Octree Guided Unoriented Surface Reconstruction

Chamin Hewa Koneputugodage[1] Yizhak Ben-Shabat[1,2] Stephen Gould[1]

[1]The Australian National University [2]Technion Israel Institute of Technology

{chamin.hewa, stephen.gould}@anu.edu.au sitzikbs@gmail.com

Abstract

We address the problem of surface reconstruction from unoriented point clouds. Implicit neural representations (INRs) have become popular for this task, but when information relating to the inside versus outside of a shape is not available (such as shape occupancy, signed distances or surface normal orientation) optimization relies on heuristics and regularizers to recover the surface. These methods can be slow to converge and easily get stuck in local minima. We propose a two-step approach, OG-INR, where we (1) construct a discrete octree and label what is inside and outside (2) optimize for a continuous and high-fidelity shape using an INR that is initially guided by the octree's labelling. To solve for our labelling, we propose an energy function over the discrete structure and provide an efficient move-making algorithm that explores many possible labellings. Furthermore we show that we can easily inject knowledge into the discrete octree, providing a simple way to influence the result from the continuous INR. We evaluate the effectiveness of our approach on two unoriented surface reconstruction datasets and show competitive performance compared to other unoriented, and some oriented, methods. Our results show that the exploration by the move-making algorithm avoids many of the bad local minima reached by purely gradient descent optimized methods (see Figure 1).

1. Introduction

Surface reconstruction from 3D point clouds has been studied extensively in computer vision and computer graphics. The task requires estimating a mesh of the shape's surface from a point cloud sampled from the surface of the shape. We focus on the reconstruction of watertight 3D shapes, *i.e.*, shapes that have a well defined interior and exterior. Such shapes are often represented as signed distance fields (SDFs) or occupancy fields, which can be efficiently encoded by a neural network [28, 30]. As these representations are fields parameterized by neural networks, they are often called neural fields. Furthermore, they implicitly represent the shape by a level set of the field, thus they are also often referred to as implicit neural representations (INRs).

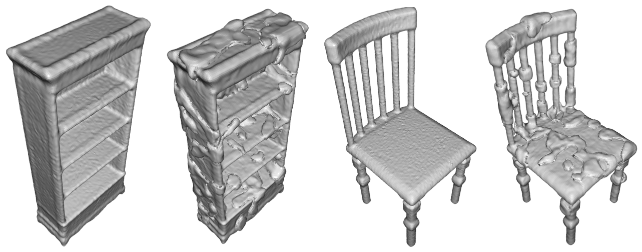
Figure 1. Our method with a SIREN INR (OG-SIREN, left) compared to SIREN wo n (right) for two shapes from the ShapeNet dataset. Our octree guidance allows for consistent inside-outside determinism. On the other hand, SIREN wo n gets stuck in local minima from which it cannot escape (due to needing to completely change the occupancy of certain areas) creating extraneous surfaces (often called ghost geometries in the literature).

The broader class of neural fields, including INRs, have been very popular over the last few years as they can handle arbitrary topology, are memory efficient, and are continuous with potentially infinite resolution [39, 46]. Among the INRs for 3D shapes, SDFs are the most popular as they provide more useful information (distances not just occupancy) and are required for downstream graphics algorithms such as sphere tracing and approximate soft shadows [29, 35].

When learning an implicit representation of a shape, a major difficulty is predicting whether points in space are inside or outside the shape. Many methods require oriented surface normals for the input points or signed distances from the surface, which usually are not given by raw data from scans. While this can be estimated using the line of sight information [17] or algorithms [4, 6, 15, 21], they yield noisy predictions that after postprocessing still can lead to bad results (see Section 4.4). We consider the task of unoriented surface reconstruction, where such information is not available, and only the sampled surface points are given. We demonstrate that our method performs competitively and sometimes better than oriented methods, even when they are given the ground truth (GT) normals.

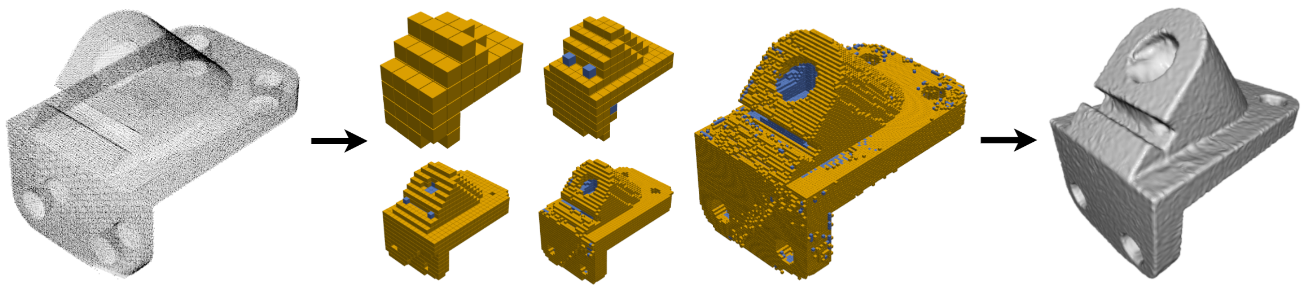
Figure 2. An illustration of our method, OG-INR. Given an unoriented point cloud (left), we progressively build and label an octree around the points (middle). The octree at depths 3-7 are shown. Surface leaves of the octree (yellow) are leaves that contain points from the point cloud, other leaves are labelled as inside (blue) or outside (transparent) by minimizing an energy function. We then train an INR model to obtain an SDF, using the labelling as supervision for the initial training, after which we can extract a mesh (right).

We propose OG-INR, which uses a discrete representation in conjunction with the continuous INR representation (see Figure 2). Given an input unoriented point cloud, we progressively build an octree from the input points and determine which leaf nodes are surface leaves (*i.e.*, leaves that contain a point). We also label all other leaves as inside or outside leaves (if they should be within or outside the shape respectively). To do this we minimize an energy function that trades off the watertight surface property, that every surface point should border with the inside and outside of the shape, with a minimal surface constraint. We then train standard INR architectures, initially guiding the training by the octree labelling, and show that the INR converges much faster and is less prone to large failure regions.

Our main contributions are:

- We introduce a novel method to guide the initial stage of INR training. It uses a labelled octree structure to allow the INR to converge significantly faster and alleviates the local minima problem of INRs.
- We propose an energy function over octree labels that captures the task of surface reconstruction. It balances the constraint of maintaining known surface regions with minimising the overall surface area.
- We provide an efficient move-making algorithm to optimize the energy function, which explores many inside-outside possibilities in a structured manner.
- Our discrete representation is easily understandable and human-modifiable, giving an intuitive method for applying changes to the resulting SDF.

2. Related Work

Unsupervised Surface Reconstruction. Representing shapes with INRs was popularized by DeepSDF [30], where a ReLU network directly models an SDF using GT SDF supervision. SAL [1] instead uses unsigned distance supervision, approximated from the input point cloud, as an unsupervised and unoriented approach. SALD [2] showed that adding unoriented normal supervision improved upon this, and IGR [14] further added the eikonal constraint loss (see Equation 5) that has been widely adopted [5, 22, 34]. NSP [45] addresses the task by solving a kernel regression problem but needs input normals. PHASE [22] introduces a novel loss inspired by the phase transition problem that encourages boundary crossings at the surface points.

However, as the inputs to INRs are low dimensional, they are biased to low frequency solutions that do not recover high fidelity detail [36]. To introduce high frequencies, SIREN [34] uses sine activations and Fourier Feature Networks [36] use a fixed Fourier encoding. While being able to reconstruct high fidelity shapes with a modest number of parameters, these methods are hard to train as the high frequencies overfit quickly. A different approach is parametric encoding [29, 35], where spatial regions are represented by trainable parameters (often using an octree structure). We cover such methods in the next subsection.

Note that IGR and SIREN can operate without normals, which we refer to by "IGR wo n" and "SIREN wo n".

Octrees for Surface Reconstruction. Octrees were first introduced by Meagher [27] as an efficient data structure for representing and manipulating 3D data, and are now used extensively. The classic method Poisson Surface Reconstruction (PSR) [18] and its followup SPSR [19] use octrees to structure their implicit functions.

Early works on using octrees with neural networks include HSP [16] and OGN [38], who train encoder-decoder networks to learn octrees with occupancy values, and Oct-Net [32] and O-CNN [40], who implement 3D CNNs that operate over octrees rather than voxel grids for shape analysis tasks. O-CNN's descendants [41, 43] build on this and adapt it to shape generation, completion and reconstruction.

More recently octrees have been used in INRs for representing fine details efficiently. OctField [37] and NSVF [23] both learn local implicit functions for individual octree nodes. NGLOD [35] and ACORN [26] instead learn a single INR, but store trainable parameters at octree node vertices to encode spatial regions. IMLSNet [24] uses O-CNN to predict both an octree and oriented points per octree node to represent the surface using implicit moving least squares.

However, previous octree reconstruction methods were supervised, trained on GT SDF/occupancy of either the current shape or a separate training set. Recently Dual Octree Graph CNNs [42] addressed the unsupervised setting, where they learn local features on the dual graph of an octree. However their method has parameters that need unsupervised training, and they require input normals.

SAP [31] truly addresses the unsupervised and unoriented task by making the PSR pipeline differentiable. Starting from an oriented point cloud of a sphere, they use Chamfer loss to guide that point cloud to a dense, oriented representation. Their method is fast and among state-of-the-art, however it can fail drastically and often does not produce a watertight mesh (see Section 4). It is also hard to adapt to different applications and losses, unlike INRs.

Initialization for INRs. Due to the difficulty of training INRs, initialization is very important [1, 5]. However regardless of input, most methods initialize to a sphere. These include SAL [1] and its descendants [2, 14, 22], DiGS [5] and IDF [47]. This gives many important properties, such as biasing shape interior to the domain's center and satisfying the eikonal condition at the start of training, but it is problematic for shapes with sharp concavities. Our method, on the other hand, provides firm guidance for both the eikonal constraint and the shape based on the input point cloud, and thus does not bias the training toward convex objects.

Energy Minimization. Discrete energy minimization has been used extensively in both 2D and 3D vision for segmentation and object labelling [9, 10]. An energy function encodes our preference over the joint assignment to a set of variables, e.g., binary variables representing occupancy of voxels in 3D space. The goal of energy minimization is to find an assignment to the variables with minimal cost. This is NP-hard in general and a number of approximation algorithms have been proposed [9]. One class of algorithms, known as move-making, start with an initial assignment and iteratively evaluates changes to a subset of the variables (called a move). If the change results in lower energy then the move is accepted, otherwise the changed values are reverted. Moves are repeatedly tried until no further reduction in energy is found. Iterated Conditional Modes (ICM) [8] is the simplest example, where each move involves changing a single variable and so is very efficient, but it is prone to getting stuck in local minima. Generalized ICM improves on this by allowing multiple variables to change in a move, but it is very costly. Our method is a variant of generalized ICM specialized to our problem, where we introduce restrictions on the move space improve efficiency.

Algorithm 1 Overall octree labeling algorithm

```
1:  procedure LabelOctree($\chi$, $d_i$, $k$, $C$)
2:      $\mathcal{O}$ = CreateOctree($\chi$,$d_i$)
3:      Initialization($\mathcal{O}$)
4:      for $j$ in $0, \ldots, k$ do
5:          if $j \neq 0$ then
6:              ExpandTree($\mathcal{O}$)
7:          end if                    ▷ Now at depth $d = d_i + j$
8:          Grow($\mathcal{O}$)
9:          for $s$ in $C$ do
10:             makeMove($\mathcal{O}$, $s$)
11:         end for
12:     end for
13:     return $\mathcal{O}$
14: end procedure
```

3. OG-INR: Octree Guided Shape INRs

3.1. Problem Formulation

Let $\chi = \{x_i\}_{i \in I}$ be a point cloud sampled from a surface $\mathcal{S} = \partial\mathcal{V}$ of some watertight volume $\mathcal{V} \subset \mathcal{D}$ where $\mathcal{D} \subset \mathbb{R}^3$ is compact. We wish to reconstruct the surface $\mathcal{S}$. We choose to represent $\mathcal{S}$ as the zero-level set of a signed distance function (SDF), and thus wish to train a neural network Φ to output signed distances for all points $x \in \mathcal{D}$,

$$\Phi(x, \theta) = (-1)^{o(x)} d(x, \mathcal{S}) \tag{1}$$

where θ are the model parameters, $o(x)$ is the occupancy of point x (1 if inside and 0 if outside) and $d(x, \mathcal{S})$ is the distance from x to its closest point on $\mathcal{S}$.

3.2. Overview

We first aim to produce a rough and discrete, but globally consistent, occupancy determination for our domain by solving for an inside-outside labelling on nodes of an octree. To do this we propose an energy function that combines both the watertight surface property and the minimal surface objective. Despite this energy being intractable to solve exactly, we provide a domain-guided move-making algorithm to get low energy solutions (see Section 3.3).

We then design additional INR loss terms to guide INRs using our octree labelling (see Section 3.4). These losses quickly guide the INR towards a rough but globally consistent representation that allows it to avoid many bad local minima. Since these loss terms are agnostic to INR architecture, our method can work with any INR model.

As we assume our domain is compact, we first normalize the point cloud to fit within the unit sphere in 3D, and then set our domain as $\mathcal{D} = [-1.1, 1.1]^3$.

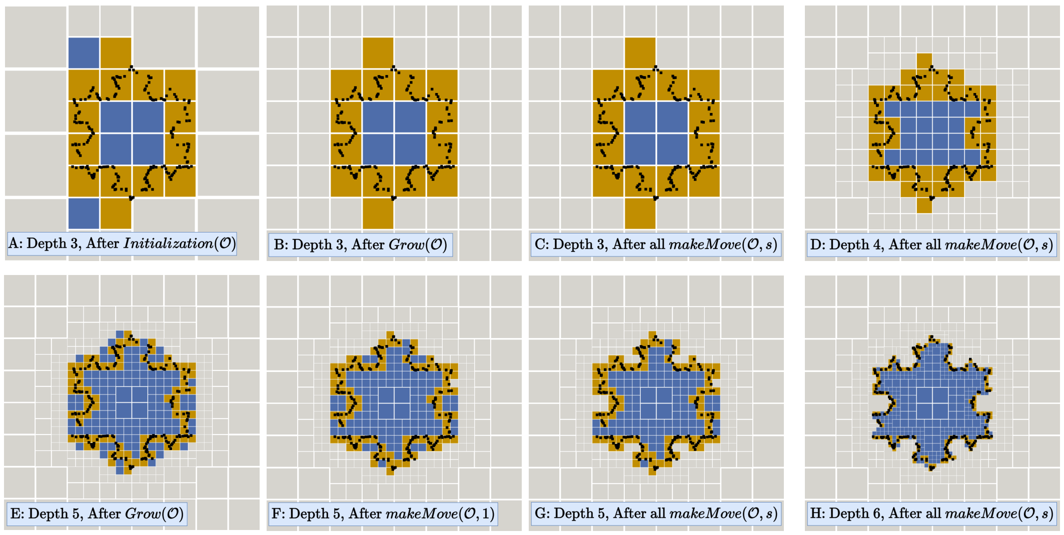
A: Depth 3, After $Initialization(\mathcal{O})$
B: Depth 3, After $Grow(\mathcal{O})$
C: Depth 3, After all $makeMove(\mathcal{O}, s)$
D: Depth 4, After all $makeMove(\mathcal{O}, s)$
E: Depth 5, After $Grow(\mathcal{O})$
F: Depth 5, After $makeMove(\mathcal{O}, 1)$
G: Depth 5, After all $makeMove(\mathcal{O}, s)$
H: Depth 6, After all $makeMove(\mathcal{O}, s)$

Figure 3. Our algorithm demonstrated on a 2D quadtree example. The black dots are our input point cloud, the yellow squares are surface leaves, and the blue and gray squares are non-surface leaves labelled as inside and outside respectively.

3.3. Octree Construction and Labelling

Setup. We present our octree labelling algorithm in Algorithm 1, which takes in point cloud χ, an initial depth to build the octree to d_i, a number of further depth expansions k (the final octree has depth $d_f = d_i + k$) and a set of move set sizes C. We set $d = 3$, $k = 4$ and $C = [1, 2, 10, 10000]$ in our experiments. The algorithm returns a labelled octree $\mathcal{O} = (\mathcal{N}, \mathcal{L}_{ns}, \mathcal{L}_s, y)$ where

- $\mathcal{N}$ are the nodes of the octree. The octree is built by repeatedly expanding nodes containing or near points from χ, so it is highly unbalanced.
- $\mathcal{L} \subset \mathcal{N}$ are the leaf nodes, and thus $\mathcal{L}$ partitions $\mathcal{D}$. Furthermore, we define surface leaves $\mathcal{L}_s \subset \mathcal{L}$ to be leaves that contain points from the point cloud (colored yellow in Figure 2 and Figure 3) and all other leaves to be non-surface leaves $\mathcal{L}_{ns} = \mathcal{L} \setminus \mathcal{L}_s$.
- y is the binary occupancy label vector indexed by non-surface leaves $\ell \in \mathcal{L}_{ns}$ denoting whether the space defined by each ℓ is inside, $y_\ell = 1$, or outside, $y_\ell = 0$.

We will construct our algorithm so that the neighbours of every surface leaf ℓ is at the same depth. Thus let n_ℓ^s and n_ℓ^{ns} be the number of its surface and non-surface neighbours and $n_\ell^{(1)}(\boldsymbol{y})$ and $n_\ell^{(0)}(\boldsymbol{y})$ be the number of inside and outside neighbors w.r.t. labelling $\boldsymbol{y}$. Then $n_\ell^{ns} = n_\ell^{(1)}(\boldsymbol{y}) + n_\ell^{(0)}(\boldsymbol{y})$ and $n_\ell^s + n_\ell^s = 26$ (or 8 for our 2D example in Figure 3).

We will constantly refer to the *boundary* as the leaves touching the boundary of $\mathcal{D}$ and the inside-outside *border* to be the non-surface leaves who have a non-surface neighbor with a different label (different occupancy value).

We determine inside-outside occupancy based on three properties: (1) *surface property:* surfaces should separate inside and outside regions, (2) *minimal surface constraint:* the total surface area of the shape's surface $\mathcal{S}$ should be as small as possible, (3) *boundary constraint:* the boundary of the domain $\mathcal{D}$ should be outside.

Energy function. We propose an energy function over labellings $\boldsymbol{y} = \{y \mid y_i \in \{0, 1\}, i \in \mathcal{L}_{ns}\}$ based on the first two criteria, the third is hard-coded at the start of the algorithm. For the surface property, ideally every surface leaf would have both inside and outside neighbours, however it may only have surface leaf neighbours which contain both inside and outside regions by definition (see Figure 3 C,D,G,H, some surface leafs have no inside neighbours). For the minimal surface constraint, we only create surfaces when adjacent non-surface leaves have different labellings (note surface leaves are fixed), so we want to minimise this.

For the surface property, we want every surface leaf ℓ to have some amount of both inside and outside neighbours, where surface neighbours partially count towards both. Thus, we aim for $\gamma_\ell^{(1)} = \gamma^{(1)} - \eta^{(1)} n_\ell^s$ non-surface inside neighbors and $\gamma_\ell^{(0)} = \gamma^{(0)} - \eta^{(0)} n_\ell^s$ non-surface outside neighbors where $\gamma^{(\cdot)}, \eta^{(\cdot)}$ are hyperparameters. Then the energy cost for a surface leaf ℓ is

$$E_\ell^{SP}(\boldsymbol{y}) = \max\left\{\gamma_\ell^{(1)} - n_\ell^{(1)}(\boldsymbol{y}), \gamma_\ell^{(0)} - n_\ell^{(0)}(\boldsymbol{y}), 0\right\}. \quad (2)$$

For the minimal surface constraint, we want to minimise the total surface area between every pair of non-surface neighbors with different labellings. Then for two non-surface leaves ℓ, ℓ', our energy cost is

$$E_{\ell,\ell'}^{MSC}(\boldsymbol{y}) = a_{\ell\ell'} |y_\ell - y_{\ell'}| \quad (3)$$

where $a_{\ell\ell'}$ is the (surface) area in common between them (note $a_{ij} = 0$ if they are not adjacent).

The two criteria can be formalized as an energy function over labellings $\boldsymbol{y} = \{y_i \in \{0,1\} \mid i \in \mathcal{L}_{ns}\}$ by

$$E(\boldsymbol{y}) = \sum_{j \in \mathcal{L}_s} E_j^{SP}(\boldsymbol{y}) + \lambda \sum_{i,j \in \mathcal{L}_{ns}} E_{i,j}^{MSC}(\boldsymbol{y}) \quad (4)$$

where λ is a hyperparameter.

3.3.1 Octree Creation and Labelling Algorithm

The energy function in Equation 4 is non-trivial to minimize. It is a discrete energy minimisation problem with hard unary terms (the boundary constraint), pairwise terms (Equation 3 which is essentially a smoothness constraint) and n-ary terms (Equation 2). The n-ary term however is not concave, so we cannot use existing methods to perform exact inference using graph-cuts [13, 20].

Instead, we employ a move-making algorithm [9] with moves specialized for our task (see Figure 3). We first hard code the boundary constraint (leaves on the boundary of the domain are outside) and initialize all other leaves to inside. To be efficient, we **(1)** only consider moves that change leaves from inside to outside (never the other way around), **(2)** require such moves include an inside leaf from the inside-outside border, **(3)** iteratively grow such potential moves, and **(4)** apply moves at multiple depths. We also have a grow stage to remove extraneous inside leaves without needing guidance from the energy function. We now describe the stages of our algorithm, depicted in Algorithm 1.

Octree Creation. We start with a single node n_0 covering the whole space $\mathcal{D}$. We then recursively expand any node (into 8 equally sized child nodes) that contains points from χ in it until depth d_i is reached. Note that every surface leaf is at depth d.

Initialisation. We hard code our boundary condition and set every other leaf node to inside (see Figure 3 A).

Grow Stage. We first expand any neighbor of a surface leaf that is not also at the current depth d (new leaves keep the label of its parent). We repeatedly expand the outside region changing inside leaves to outside if they stick out from a plane, *i.e.*, have at least $26/2 + 1 = 14$ outside neighbors (or $8/2 + 1 = 5$ for our 2D example in Figure 3). Finally we expand nodes that are on the inside-outsider border and are not at the current octree depth.

Move-Making Stage. To overcome local minima we propose sets of leaves to change from inside to outside in a single move. Sets are constructed as follows. Starting with an inside leaf on the border we greedily add neighboring inside leaves to the set, selecting the neighbor which results in lowest energy, and calculate the cost of the current move set. Note that the energy can temporarily increase. We do this until there are no inside neighbors remaining or a maximum set size s is reached.[1] If any of these moves result in a decrease in energy, we apply the move with the largest decrease, We repeat the above for different starting leaves and for multiple increasing set sizes $s \in C$.

Figure 3 illustrates this process in 2D. After reaching labelling (E), all single variable changes ($s = 1$) are considered, resulting in (F). Observe that the inside-outside border has been refined but larger regions are still incorrectly labelled. Considering larger moves as shown in (G) fixes many of these regions, with further refinements being made at deeper levels of the octree as shown in (H). We explain this 2D example in more detail in the supplemental material, including why we cannot reach a lower energy from fixing certain regions until a deeper level is reached.

Expanding. We find that labelling at multiple depths is faster and more stable than directly labelling the largest depth. Note that expanding completely changes $\mathcal{L}_s$ and adds more nodes to $\mathcal{L}_{ns}$, and thus expands y. We keep the labels for previous non-surface leaves, and assign all the new non-surface leaves to inside.

3.4. SDF learning

We use the octree to guide the SDF learning. Given an INR, we train it with the combination of four losses. The first two are the standard losses in the literature [1, 2, 5, 14, 34], the surface loss to enforce that the input point cloud should lie on the zero level set and the eikonal loss [14] to enforce that the gradient has magnitude one everywhere

$$\mathcal{L}_1 = \sum_{x \in \chi} |\Phi(x,\theta)|, \; \mathcal{L}_2 = \int_{\mathcal{D}} |\|\nabla_x \Phi(x,\theta)\|_2 - 1| \, \mathrm{d}x. \quad (5)$$

We now introduce two loss functions that take guidance from the labelled octree. We first produce an approximate signed distance function $\tilde{d}_s(x)$ for finely sampled points $x \in \mathcal{D}$ by computing the distance to the closest input point cloud point and choose its sign based on the label of the octree leaf that contains that point (for surface leaves we choose inside). We use this as direct supervision

$$\mathcal{L}_3 = \int_{\mathcal{D}} |\Phi(x,\theta) - \tilde{d}_s(x)| \, \mathrm{d}x. \quad (6)$$

We also enforce the sign of each octree leaf explicitly (again taking surface leaves to be considered as inside) at a fixed number of randomly sampled points in that leaf S_ℓ

$$\mathcal{L}_4 = \sum_{\ell \in \mathcal{L}} \left(\sum_{x \in S_\ell} \left[(-1)^{(1-y'_\ell)} \Phi(x,\theta) \right]_{>0} \right) \quad (7)$$

where $y'_\ell = y_\ell$ for $\ell \in \mathcal{L}_{ns}$ and $y'_\ell = 1$ for $\ell \in \mathcal{L}_s$. Note that sampling a fixed number per leaf automatically adjusts our sampling to be denser in more important regions (*i.e.*, closer to the surface) due to the octree structure.

[1] In practice we include additional early stopping criteria to avoid moves that are unlikely to eventually be accepted (see supplemental for details).

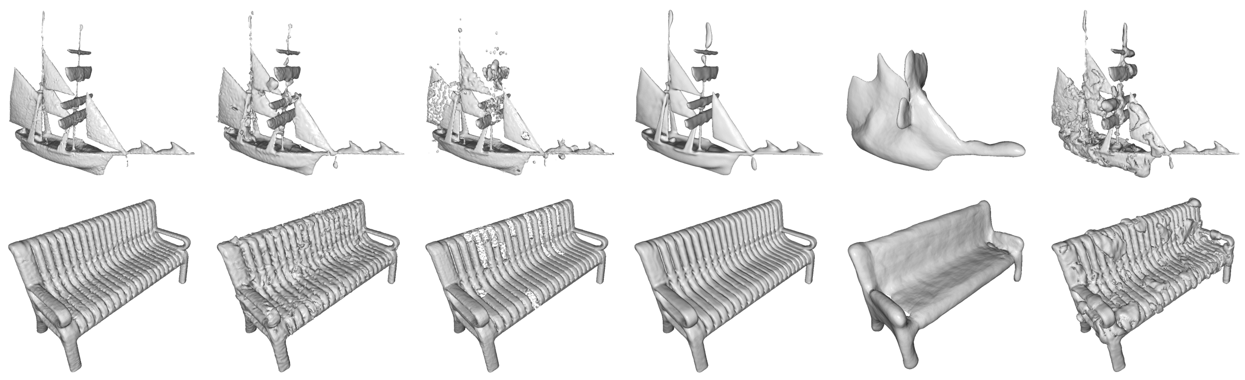

Figure 4. A comparison of surface reconstructions of shapes from the `watercraft` and `bench` classes of ShapeNet. The left two reconstructions are using OG-SIREN with the octree built to different depths.

Our overall loss is then the weighted combination

$$\mathcal{L}(\Phi) = \lambda_1\mathcal{L}_1 + \lambda_2\mathcal{L}_2 + \lambda_3\mathcal{L}_3 + \lambda_4\mathcal{L}_4. \tag{8}$$

We keep λ_1 and λ_2 fixed throughout training as their loss terms have no bias. As the guidance in $\mathcal{L}_3$ and $\mathcal{L}_4$ is approximate but highly informative, we initially use large values for λ_3, λ_4, allowing the INR to quickly fit to the rough shape, and then reduce them during training. More training details are in the supplemental material.

4. Results

Datasets. We evaluate the performance of our method on the ShapeNet [12] and Surface Reconstruction Benchmark (SRB) [7] datasets. ShapeNet has a wide variety of shapes, with thin surfaces, hollow regions and disjoint objects. However the shapes are rather uniformly sampled. We use the preprocessing and split of Neural Splines [45], which consists of 13 shape classes with 20 shapes in each shape class. For each shape it provides the ground truth mesh (preprocessed to be watertight), an input point cloud and occupancy annotation at dense points. SRB contains five shapes from simulated noisy range scans, but it is a popularly used dataset due to the shapes exhibiting many challenging features. These include large missing regions, thin and sharp regions, and small surface details. The former means that good priors in the reconstruction methods, such as smoothness priors, are very important.

We follow the metrics used in the literature for each dataset: the squared Chamfer distance and IoU for ShapeNet and Chamfer and Hausdorff distance for SRB.

INRs. We demonstrate our method on two INRs, SIREN [34], and NGLOD [35] (note that NGLOD only works in a supervised setting as it has no global consistency, but our octree guidance is sufficient). Thus we name our two methods OG-SIREN and OG-NGLOD where OG stands for "Octree Guided". To extract a mesh we first evaluate our INRs on a 256^3 grid and then use marching cubes [25]. We use the same settings for the octree and the INRs on both datasets, and give more details in our supplemental.

4.1. Reconstruction Performance

ShapeNet dataset. The results are shown in Table 1 (left) and visualizations on two shapes are shown in Figure 4. Our methods get the smallest mean squared Chamfer distance out of the unoriented methods by almost an order of magnitude, and are even better than the best performing oriented method. Likewise for mean IoU our methods are better than both unoriented and oriented methods. For median performance, our methods are comparable to the best oriented and unoriented methods. SAP [31] gets the best median IoU, but their mean is much lower than ours, indicating that in a few instances they misrepresent large regions. On the other hand, due to our move-making algorithm we reduce the likelihood of vastly incorrect labellings, and as a result achieve low standard deviations on both metrics despite the high performance.

SRB dataset. Table 1 (right) shows that DiGS, SAP, PHASE+FF and our two methods do comparably, with DiGS performing slightly better. Note the performance of these methods are comparable to most oriented methods. Also note that the difference between our two methods is more prominent on SRB as the large missing regions means that smoothness priors are more important. Octree guidance is NGLOD's only global supervision, as it fits locally, while SIREN is biased towards smooth shapes.

4.2. Octree Depth Ablations

Table 4 shows the results of OG-SIREN on ShapeNet when evaluating at different depths of the octree. We include an octree IoU metric, where we compute the IoU of the labelling of the octree directly (taking surface leaves to

	Squared Chamfer ↓			IoU ↑		
Method	mean	median	std	mean	median	std
SPSR [19]	5.44e-5	1.97e-5	5.06e-4	0.9926	0.9956	0.0105
IGR [14]	5.12e-4	1.13e-4	2.15e-3	0.8102	0.8480	0.1519
SIREN [34]	1.03e-4	5.28e-5	1.93e-4	0.8268	0.9097	0.2329
FFN [36]	9.12e-5	8.65e-5	3.36e-5	0.8218	0.8396	0.0989
Biharmonic RBF [11]	1.11e-4	8.97e-5	7.06e-5	0.8247	0.8642	0.1350
SVR [33]	1.14e-4	1.04e-4	5.99e-5	0.7625	0.7819	0.1300
NSP [45]	**5.36e-5**	4.06e-5	3.64e-5	0.8973	0.9230	0.0871
DiGS + n [5]	2.74e-4	**2.32e-5**	9.90e-4	**0.9200**	**0.9774**	0.1992
N Est+SPSR [19]	3.76e-3	4.37e-5	1.14e-2	0.7187	0.9761	0.3767
SIREN wo n [34]	3.08e-4	2.58e-4	3.26e-4	0.3085	0.2952	0.2014
SAL [1]	1.14e-3	2.11e-4	3.63e-3	0.4030	0.3944	0.2722
DiGS [5]	1.32e-4	2.55e-5	4.73e-4	0.9390	0.9764	0.1262
SAP [31]	4.09e-4	2.46e-5	2.60e-3	0.9118	**0.9923**	0.2002
Our OG-SIREN	**3.75e-5**	**2.17e-5**	8.24e-5	**0.9615**	0.9871	0.1048
Our OG-NGLOD	5.07e-5	2.57e-5	9.39e-5	0.9593	0.9870	0.1057

Method	$d_C \downarrow$	$d_H \downarrow$
SPSR [19]	0.21	4.69
DGP [44]	0.21	5.18
IGR [14]	0.19	2.99
SIREN [34]	0.19	3.86
NSP [45]	0.17	2.85
PHASE [22]	**0.16**	**2.77**
DiGS + n [5]	0.18	3.55
N Est.+SPSR [19]	1.25	22.59
IGR wo n [14]	1.38	16.33
SIREN wo n [34]	0.42	7.67
SAL [14]	0.36	7.47
IGR+FF [14]	0.96	11.06
PHASE+FF [22]	0.22	4.96
DiGS [5]	**0.19**	**3.52**
SAP [31]	0.21	4.51
Our OG-SIREN	0.20	4.06
Our OG-NGLOD	0.22	6.03

Table 1. Surface Reconstruction results on ShapeNet [12] (left) and SRB [7] (right). Methods above the line use ground truth normal information, and methods below do not. **Left:** The mean, median and standard deviation of the squared Chamfer distance and IoU of all 260 shapes are reported. **Right:** The mean Chamfer d_C and Hausdorff distance d_H for all 5 shapes are reported.

Method	$d_C \downarrow$	$d_H \downarrow$
SIREN wo n [34]	0.42	7.67
OG-SIREN	0.20	4.06
SIREN (+n) [34]	0.19	3.86
OG-SIREN + H^o	0.19	2.96
OG-SIREN + GT^o	0.17	2.44
OG-NGLOD	0.22	6.03
OG-NGLOD + H^o	0.19	5.50
OG-NGLOD + GT^o	0.17	5.12
NGLOD+GT^d [35]	0.13	1.97

Table 2. Results on SRB [7] when supervising INRs with varying levels of information (see Section 4.3). GT^d means dense GT SDF supervision, while GT^o and H^o mean GT and human estimated occupancy per octree node.

be inside). The IoU of the octree is much lower than the IoU of the final mesh, indicating that the INR training is learning beyond the octree supervision. Furthermore as the depth increases, the octree IoU improves and so do the metrics on the final mesh, indicating that the INR training is heavily affected by the initial training guidance from the octree. See the supplemental material for a depth ablation on SRB.

4.3. Modifying the octree

To show the versatility of our method, in Table 2 we demonstrate how modifying the octree labelling can affect the performance of the resulting shape. For OG-INR+GT^o we set octree labels using GT occupancy. For OG-INR+H^o we set octree labels using sparse human supervision: we change up to 30 large octree leaves at the lower depths (up to depth 5) where the changes are obvious from a human perspective. With both octree supervision signals we get increases in performance, showing that the discrete nature of octree guidance does not introduce much bias.

We also compare to the INRs with other forms of supervision. As NGLOD cannot be used in an unsupervised setting, we provide result when trained on dense GT.

4.4. Further Comparisons

The need for unoriented methods. We compare to a recent industry-standard implementation [48] of normal estimation + SPSR [19] (SPSR+N Est.). Table 1 (left) shows that for the well sampled surfaces in ShapeNet, when ground truth normals are available SPSR achieves close to perfect IoUs. However when normals need to be estimated, it has good median performance but very poor average performance, indicating the normal estimation procedure will often be quite inaccurate. Table 1 (right) shows that for point clouds with large missing regions, SPSR with ground truth normals has room for improvement, while SPSR with estimated normals performs extremely poorly.

While neither dataset has provided line of sight information (as would be given by sensors), a more informative use of such information than estimating rough normals would be to use it to label parts of our octree. See Section 4.3 for how additional octree supervision improves performance.

Overall Timing Comparison. Table 3 shows the time taken for each method. Our method's gradient descent training is the fastest to converge as it needs much fewer iterations. Even when adding on the average octree building time from ShapeNet, we can see that the time taken for both

Method	Parameters	Time per iter. (s)	Num iters	Time (s)	Speed Up
N Est. +SPSR	-	-	-	42 (13 + 29)	12× (· + 18×)
SIREN (wo n)	264K	0.052	10000	520	1×
SAL	2.1M	0.175	10000	1750	0.3×
DiGS	264K	0.120	10000	1200	0.4×
SAP	120K	-	3200	330	1.6×
Our OG-SIREN	264K	0.158	600	135 (40 + 95)	3.9× (· + 5.5×)
Our OG-NGLOD	68.7M	0.173	300	**92 (40 + 52)**	**5.7**× (· + **10**×)

Table 3. Time and number of parameters for each model on ShapeNet. For methods with a precomputation stage (normal estimation or octree building), the time for both is included with the precomputation and training times in parenthesis. Speed up is measured with respect to SIREN/SIREN wo n. We ran our experiments on a single Nvidia RTX 2080 GPU and a i7-8700K CPU.

	Octree			Final Mesh (OG-SIREN)						Octree		
	IoU ↑			IoU ↑			Squared Chamfer ↓			Build + Label Time (s)		
Depth	Mean	Median	Std	Mean	Median	Std	Mean	Median	Std	Mean	Median	Std
5	0.4748	0.4702	0.1854	0.7101	0.7712	0.2594	1.84e-4	1.35e-4	2.23e-4	2.2	1.6	2.2
6	0.6401	0.6410	0.1604	0.9503	0.9835	0.1172	3.93e-5	2.34e-5	8.20e-5	7.6	5.3	8.6
7	0.7757	0.8006	0.1305	0.9615	0.9871	0.1048	3.75e-5	2.17e-5	8.24e-5	38.1	19.3	67.7

Table 4. Octree depth ablation on ShapeNet. Octree IoU is computed using the labelling of the leaves (surface leaves chosen as inside).

stages of our method is still faster than other methods, often by an order of magnitude. We implemented the octree in Cython [3] but further optimisations are possible.

Comparison Against No Octree Guidance. Comparing SIREN wo n with and without the octree guidance we can see that the guidance drastically improves both time performance and reconstruction metrics (see Table 1,Table 3). When initially guided by our octree labelling, SIREN (wo n) converges significantly faster (92s vs 520s) and gets a 3x improvement in ShapeNet IoU. Without guidance from normals or our octree, SIREN wo n gets stuck in bad local minima where it needs to flip its inside-outsideness prediction, causing ghost geometry (see Figure 1). When comparing the two types of INR training guidance applied to SIREN wo n, ground truth normals vs our octree, we can see that the octree guidance still performs much better than SIREN on ShapeNet, while performing comparably on SRB.

4.5. Discussion

Limitations. Like all unoriented methods, the octree guidance has difficulty with large missing regions, which is demonstrated heavily in SRB, and thin surfaces, which appear in SRB (Daratech) and ShapeNet (airplane and watercraft). Despite this, our methods perform competitively on shapes with these difficulties on both datasets. See the supplemental for per class breakdown and visualizations.

Other possible limitations would be the ability to deal with noise and outliers. Note that all methods do poorly with large amounts of either corruption. For small noise the method's performance does not degrade considerably as demonstrated by the results on SRB. For larger noise, the volume occupied by surface leaves would be much larger. While this would not introduce error, the octree's benefits would be reduced. For outliers there would be significant bias introduced by the octree method, which if sparse could be preprocessed out beforehand.

Broader Impact. Faster and more accurate unoriented surface reconstruction methods will allow for accurate meshes to be constructed easily from regular scanners. Potential misuses of this technology include unauthorized replication and harmful content creation.

5. Conclusion

We have introduced an octree guided method for training implicit neural representations (INRs) for the task of unoriented surface reconstruction. We formulate the main difficulty of the task, inside-outside determinism, as an octree labelling problem with respect to an energy function that captures watertight surface properties and the minimal surface constraint. We also detail an efficient method to minimize the energy. We then use this supervision to guide the initial stage of INR training, using loss functions that can be used by any INR architecture. Our results show that applying our octree guided supervision provides significant time and reconstruction performance improvements, with competitive or better results than the current state-of-the-art.

Acknowledgements. This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 893465. S. Gould is a recipient of an ARC Future Fellowship (proj. no. LP200100421) funded by the Australian Government.

References

[1] Matan Atzmon and Yaron Lipman. SAL: Sign Agnostic Learning of shapes from raw data. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2565–2574, 2020. 2, 3, 5, 6, 7

[2] Matan Atzmon and Yaron Lipman. SALD: Sign Agnostic Learning with Derivatives. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. 2, 3, 5

[3] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011. 8

[4] Yizhak Ben-Shabat and Stephen Gould. DeepFit: 3D surface fitting via neural network weighted least squares. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 20–34. Springer, 2020. 1

[5] Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. Digs: Divergence guided shape implicit neural representation for unoriented point clouds. *arXiv preprint arXiv:2106.10811*, 2021. 2, 3, 5, 6, 7

[6] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. Nesti-net: Normal estimation for unstructured 3D point clouds using convolutional neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10112–10120, 2019. 1

[7] Matthew Berger, Joshua A Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T Silva. A benchmark for surface reconstruction. *ACM Trans. on Graphics (ToG)*, 32:1–17, 2013. 6, 7

[8] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society: Series B (Methodological)*, 48(3):259–279, 1986. 3

[9] Andrew Blake, Pushmeet Kohli, and Carsten Rother. *Markov Random Fields for Vision and Image Processing*. The MIT Press, 07 2011. 3, 5

[10] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001. 3

[11] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. of the Conference on Computer Graphics and Interactive Techniques*, pages 67–76, 2001. 7

[12] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 6, 7

[13] Stephen Gould. Max-margin learning for lower linear envelope potentials in binary markov random fields. In *Proc. of the International Conference on Machine Learning (ICML)*. Omnipress, 2011. 5

[14] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit Geometric Regularization for learning shapes. In *Proc. of the International Conference on Machine Learning (ICML)*, volume 119, pages 3789–3799. PMLR, 2020. 2, 3, 5, 7

[15] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. PCPNet: Learning local shape properties from raw point clouds. In *Computer Graphics Forum*, volume 37, pages 75–85. Wiley Online Library, 2018. 1

[16] 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. 2

[17] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *UIST '11 Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, October 2011. 1

[18] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proc. of the Eurographics Symposium on Geometry Processing (SGP)*, volume 7, 2006. 2

[19] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. on Graphics (ToG)*, 32:1–13, 2013. 2, 7

[20] Pushmeet Kohli, M. Kumar, and Philip Torr. P3 & beyond: Solving energies with higher order cliques. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 5

[21] Jan Eric Lenssen, Christian Osendorfer, and Jonathan Masci. Deep iterative surface normal estimation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11247–11256, 2020. 1

[22] Yaron Lipman. Phase transitions, distance functions, and implicit neural representations. *arXiv preprint arXiv:2106.07689*, 2021. 2, 3, 7

[23] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 2

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

[25] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proc. of the Intl. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, volume 21, pages 163–169. ACM New York, NY, USA, 1987. 6

[26] Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *arXiv preprint arXiv:2105.02788*, 2021. 3

[27] Donald J. Meagher. Geometric modeling using octree encoding. *Comput. Graph. Image Process.*, 19:85, 1982. 2

[28] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks:

Learning 3D reconstruction in function space. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470, 2019. 1

[29] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 1, 2

[30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. 1, 2

[31] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape As Points: A differentiable poisson solver. *arXiv preprint arXiv:2106.03452*, 2021. 3, 6, 7

[32] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017. 2

[33] Bernhard Schölkopf, Joachim Giesen, and Simon Spalinger. Kernel methods for implicit surface modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, NIPS'04, page 1193–1200, Cambridge, MA, USA, 2004. MIT Press. 7

[34] 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, 5, 6, 7

[35] 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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 3, 6, 7

[36] 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. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2, 7

[37] Jia-Heng Tang, Weikai Chen, Jie Yang, Bo Wang, Songrun Liu, Bo Yang, and Lin Gao. Octfield: Hierarchical implicit functions for 3d modeling. In *The Thirty-Fifth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 2

[38] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE international conference on computer vision*, pages 2088–2096, 2017. 2

[39] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. *arXiv preprint arXiv:2111.05849*, 2021. 1

[40] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11, 2017. 2

[41] Peng-Shuai Wang, Yang Liu, and Xin Tong. Deep octree-based cnns with output-guided skip connections for 3d shape and scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 266–267, 2020. 2

[42] Peng-Shuai Wang, Yang Liu, and Xin Tong. Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Transactions on Graphics (TOG)*, 41(4):1–15, 2022. 3

[43] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. Adaptive o-cnn: A patch-based deep representation of 3d shapes. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018. 2

[44] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep Geometric Prior for surface reconstruction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10130–10139, 2019. 7

[45] Francis Williams, Matthew Trager, Joan Bruna, and Denis Zorin. Neural Splines: Fitting 3D surfaces with infinitely-wide neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9949–9958, 2021. 2, 6, 7

[46] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022. 1

[47] Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. Geometry-consistent neural shape representation with implicit displacement fields. *arXiv preprint arXiv:2106.05187*, 2021. 3

[48] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 7