene understanding [16, 34]. SurfR proposes a framework that is towards real-time without compromising surface details and without over-smoothing.

A robust surface reconstruction method should be capable of handling noisy/partial point clouds sampled from a

---

[*]Equal contribution.

variety of shapes. Neural implicit functions show promising results on various 3D tasks, including surface reconstruction. Many of these methods can be divided as per-object or general shape reconstruction. More object-specific methods [13, 46], inspired by [32] and [37], take advantage of the object's shape priors to encode the objects explicitly. However, these methods are not generalizable to unseen shapes. On the other hand, general shape reconstruction methods [8, 18, 35, 37, 46, 52] rely on general encoders to reconstruct the 3D implicit surface. These methods are more generalizable but suffer from a loss in reconstruction quality. The general idea is to use a neural network to predict a distance value given a query point based on a feature representation. One can achieve this by first extracting a global feature (e.g., using a PointNet [39]) and then regressing the distance function for the query point [35]. However, using a single global feature for the entire point cloud fails to capture the local geometry, and recent methods extract multiple features from local patches [2, 8, 11]. The proposed algorithm also uses multiple resolution features to better model the local and global information to achieve high-fidelity implicit representation. Our contributions are:

- Lazy query processing in feature extraction to achieve significant computational efficiency.
- Parallel multi-scale feature extraction to achieve robustness to noise and different input resolutions.
- Attention using a transformer encoder layer [48] to fuse features across scales, significantly improving reconstruction results.
- Best accuracy-speed trade-off. SurfR is competitive with state-of-the-art reconstruction fidelity and is faster than baselines at their optimal resolution.

2. Related Work

Surface reconstruction spans several decades of research in geometry processing, addressing a challenging and long-standing problem. We can study these methods across different dimensions: generalizable (single model for multiple objects), feature extraction (single point feature or multi-scale features), method architecture (query feature regression or learned kernels), etc.

Generability: Learning-based approaches for surface reconstruction take advantage of rich data priors to better handle noise, incomplete data, and inconsistent normals, which can be separated into object-specific and generalizable methods. Object-specific methods [4, 9, 10, 13, 14, 21, 27, 29–32, 35, 37, 41, 43, 44, 46, 47, 49] focus on a single object/scene to create an implicit surface representation from the input point cloud that is decoded when necessary. These methods can lead to high-quality reconstructions, though they are limited in generalizing newer shapes and geometry. In contrast, generalizable methods

such as [2, 7, 8, 11, 18, 35, 37, 50–52] can generalize to a wide range of shapes, with a penalty in the reconstruction quality. These typically differ in how the query representation is built. The proposed method follows the latter.

Feature Extraction: In general, all learning methods have a similar structure. A feature representation is built from the input point cloud and then decoded to get its implicit surface. In the object-centric methods, this feature representation is associated with an explicit per-object geometric structure [13, 37, 46] known *a priori* since the object's geometry is known. The generalizable methods build that representation based on the query object and then decode it to obtain the surface representation. Some methods rely only on a single point feature [2, 11, 35, 50] while others use multiple scale features [18, 51, 52]. Multi-scale features have been shown to provide benefits in dealing with noise and different input resolutions for different tasks [7, 8, 18, 35]. In IF-Nets [7, 8], a 3D convolutional network (with pooling) processes a discretized point cloud to create feature grids at multiple scales, with the early stages capturing shape detail and the later stages the global structure. NKSR [18] uses a sparse UNet to obtain per-resolution features. In contrast, SurfR uses a PointNet for each scale to compute cell features in parallel, considering only points within the cell without discretization. Cell embedding is sensitive to local detail at larger scales, giving us a multi-scale 3D representation. The cross-scale attention balances feature extraction between scales. The contributions are this way of extracting multi-scale features and fusing them with attention. Also, avoiding 3D convolutions and discretization makes SurfR faster and more accurate.

Architecture: These neural models can be constructed in two ways: estimate the implicit representation from anchors following the estimated kernels or regression from an explicit feature representation. Kernel methods like [18, 51, 52] create a grid kernel from the input point cloud that predicts the implicit representation. The output representation of the input points is the sum of all neighbor grid voxels output weighted by the learned kernel. In [52], NKS proposes a single-scale kernel to predict the grid occupancy while considering noise. In [18], NKSR extends NKS for multi-resolution kernels conditioned by UNet features.

On the other hand, the regression methods [2, 7, 8, 11, 35, 50] create an intermediate feature representation and then decode it to obtain the surface representation. Using a sparse convolutional backbone, POCO [2] encodes the input points in feature vectors. Given an arbitrary query point, its feature is computed from the nearest input points' features. POCO proposes to use an attention module to weigh the contribution of each neighbor feature to the query point. ALTO [50] extends POCO's approach and builds successive triplanes [1, 5] to query with attention to the points' location for better feature extraction. In contrast, the proposed
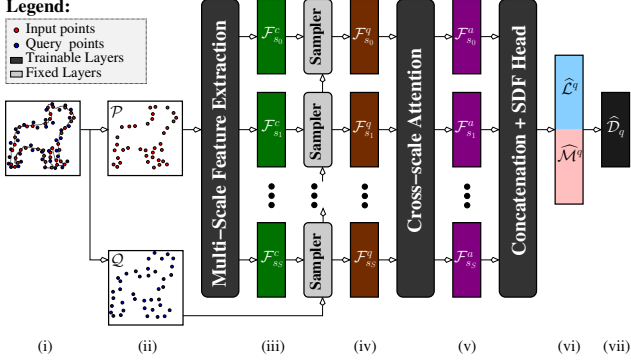
Figure 2. **Overview** The input consists of point cloud $\mathcal{P}$ and query points $\mathcal{Q}$. For each scale $s \in \{s_0, s_1, \ldots, s_S\}$, an encoder network is used to extract per-cell features $\mathcal{F}_s^c$. These features are sampled at the query points to get query point features $\mathcal{F}_s^q$ with our novel query-feature sampling technique. A self-attention mechanism is applied to these features to obtain a new set of per-query features $\mathcal{F}_s^a$, which are subsequently concatenated and regressed by the SDF head to sign logits $\hat{\mathcal{L}}^q$ and magnitudes $\hat{\mathcal{M}}^q$, which are used to compute the final signed distance $\hat{\mathcal{D}}_q$.

Figure 3. **Parallel Multi-scale feature extraction** The input points $\mathcal{P}$ in (i) are transformed to cell coordinates $\mathcal{P}_s$, at each scale s (ii), and each individual cell is processed in parallel as shown in the inset. Then, the points are encoded by PointNet e_s^l to get the per-point features $\mathcal{F}_s^l$ (iii). Further they are encoded by the cell encoder e_s^c and pooled cell-wise to get the per-cell features $\mathcal{F}_s^c$ (iv).

method also encodes the input points in feature vectors but uses a PointNet [39] for each resolution and utilizes an attention layer per resolution. The feature of the query point is constructed from the different resolutions weighted by the inverse distance of the query point to the input points. This scheme allows cross-scale information sharing without over-smoothing and at the cost of details.

3. Surface Reconstruction with Multi-scale Attention

Given a set of unorganized points $\mathcal{P}$, the goal of SurfR is to reconstruct the original object surface using implicit surface reconstruction with an SDF. Then, we define the surface $\mathcal{S}$ as the zero level-set of its SDF $d_{\mathcal{S}}(\mathbf{x})$, i.e., $\mathcal{S} = \{\mathbf{x}, d_{\mathcal{S}}(\mathbf{x}) = 0\}$, where $\mathbf{x}$ is a point in 3D. We are essentially learning the implicit function using a neural network in order to produce SDF estimates. Similar to other implicit neural methods, we estimate the SDF on a regular grid of desired resolution, and a suitable isosurfacing algorithm is necessary to recover the underlying surface [28]. The overview of the method is shown in Fig. 2.

3.1. Overview and Notation

Overview: We highlight the five main components of SurfR in this section. First, we extract features at multiple scales from the input point cloud. The basic idea is to partition the space into cells and then extract features from the points lying in each of these cells. Second, we lazily bring the query points and use the proposed query feature sampler to extract query features. Third, a cross-scale attention is
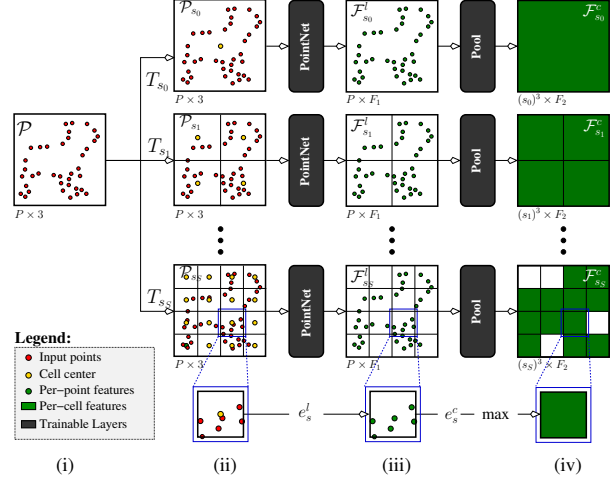
applied to these features using a transformer layer. Next, we concatenate the features across scales and use the SDF head to regress the signed distance. Finally, we use the loss functions on the sign and magnitude, and an L1 penalty on weights for regularization. In the subsections that follow, we discuss the individual components of the algorithm in detail.

Input and groundtruth: The proposed method takes as input the point cloud $\mathcal{P} = \{\mathbf{p}_i : i \in \{0, \ldots, P-1\}\}$ and the query points $\mathcal{Q} = \{\mathbf{q}_j : j \in \{0, \ldots, Q-1\}\}$, where $\mathbf{p}_i, \mathbf{q}_j \in \mathbb{R}^3$, shown in Fig. 2 (i) and (ii). For training, we use the ground-truth signed distances (i.e. to the mesh) $\mathcal{D}^q = \{d_j^q : j \in \{0, \ldots, Q-1\}\}$ corresponding to each query point. All input and query points are normalized to lie inside the unit cube $[-1, 1]^3$.

Features: We extract features at different stages of the algorithm, and we briefly introduce the notation here. For each scale $s \in \{s_0, s_1, \ldots, s_S\}$, we extract per-cell features $\mathcal{F}_s^c$, where the superscript c denotes the cell and the subscript s denotes the scale. The query features are denoted by $\mathcal{F}_s^q$. The features generated through cross-scale attention are denoted by $\mathcal{F}_s^a$. The scale s is usually used as the subscript in different entities that follow.

Encoders: The local and cell encoders at scale s are denoted by e_s^l and e_s^c, and the attention across scales by e^a.

3.2. Parallel Multi-scale Feature Extraction

Figure 3 presents the proposed approach to multi-scale feature extraction from a given point cloud. At each scale, the
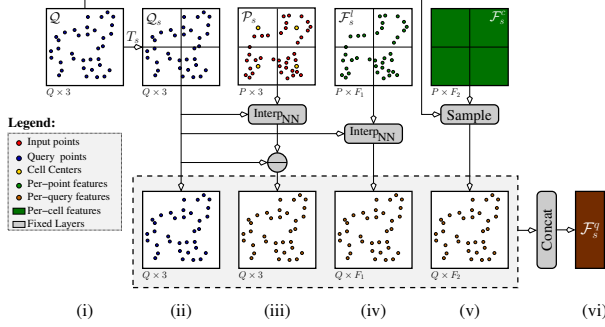
Figure 4. **Query Feature Sampling** The input to this stage consists of $\mathcal{P}_s$, $\mathcal{F}_s^l$, $\mathcal{F}_s^c$ from the multi-scale feature extraction, and the query points $\mathcal{Q}$. The query points *ii* are transformed to cell coordinates $\mathcal{Q}_s$ (*ii* - top). The output per-query feature $\mathcal{F}_s^q$ (*vi*) is a concatenation of the query points in cell coordinates (*ii* - bottom), mean relative nearest neighbor position (*iii*), mean nearest neighbor feature (*iv*), and feature of the cell containing the query point (*v*).

input point set is spatially partitioned into grid cells, and points within each cell are processed independently of other cells, as shown in the inset. At a larger scale, we partition the space into a larger number of cells and vice versa. Before extracting features from the points falling in each of these cells, we transform the points to the local coordinate frame of the cells. As shown in Fig. 3 (ii), the points $\mathbf{x}$ at scale s are first transformed to cell coordinates using $T^s(\mathbf{x})$, which corresponds to a translation of the origin to the cell center and scaling. Then, a per-point feature (Fig. 3 (iii)) is extracted using a PointNet e_s^l. Finally, a single feature is computed using the cell encoder e_s^c followed by cell-wise max pooling. The result is shown in Fig. 3 (iv). Next, we detail these steps.

Transformation to cell coordinates $T_s(\mathbf{x})$**:** Transforming the points to cell coordinates allows the local encoder to independently estimate features across cells. At scale s, the unit cube is divided into a total of s^3 cells. $T_s(\mathbf{x})$ denotes the cell-coordinate of point $\mathbf{x}$ at scale s.

Per-point features: Let $\mathcal{P}_s = \{T_s(\mathbf{p}_i) : i \in \{0, \ldots, P - 1\}\}$ denote the set of points in local coordinates at scale s. For each point $\mathbf{p}_i \in \mathcal{P}_s$, the local encoder e_s^l is used to extract per-point features as follows:

$$\mathbf{f}_s^{l,i} = e_s^l(T_s(\mathbf{p}_i)), \tag{1}$$

which is used by the query feature sampler in computing the mean nearest neighbor feature in Sec. 3.3. We denote the set of all per-point features at scale s as $\mathcal{F}_s^l = \{\mathbf{f}_s^{l,i} : \mathbf{p}_i \in \mathcal{P}\}$.

Per-cell features: The per-point features $\mathbf{f}_s^{l,i}$ are further encoded by the cell encoder e_s^c and pooled over cells to get the per-cell features using

$$\mathbf{f}_s^c = \max_{i \in c_k} e_s^c(\mathbf{f}_s^{l,i}), \tag{2}$$

where max leads to a channel-wise maximum overall point in cell c_k. These per-cell features account for the local geometry of the cell, which is crucial for estimating the SDF. Cells that do not contain any points have the all-zero feature, shown in white in Fig. 3 (iv). The set of all per-cell features (also at scale s) are denoted as $\mathcal{F}_s^c = \{\mathbf{f}_s^c : c \in \{0, \ldots, s - 1\}^3\}$.

3.3. Query Feature Sampling

Both local and global shape information are critical in estimating the SDF at a given point. Methods that use a single global feature descriptor, e.g. [14, 35], do not perform as well as methods that take the local shape into account. One way to incorporate the local shape is to use patches around query points as input. However, this is expensive. The proposed architecture addresses this issue by first extracting shape features independently of the query points and then *samples* these features at the query points. At each scale, the set of per-point features $\mathcal{F}_s^l$, (as shown in Eq. 1), and per-cell features $\mathcal{F}_s^c$ (see Eq. 2) are combined to obtain features $\mathcal{F}_s^q$ for each query point, as shown in Fig. 4 (Page 4). The feature $\mathbf{f}_s^{q,j}$ corresponding to a query point $\mathbf{q}_j$ lying in cell s is defined as the concatenation of:

- **Query point position**: $T_s(\mathbf{q}_j)$.
- **Cell feature**: $\mathbf{f}_s^c$.
- **Mean nearest neighbor feature** Given $\mathcal{N}_j \subseteq \{i : \mathbf{p}_i \in \mathcal{P}_s^c\}$, the set of K nearest neighbors of $\mathbf{q}_j$, the mean nearest neighbor feature is given by:

$$\mathbf{f}_s^{\mathcal{N}_j} = \sum_{i \in \mathcal{N}_j} w_s^{j,i} \mathbf{f}_s^{l,i}, \tag{3}$$

where the weights $w_s^{j,i} = \frac{\text{sim}(i,j)}{\sum_{l \in \mathcal{N}_j} \text{sim}(l,j)}$, $\sum_i w_s^{j,i} = 1$ (**InterpNN**), and $\text{sim}(i,j) = \frac{1}{\|\mathbf{q}_j - \mathbf{p}_i\|^2}$, similar to [40].

- **Mean relative nearest neighbor position** a weighted average of the relative positions of the K nearest neighbors in cell coordinates, i.e.,

$$\bar{\mathbf{p}}_s^{\mathcal{N}_j} = s \sum_{i \in \mathcal{N}_j} w_s^{j,i}(\mathbf{p}_i - \mathbf{q}_j). \tag{4}$$

The final concatenated feature of $\mathbf{q}_j$ in cell c is given by

$$\mathbf{f}_s^{q,j} = \left[T_s(\mathbf{q}_j), \ \mathbf{f}_s^c, \ \mathbf{f}_s^{\mathcal{N}_j}, \ \bar{\mathbf{p}}_s^{\mathcal{N}_j} \right]. \tag{5}$$

The set of all per-query features at scale s is denoted as $\mathcal{F}_s^q = \{\mathbf{f}_s^{q,j} : j \in \{0, \ldots, Q - 1\}\}$.

3.4. Cross-Scale Attention

The per-query features from different scales contain complementary information. Therefore, we adopt a weighted sum across all scales to emphasize the more significant features from different scales and improve feature extraction.

For that purpose, a cross-scale attention mechanism is included after the query feature sampling. For each query point, we apply a transformer [48] encoder layer to fuse the features across levels, here denoted as e^a. Given the query features $\mathbf{f}_s^{q,j}, \ldots, \mathbf{f}_{s_S}^{q,j}$ as inputs (shown in Eq. 5), the attention mechanism outputs a new set of features $\mathcal{F}_s^a = \{\mathbf{f}_{s_0}^{a,j}, \ldots, \mathbf{f}_{s_S}^{a,j}\}$ as follows:

$$\mathbf{f}_{s,j}^a = e^a(\mathbf{f}_s^{q,j}, \{\mathbf{f}_{s_0}^{q,j}, \ldots, \mathbf{f}_{s_S}^{q,j}\}). \tag{6}$$

These features are shown in Fig. 2 (v).

As will be shown in Sec. 4.3, we find that this simple self-attention mechanism across scales improves the performance of the method. In particular, it helps reduce the artifacts of the cell boundaries and produces smoother results overall.

3.5. Signed Distance Regression

The set of features $\mathcal{F}_s^a$, in Eq. 6, corresponding to query points $\mathbf{q}_j$ are concatenated across scales and regressed by the SDF head e^h to sign logits and magnitudes of the signed distance, denoted by $\widehat{l}_j^q$ and $\widehat{m}_j^q$, respectively (see Fig. 2(vi)), as

$$(\widehat{l}_j^q, \widehat{m}_j^q) = e^h([\mathbf{f}_{s_0}^{a,j}, \ldots, \mathbf{f}_{s_S}^{a,j}]).$$

The final estimate of the SDF is computed as $\widehat{d}_j^q = \text{sgn}(\widehat{l}_j^q) \cdot \widehat{m}_j^q$, shown in Fig. 2(vii).

3.6. Loss Functions

The loss function minimizes the error on the magnitude and sign of the SDF and uses L1 penalty on the weights for regularisation (similar to [11]):

$$L = \lambda_{\text{mag}} L_{\text{mag}} + \lambda_{\text{sgn}} L_{\text{sgn}} + \lambda_{\text{reg}} L_{\text{reg}} \tag{7}$$

The term L_{mag} denotes the error on the estimated magnitude with respect to the ground-truth as given by $L_{\text{mag}} = \sum_j |\tanh(\widehat{m}_j^q) - \tanh(\text{abs}(d_j^q))|$. The sign term L_{sgn} is the binary cross-entropy on the sign logits. The term L_{reg} is an L_1 regularizer on the weights of e^h. The coefficients λ_{mag}, λ_{sgn}, and λ_{reg} denote the tunable hyperparameters.

3.7. Surface Reconstruction

Given a new point cloud as input, we estimate the SDF at points on a regular grid and then use marching cubes to extract the (iso-surface). Estimating the SDF at all the points on a regular grid is prohibitively expensive for any reasonable voxel grid resolution, and instead, one can argue that a truncated SDF is sufficient for surface extraction. As in [11], we evaluate the SDF close to the input points (leaving other samples empty), and propagate the sign by repeatedly applying a box filter of size ϵ^3 voxels at the empty voxels until convergence, updating the sign only if the filter response is greater (in magnitude) than a user-defined threshold $t_{\text{update}} = 13$. Since ϵ directly affects reconstruction time, we use the same value, $\epsilon = 5$.

4. Experiments

4.1. Datasets

The training and validation data consist of 4950 and 100 examples from the ABC dataset [24], a collection of CAD models. The test set comprises 100 additional examples from the ABC dataset (unseen in training), the FAMOUS dataset (a collection of 22 meshes popular in the geometry processing literature, such as the Utah teapot, the Stanford bunny, and the Dragon), and the Thingi10K dataset [54], with varying levels of noise and sparsity (for Thingi10K and FAMOUS). For a fair comparison, we highlight that the test data was not seen during training or validation for any of the tested methods in Sec. 4.4. We also show qualitative results on two scans of complex real-world objects.

All meshes are first transformed to the unit cube, following which the input point cloud is sampled using BlenSor [15], which simulates scanning of a given surface mesh with a time-of-flight sensor, including realistic noise characteristics and sensor artifacts. For each shape, 2000 query points are used, 1000 samples on the surface and offset along the surface normal by a distance uniformly sampled from [-0.02, 0.02], and 1000 sampled uniformly at random in the unit cube. While training, 6000 input points and 1000 query points were used per shape. During training, random rotations are applied to all shapes.

4.2. Implementation Details

Network Architectures: The local encoder e_s^l is implemented as a 2-layer MLP followed by a spatial transformer [20]. We pick the per-point feature size to be $F_1 = 64$. The cell encoder e_s^c is a 3-layer MLP, with the per-cell feature size $F_2 = 128$. The query feature dimension (at each scale) is thus $F = 3 + 3 + F_1 + F_2 = 198$ (see Fig. 4). We use scales $s \in \{1, 4, 16\}$, $S = 3$. Together, e_s^l and e_s^c (Eqs. 1 and 2) resemble a PointNet [39]. The cross-scale attention e^a (Eq. 6) is composed of one head transformer encoder layer with a 512 fully connected layer. The SDF head e^h has an input dimension of $S \times F = 594$ and is a 5-layer MLP with an abs($\cdot$) non-linearity for the magnitude. All MLPs use BatchNorm [19] followed by ReLU non-linearity for hidden layers.

Training Details: The loss weights are $\lambda_{\text{mag}} = 5.0$, $\lambda_{\text{sgn}} = 2.0$, $\lambda_{\text{reg}} = 10^{-6}$, and the batch size is 16. We use the Adam [23] optimizer with an initial learning rate of $7.5 * 10^{-4}$, which decays exponentially after every epoch, halving every 100 epochs, and train for a total of 750 epochs.

Software/Hardware: Our code is written in PyTorch [36] with the PyTorch-Geometric [12] extension library. We take utmost care to ensure our experiments are reproducible; We use one NVidia Titan Xp GPU for training, taking about 20

	Chamfer Distance L_2 ($\downarrow$)		
Noise	Base	+ multi-scale	+ multi-scale + attention
no	2.6	2.3	**2.2**
med	3.0	2.7	**2.4**
max	3.6	3.7	**3.2**
Avg.	3.1	2.8	**2.6**

Table 1. **Ablations**: Chamfer distance for: **Base**: single LOD of value 1, **Base + multi-scale**: multi-scales, with values $\{1, 4, 16\}$, without attention, **Base + multi-scale + attn.**: multi-scales $\{1, 4, 16\}$ with cross-scale attention.

Table 2. **Comparisons on the Different Multi-scales and Weighting Schemes**: Chamfer distance on different scales with cross-scale attention (lower is better). We also evaluate the different weighting schemes for grid cell aggregation. **EW** samples equal weights and **LW** is a learnt weight parameter.

Noise	InterpNN				LW	EW
	$\{1, 4\}$	$\{1, 4, 16\}$	$\{1, 5, 25\}$	$\{1, 3, 9, 27\}$	$\{1, 4, 16\}$	$\{1, 4, 16\}$
no	3.3	**2.2**	2.6	3.3	2.5	2.5
med	2.8	**2.4**	2.7	3.2	2.6	2.8
max	3.6	**3.2**	3.5	3.9	3.4	3.6
Avg.	3.3	**2.6**	2.9	3.5	2.8	3.0

minutes per epoch.

4.3. Ablation Study

Using a surface reconstruction of voxel grid of dimension 256, we evaluate our design choices related to multi-scale feature extraction compared to using a single scale of 1, and query feature components by comparing their performance. We report the chamfer distance on the ABC dataset in Tab. 1. This analysis validates the design of the query feature. Due to resource constraints, these ablations were performed with a batch size of 4, training for 100 epochs.

In addition to the choices related to multi-scale grid feature extraction and cross-scale attention, we study the effect of using different scales and weighting schemes for query feature sampling. Table 2 shows the results. We tried with equal **EW** and learned weights **LW** and InterpNN (see Sec. 3.3) which achieves the best results. The final method uses scales $\{1, 4, 16\}$, which obtained the best average performance in all ablations. The multi-scale features allow for a better global understanding of the object, while the cross-scale attention parallel module fuses the global and local features, producing more detail.

4.4. Results

Baselines: We compare the proposed approach with the data-driven methods P2S [11], IF-Net [8], CON [37], SAP [38], POCO [2], and NKSR [18]. We follow the same training protocol as POCO and P2S. We note that previous methods obtained the best results at different grid resolutions. While CON uses a voxel grid of size 32, IF-Net uses 128, P2S 256, SAP 256, POCO 256, and NKSR 256. CON and IF-Net are trained on ≈ 40000 models across 13 categories from ShapeNet [6], while P2S, POCO, NKSR, and SurfR are trained on ≈ 5000 shapes from the ABC dataset. All the baselines except NKSR only require the input point cloud to obtain the surface representation. Since NKSR also requires normals as input, we use the evaluation point cloud to estimate the normals with 16 nearest neighbors as a fair comparison to the other baselines. Our evaluation scheme does not use additional refinement steps such as POCO and NKSR as detailed in Sec. 3.7.

Quantitative results: We report the chamfer distance (multiplied by 100) and normal consistency [30] on the test datasets (see Sec. 4.1) in Tab. 3 at Page 8. The chamfer distance measures the geometric quality of the reconstructed surface, penalizing missing and extra geometry, whereas normal consistency penalizes surface roughness and over-smoothing. For a fair comparison, we consider SurfR with three different surface reconstruction resolutions: 64, 128, and 256. The results show that SurfR is competitive with the current state-of-the-art at a speed-up. Considering all methods with the same resolution, SurfR is 33 times faster than P2S and 7.3 times faster than POCO (both at 256 resolution), as shown in Tab. 4 and Fig. 1. The results of no-noise, med-noise, or dense reconstruction are very competitive with those of P2S and POCO; for example, in Thingi10k, SurfR achieves about the same reconstruction results as P2S and POCO. The main difference is in the max-noise case, where the proposed method obtains slightly worse results than both P2S and POCO and where CON, IF-Net, and NKSR fail as well since they are constrained to 3D convolutions and the kernel for NKSR. IF-Net creates holes in the generated meshes, explaining the lower normal consistency. CON produces smoothed meshes at optimal 32 resolution, losing fine detail and resulting in a high chamfer distance, though its overly smoothed surfaces achieve a high normal consistency. NKSR produces many holes that, as a consequence, obtain low normal consistency and high chamfer distance. Additionally, SurfR is faster and more accurate than IF-Net, CON, and NKSR while running at the same or higher resolution (CON 32, IF-Net 128, NKSR 256). SurfR, P2S, and POCO perform better at filtering noise and preserving fine detail, corroborated by the low chamfer distance and high normal consistency.

Qualitative results: We visualize some meshes reconstructed at a resolution of 256 in Fig. 5 with different noise levels for the input point cloud. Finally, we show reconstructions of two real-world objects in Fig. 6. Qualitatively, SurfR's results are better than most baselines, preserving more detail and completing sparsely sampled surfaces better. The reconstructed surfaces are rougher than POCO's, but the object details are still captured. The other methods over-smooth the surface, losing detail and producing holes.
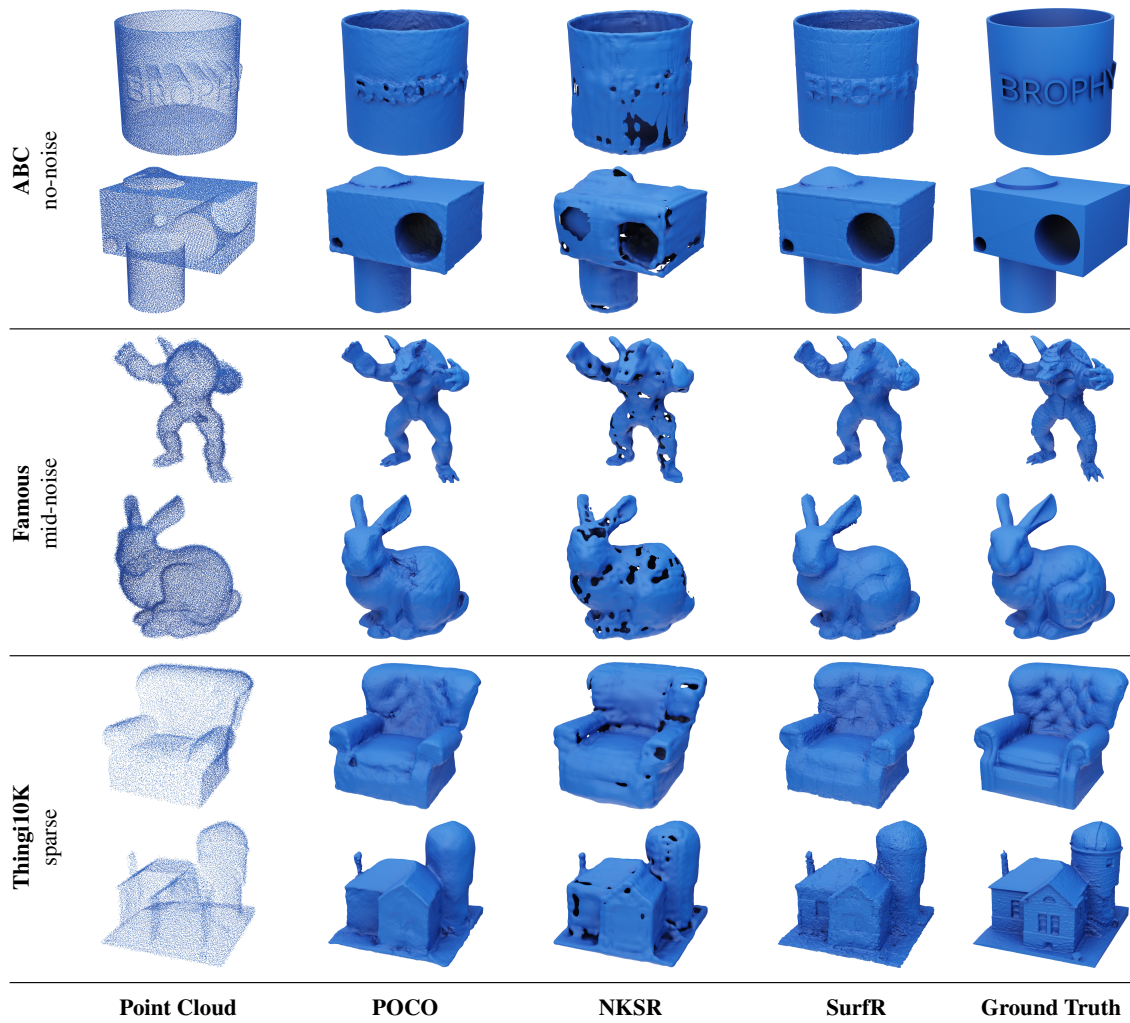
Figure 5. **Qualitative Baseline Comparison**: the results of the proposed method are qualitatively on par with the baselines, or even better, particularly on the Famous and Thingi10K datasets, with different noise/density levels. **SurfR** preserves more detail even in sparsely sampled surfaces.
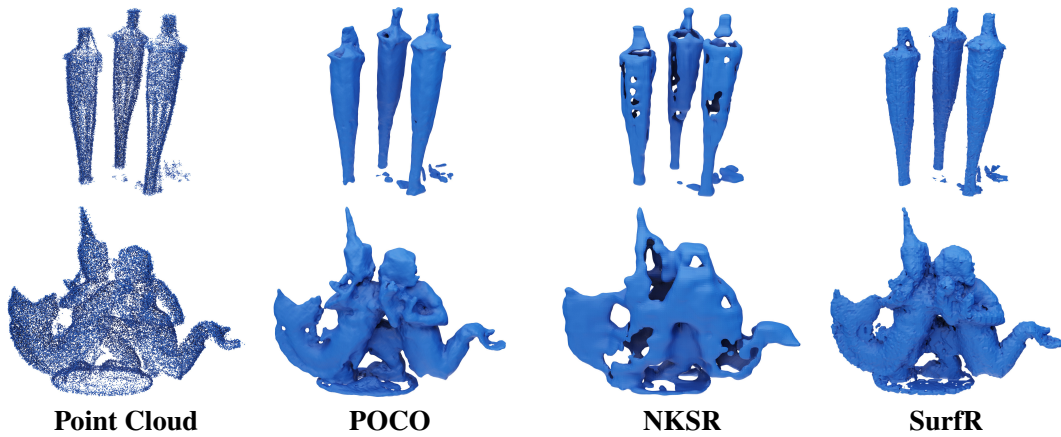

Figure 6. **Reconstruction of Real-world Objects**: SurfR generalizes well, even in noise and variable sampling density. The proposed method shows it can handle real-world objects taken from SfM methods.

Table 3. **Quantitative Baseline Comparison**: Chamfer distance between the reconstructed and ground truth meshes (lower is better) and normal consistency measuring the accuracy of the shape normals (higher is better).

Noise & Sparsity	Chamfer Distance L_2 (↓)									Normal consistency (↑)						
	CON [37]	P2S [17]	SAP [38]	IF [8]	POCO [2]	NKSR [18]	SurfR 64	128	256	CON [37]	IF [8]	POCO [2]	NKSR [18]	SurfR 64	128	256
ABC dataset [24] (unseen in training)																
no-noise	2.6	1.8	7.6	2.8	1.7	3.0	2.3	2.1	2.1	0.89	0.62	0.89	0.29	0.87	0.89	0.89
med-noise	7.7	2.1	8.1	3.8	2.1	3.3	2.5	2.4	2.4	0.8	0.61	0.84	0.33	0.81	0,82	0.80
max-noise	14.0	2.8	7.2	5.9	2.7	4.1	3.3	3.2	3.1	0.74	0.57	0.75	0.4	0.78	0.75	0.71
Famous dataset (a collection of 22 popular meshes)																
no-noise	2.4	1.4	9.0	2.2	1.4	2.7	1.8	1.5	1.4	0.84	0.63	0.78	0.18	0.82	0.86	0.86
med-noise	3.3	1.5	8.8	2.3	1.7	2.7	1.9	1.7	1.6	0.83	0.63	0.73	0.17	0.81	0.83	0.81
max-noise	12.8	2.5	7.0	5.2	2.9	3.9	3.3	3.1	3.0	0.7	0.56	0.55	0.26	0.72	0.70	0.68
sparse	3.2	1.9	10.4	2.6	2.0	3.2	2.4	2.1	2.1	0.81	0.63	0.67	0.18	0.77	0.79	0.77
dense	3.7	1.3	7.8	2.4	1.5	2.3	1.8	1.6	1.5	0.84	0.64	0.76	0.19	0.82	0.84	0.83
Thingi10k dataset [54]																
no-noise	1.9	1.4	8.4	2.1	1.4	2.7	1.8	1.5	1.4	0.92	0.65	0.92	0.22	0.89	0.92	0.91
med-noise	3.1	1.5	8.2	2.4	1.5	2.7	1.9	1.6	1.5	0.9	0.65	0.9	0.21	0.88	0.89	0.87
max-noise	12.8	2.6	6.9	5.5	2.7	3.9	3.3	3.1	2.8	0.76	0.58	0.71	0.3	0.79	0.76	0.73
sparse	3.1	2.1	10.1	2.8	2.1	3.3	2.7	2.2	2.4	0.88	0.64	0.81	0.21	0.82	0.84	0.82
dense	3.5	1.4	7.1	2.4	1.4	2.3	1.8	1.5	1.4	0.91	0.65	0.91	0.2	0.89	0.91	0.89
Avg.	5.7	1.9	8.2	3.2	1.9	3.1	2.4	2.1	2.1	0.83	0.84	0.62	0.79	0.24	0.83	0.81

Table 4. **Inference Time Comparison in Seconds**: For the test, we compute the average 3D reconstruction time per mesh.

CON [37]	P2S [17]	IF [8]	POCO [2]	NKSR [18]	SurfR 64	128	256
2.31	232	9.2	51	0.5	0.2	1.8	7

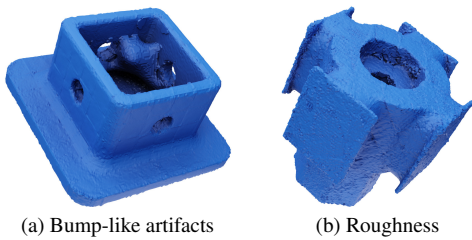(a) Bump-like artifacts   (b) Roughness

Figure 7. **Limitations**: (a) shows Bump-like surfaces and (b) Surface roughness in noisy input point cloud.

5. Limitations

To obtain faster evaluation speeds, the strategy used in this work can sometimes result in the network assigning the same sign to points on both sides of the surface, leaving it up to the sign propagation step to decide the surface boundary, which results in bump-like artifacts. Figure 7 (a) illustrates this limitation of the proposed method. Moreover, from the sparse nature of the architecture, rougher surfaces tend to appear when noise levels increase, as shown in Fig. 7 (a). Due to the convolutional layers, other methods can smooth the surface, but at the cost of losing detail.

6. Discussion

We present SurfR, a new method for accurate and efficient implicit surface reconstruction from point clouds. The key to efficiency is to show a novel query feature sampling technique that allows us to extract features without relying on the input points. Other novel contributions include techniques for using parallel multi-scale features and attention across features to improve surface reconstruction results. The proposed method achieves the right accuracy-speed trade-off. We demonstrate that this approach is faster than the current state-of-the-art at their optimal resolutions while achieving compelling results.

References

[1] Yuval Bahat, Yuxuan Zhang, Hendrik Sommerhoff, Andreas Kolb, and Felix Heide. Neural volume super-resolution. *arXiv preprint arXiv:2212.04666*, 2022. 2

[2] Alexandre Boulch and Renaud Marlet. Poco: Point convolution for surface reconstruction. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 6302–6314, 2022. 2, 6, 8

[3] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *ACM SIGGRAPH*, pages 67–76, 2001. 1

[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 Conf. Computer Vision (ECCV)*, pages 608–625, 2020. 2

[5] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16123–16133, 2022. 2

[6] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 6

[7] Julian Chibane and Gerard Pons-Moll. Implicit feature networks for texture completion from partial 3d data. In *European Conference on Computer Vision (ECCV) Workshops*, pages 717–725, 2020. 2

[8] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 6970–6981, 2020. 2, 6, 8

[9] Angela Dai and Matthias Niessner. Scan2mesh: From unstructured range scans to 3d meshes. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5569–5578, 2019. 2

[10] Angela Dai, Christian Diller, and Matthias Niessner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 846–855, 2020. 2

[11] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J. Mitra, and Michael Wimmer. Points2surf: Learning implicit surfaces from point clouds. In *European Conf. Computer Vision (ECCV)*, pages 108–124, 2020. 2, 5, 6

[12] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 5

[13] Arihant Gaur, G Dias Pais, and Pedro Miraldo. Oriented-grid encoder for 3d implicit representations. *Int'l Conf. 3D Vision (3DV)*, 2024. 2

[14] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. A papier-mache approach to learning 3d surface generation. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 216–224, 2018. 2, 4

[15] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Blensor: Blender sensor simulation toolbox. In *International Symposium on Visual Computing (ISVC)*, pages 199–208, 2011. 5

[16] David Hall, Feras Dayoub, John Skinner, Haoyang Zhang, Dimity Miller, Peter Corke, Gustavo Carneiro, Anelia Angelova, and Niko Sünderhauf. Probabilistic object detection: Definition and evaluation. In *IEEE Winter Conf. on Applications of Computer Vision (WACV)*, pages 1031–1040, 2020. 1

[17] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *ACM Transactions on Graphics (TOG)*, 39(4), 2020. 8

[18] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural kernel surface reconstruction. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 4369–4379, 2023. 2, 6, 8

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Int'l Conf. Machine learning (ICML)*, page 448–456, 2015. 5

[20] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 5

[21] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Niessner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 6000–6009, 2020. 2

[22] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):1–13, 2013. 1

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Int'l Conf. Learning Representations (ICLR)*, 2015. 5

[24] 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 *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 9593–9603, 2019. 5, 8

[25] Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Eurographics*, pages 11–21, 2004. 1

[26] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Robust and efficient surface reconstruction from range data. *Computer Graphics Forum*, 28(8):2275–2290, 2009. 1

[27] Yiyi Liao, Simon Donné, and Andreas Geiger. Deep marching cubes: learning explicit surface representations. In

IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR), pages 2916–2925, 2018. 2

[28] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH*, 21(4):163–169, 1987. 3

[29] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *ACM SIGGRAPH*, 40(4), 2021. 2

[30] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470, 2019. 6

[31] M. Meshry, D. B. Goldman, S. Khamis, H. Hoppe, R. Pandey, N. Snavely, and R. Martin-Brualla. Neural rerendering in the wild. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 6871–6880, 2019.

[32] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conf. Computer Vision (ECCV)*, pages 405–421, 2020. 2

[33] Samer Nashed and Joydeep Biswas. Human-in-the-loop slam. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 32(1), 2018. 1

[34] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters (RA-L)*, 4(1):1–8, 2018. 1

[35] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. 2, 4

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, 2019. 5

[37] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conf. Computer Vision (ECCV)*, pages 523–540, 2020. 2, 6, 8

[38] Songyou Peng, Chiyu Max Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 6, 8

[39] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 2, 3, 5

[40] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5105–5114, 2017. 4

[41] Shunsuke Saito, , Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *IEEE/CVF Int'l Conf. Computer Vision (ICCV)*, pages 2304–2314, 2019. 2

[42] Abbas Sidaoui, Mohammad Kassem Zein, Imad H. Elhajj, and Daniel Asmar. A-slam: Human in-the-loop augmented slam. In *icra*, pages 5245–5251, 2019. 1

[43] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2437–2446, 2019. 2

[44] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020. 2

[45] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rossl, and H-P Seidel. Laplacian surface editing. In *Eurographics*, pages 175–184, 2004. 1

[46] 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 *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 11358–11367, 2021. 2

[47] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7537–7547, 2020. 2

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 2, 5

[49] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *European Conf. Computer Vision (ECCV)*, pages 55–71, 2018. 2

[50] Zhen Wang, Shijie Zhou, Jeong Joon Park, Despoina Paschalidou, Suya You, Gordon Wetzstein, Leonidas Guibas, and Achuta Kadambi. Alto: Alternating latent topologies for implicit 3d reconstruction. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 259–270, 2023. 2

[51] Francis Williams, Matthew Trager, Joan Bruna, and Denis Zorin. Neural splines: Fitting 3d surfaces with infinitely-wide neural networks. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 9949–9958, 2021. 2

[52] Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural fields as learnable kernels for 3d reconstruction. In *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 18500–18510, 2022. 2

[53] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. In *ACM SIG-GRAPH*, pages 644–651, 2004. 1

[54] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 5, 8