# Shape As Points: A Differentiable Poisson Solver

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

In recent years, neural implicit representations gained popularity in 3D reconstruction due to their expressiveness and flexibility. However, the implicit nature of neural implicit representations results in slow inference time and requires careful initialization. In this paper, we revisit the classic yet ubiquitous point cloud representation and introduce a differentiable point-to-mesh layer using a differentiable formulation of Poisson Surface Reconstruction (PSR) that allows for a GPU-accelerated fast solution of the indicator function given an oriented point cloud. The differentiable PSR layer allows us to efficiently and differentiably bridge the explicit 3D point representation with the 3D mesh via the implicit indicator field, enabling end-to-end optimization of surface reconstruction metrics such as Chamfer distance. This duality between points and meshes hence allows us to represent shapes as oriented point clouds, which are explicit, lightweight and expressive. Compared to neural implicit representations, our Shape-As-Points (SAP) model is more interpretable, lightweight, and accelerates inference time by one order of magnitude. Compared to other explicit representations such as points, patches, and meshes, SAP produces topology-agnostic, watertight manifold surfaces. We demonstrate the effectiveness of SAP on the task of surface reconstruction from unoriented point clouds and learning-based reconstruction.

## 1 Introduction

Shape representations are central to many of the recent advancements in 3D computer vision and computer graphics, ranging from neural rendering [39, 41, 45, 54] to shape reconstruction [10, 26, 38, 44, 47, 49, 65]. While conventional representations such as point clouds and meshes are efficient and well-studied, they also suffer from several limitations: Point clouds are lightweight and easy to obtain, but do not directly encode surface information. Meshes, on the other hand, are usually restricted to fixed topologies. More recently, neural implicit representations [10, 38, 47] have shown promising results for representing geometry due to their flexibility in encoding varied topologies, and their easy integration with differentiable frameworks. However, as such representations implicitly encode surface information, extracting the underlying surface is typically slow as they require numerous network evaluations in 3D space for extracting complete surfaces using marching cubes [10, 38, 47], or along rays for intersection detection in the context of volumetric rendering [41, 44, 46, 65].

In this work, we introduce a novel Poisson solver which performs fast GPU-accelerated Differentiable Poisson Surface Reconstruction (DPSR) and solves for an indicator function from an oriented point cloud in a few milliseconds. Thanks to the differentiablility of our Poisson solver, gradients from a loss on the output mesh or a loss on the intermediate indicator grid can be efficiently backpropagated to update the oriented point cloud representation. This differential bridge between points, indicator functions, and meshes allows us to represent shapes as oriented point clouds. We therefore call this shape representation *Shape-As-Points* (SAP). Compared to existing shape representations, Shape-As-Points has the following advantages (see also Table 1):

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

| Representations | | Points [17] | Voxels [11] | Meshes [57] | Patches [20] | Implicits [38] | **SAP** (Ours) | GT |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Efficiency | Grid Eval Time ($128^3$) | n/a | n/a | n/a | n/a | 0.33s | 0.012s | |
| Priors | Easy Initialization | ✔ | ✔ | ✔ | ✘ | ✘ | ✔ | |
| | Watertight | ✘ | ✔ | ✔ | ✘ | ✔ | ✔ | |
| Quality | No Self-intersection | n/a | n/a | ✘ | ✘ | ✔ | ✔ | |
| | Topology-Agnostic | ✔ | ✔ | ✘ | ✔ | ✔ | ✔ | |

Table 1: **Overview of Different Shape Representations.** Shape-As-Points produces higher quality geometry compared to other explicit representations [11, 17, 20, 57] and requires significantly less inference time for extracting geometry compared to neural implicit representations [38].

**Efficiency:** SAP has a low memory footprint as it only requires storing a collection of oriented point samples at the surface, rather than volumetric quantities (voxels) or a large number of network parameters for neural implicit representations. Using spectral methods, the indicator field can be computed efficiently (12 ms at $128^3$ resolution[1]), compared to the typical rather slow query time of neural implicit networks (330 ms using [38] at the same resolution). **Accuracy:** The resulting mesh can be generated at high resolutions, is guaranteed to be watertight, free from self-intersections and also topology-agnostic. **Initialization:** It is easy to initialize SAP with a given geometry such as template shapes or noisy observations. In contrast, neural implicit representations are harder to initialize, except for few simple primitives like spheres [1].

To investigate the aforementioned properties, we perform a set of controlled experiments. Moreover, we demonstrate state-of-the-art performance in reconstructing surface geometry from unoriented point clouds in two settings: an optimization-based setting that does not require training and is applicable to a wide range of shapes, and a learning-based setting for conditional shape reconstruction that is robust to noisy point clouds and outliers. In summary, the main contributions of this work are:

- We present Shape-As-Points, a novel shape representation that is interpretable, lightweight, and yields high-quality watertight meshes at low inference times.
- The core of the Shape-As-Points representation is a versatile, differentiable and generalizable Poisson solver that can be used for a range of applications.
- We study various properties inherent to the Shape-As-Points representation, including inference time, sensitivity to initialization and topology-agnostic representation capacity.
- We demonstrate state-of-the-art reconstruction results from noisy unoriented point clouds at a significantly reduced computational budget compared to existing methods.

We will release our code upon publication.

## 2 Related Work

### 2.1 3D Shape Representations

3D shape representations are central to 3D computer vision and graphics. Shape representations can be generally categorized as being either *explicit* or *implicit*. *Explicit* shape representations and learning algorithms depending on such representations directly parameterize the surface of the geometry, either as a point cloud [16, 50, 51, 59, 62], parameterized mesh [22, 24, 27, 57] or surface patches [2, 20, 36, 60, 63, 64]. Explicit representations are usually lightweight and require few parameters to represent the geometry, but suffer from discretization, the difficulty to represent watertight surfaces (point clouds, surface patches), or are restricted to a pre-defined topology (mesh). *Implicit* representations, in contrast, represent the shape as a level set of a continuous function over a discretized voxel grid [14, 25, 33, 61] or more recently parameterized as a neural network, typically referred to as neural implicit functions [10, 38, 47]. Neural implicit representations have

[1] On average, our method requires 12 ms for computing a $128^3$ indicator grid from 15K points on a single NVIDIA GTX 1080Ti GPU. Computing a $256^3$ indicator grid requires 140 ms.

been successfully used to represent geometries of objects [10, 18, 38, 44, 47, 53, 55, 58] and scenes [8, 26, 34, 43, 49, 53]. Additionally, neural implicit functions are able to represent radiance fields which allow for high-fidelity appearance and novel view synthesis [37, 42]. However, extracting surface geometry from implicit representations typically requires dense evaluation of multi-layer perceptrons, either on a volumetric grid or along rays, resulting in slow inference time. In contrast, SAP efficiently solves the Poisson Equation during inference by representing the shape as an oriented point cloud.

### 2.2 Optimization-based 3D Reconstruction from Point Clouds

Several works have addressed the problem of inferring continuous surfaces from a point cloud. They tackle this task by utilizing basis functions, set properties of the points, or neural networks. Early works in shape reconstruction from point clouds utilize the convex hull or alpha shapes for reconstruction [15]. The ball pivoting algorithm [5] leverages the continuity property of spherical balls of a given radius. One of the most popular techniques, Poisson Surface Reconstruction (PSR) [28, 29], solves the Poisson Equation and inherits smoothness properties from the basis functions used in the Poisson Equation. However, PSR is sensitive to the normals of the input points which must be inferred using a separate preprocessing step. In contrast, our method does not require any normal estimation and is thus more robust to noise. More recent works take advantage of the continuous nature of neural networks as function approximators to fit surfaces to point sets [19, 23, 40, 60]. However, these methods tend to be memory and computationally intensive, while our method yields high-quality watertight meshes in a few milliseconds.

### 2.3 Learning-based 3D Reconstruction from Point Clouds

Learning-based approaches exploit a training set of 3D shapes to infer the parameters of a reconstruction model. Some approaches focus on local data priors [2, 26] which typically result in better generalization, but suffer when large surfaces must be completed. Other approaches learn object-level [33, 38, 47] or scene-level priors [12, 13, 26, 49]. Most reconstruction approaches directly reconstruct a meshed surface geometry, though some works [3, 4, 21, 31] first predict point set normals to subsequently reconstruct the geometry via PSR [28, 29]. However, such methods fail to handle large levels of noise, since they are unable to move points or selectively ignore outliers. In contrast, our end-to-end approach is able to address this issue by either moving outlier points to the actual surface or by selectively muting outliers either by forming paired point clusters that self-cancel or reducing the magnitude of the predicted normals which controls their influence on the reconstruction.

## 3 Method

At the core of the Shape-As-Points representation is a differentiable Poisson solver, which can be used for both optimization-based and learning-based surface estimation. We first introduce the Poisson solver in Section 3.1. Next, we investigate two applications using our solver: optimization-based 3D reconstruction (Section 3.2) and learning-based 3D reconstruction (Section 3.3).

### 3.1 Differentiable Poisson Solver

The key step in Poisson Surface Reconstruction [28, 29] involves solving the Poisson Equation. Let $\mathbf{x} \in \mathbb{R}^3$ denote a spatial coordinate and $\mathbf{n} \in \mathbb{R}^3$ denote its corresponding normal. The Poisson Equation arises from the insight that a set consisting of point coordinates and normals $\{\mathbf{p} = (\mathbf{c}, \mathbf{n})\}$ can be viewed as samples of the gradient of the underlying implicit indicator function $\chi(\mathbf{x})$ that describes the solid geometry. We define the normal vector field as a superposition of pulse functions $\mathbf{v}(\mathbf{x}) = \sum_{(\mathbf{c}_i, \mathbf{n}_i) \in \{\mathbf{p}\}} \delta(\mathbf{x} - \mathbf{c}_i, \mathbf{n}_i)$, where $\delta(\mathbf{x}, \mathbf{n}) = \{\mathbf{n} \text{ if } \mathbf{x} = 0 \text{ and } 0 \text{ otherwise}\}$. By applying the divergence operator, the variational problem transforms into the standard Poisson equation:

$$\nabla^2 \chi := \nabla \cdot \nabla \chi = \nabla \cdot \mathbf{v} \tag{1}$$

In order to solve this set of linear Partial Differential Equations (PDEs), we discretize the function values and differential operators. Without loss of generality, we assume that the normal vector field $\mathbf{v}$ and the indicator function $\chi$ are sampled at $r$ uniformly spaced locations along each dimension. Denote the spatial dimensionality of the problem to be $d$. Without loss of generality, we consider the three dimensional case where $n := r \times r \times r$ for $d = 3$. We have the indicator function $\chi \in \mathbb{R}^n$, the point normal field $\mathbf{v} \in \mathbb{R}^{n \times d}$, the gradient operator $\nabla : \mathbb{R}^n \mapsto \mathbb{R}^{n \times d}$, the divergence operator $(\nabla \cdot) : \mathbb{R}^{n \times d} \mapsto \mathbb{R}^n$, and the derived laplacian operator $\nabla^2 := \nabla \cdot \nabla : \mathbb{R}^n \mapsto \mathbb{R}^n$. Under such a discretization scheme, solving for the indicator function amounts to solving the linear system by inverting the divergence operator subject to boundary conditions of surface points having zero level

set. Following [28], we fix the overall scale to $m = 0.5$ at $\mathbf{x} = 0$:

$$\chi = (\nabla^2)^{-1} \nabla \cdot \mathbf{v} \quad \text{s.t.} \quad \chi|_{\mathbf{x} \in \{\mathbf{c}\}} = 0 \quad \text{and} \quad \text{abs}(\chi|_{\mathbf{x}=0}) = m \tag{2}$$

**Point Rasterization:** We obtain the uniformly discretized point normal field $\mathbf{v}$ by rasterizing the point normals onto a uniformly sampled voxel grid. We can differentiably perform point rasterization via inverse trilinear interpolation, similar to the approach in [28, 29]. We scatter the point normal values to the voxel grid vertices, weighted by the trilinear interpolation weights. The point rasterization process has $\mathcal{O}(n)$ space complexity, linear with respect to the number of grid cells, and $\mathcal{O}(N)$ time complexity, linear with respect to the number of points. See supplementary for details.

**Spectral Methods for Solving PSR:** In contrast to the finite-element approach taken in [28, 29], we solve the PDEs using spectral methods [7]. While spectral methods are commonly used in scientific computing for solving PDEs and in some cases applied to computer vision problems [32], we are the first to apply them in the context of Poisson Surface Reconstruction. Unlike finite-element approaches that depend on irregular data structures such as octrees or tetrahedral meshes for discritizing space, spectral methods can be efficently solved over a uniform grid as they leverage highly optimized Fast Fourier Transform (FFT) operations that are well supported for GPUs, TPUs, and mainstream deep learning frameworks. Spectral methods decompose the original signal into a linear sum of functions represented using the sine / cosine basis functions whose derivatives can be computed analytically. This allows us to easily approximate differential operators in spectral space. We denote the spectral domain signals with a tilde symbol, i.e., $\tilde{\mathbf{v}} = \text{FFT}(\mathbf{v})$. We first solve for the unnormalized indicator function $\chi'$, not accounting for boundary conditions

$$\chi' = \text{IFFT}(\tilde{\chi}) \qquad \tilde{\chi} = \tilde{g}_{\sigma,r}(\mathbf{u}) \odot \frac{i\mathbf{u} \cdot \tilde{\mathbf{v}}}{2\pi \|\mathbf{u}\|^2} \qquad \tilde{g}_{\sigma,r}(\mathbf{u}) = \exp\Big(-2\frac{\sigma^2 \|\mathbf{u}\|^2}{r^2}\Big) \tag{3}$$

where the spectral frequencies are denoted as $\mathbf{u} := (u, v, w) \in \mathbb{R}^{n \times d}$ corresponding to the $x, y, z$ spatial dimensions, and $\text{IFFT}(\tilde{\chi})$ represents the inverse fast Fourier transform of $\tilde{\chi}$. $\tilde{g}_{\sigma,r}(\mathbf{u})$ is a Gaussian smoothing kernel of bandwidth $\sigma$ at grid resolution $r$ in the spectral domain to mitigate ringing effects as a result of the Gibbs phenomenon from rasterizing the point normals. We denote the element-wise product as $\odot : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^n$, the L2-norm as $\|\cdot\|^2 : \mathbb{R}^{n \times d} \mapsto \mathbb{R}^n$, and the dot product as $(\cdot) : \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times d} \mapsto \mathbb{R}^n$. Finally, we subtract by the mean of the indicator function at the point set and scale the indicator function to obtain the solution to the PSR problem in Eqn. 2:

$$\chi = \underbrace{\frac{m}{\text{abs}(\chi'|_{\mathbf{x}=0})}}_{\text{scale}} \underbrace{\Big(\chi' - \frac{1}{|\{\mathbf{c}\}|} \sum_{\mathbf{c} \in \{\mathbf{c}\}} \chi'|_{\mathbf{x}=\mathbf{c}}\Big)}_{\text{subtract by mean}} \tag{4}$$

A detailed derivation of our differentiable PSR solver is provided in the supplementary material.

### 3.2 SAP for Optimization-based 3D Reconstruction

We can use the proposed differentiable Poisson solver for various applications. First, we consider the classical task of surface reconstruction from unoriented point clouds. The overall pipeline for this setting is illustrated in Fig. 1 (top). We now provide details about each component.

**Forward pass:** It is natural to initialize the oriented 3D point cloud serving as 3D shape representation using the noisy 3D input points and corresponding (estimated) normals. However, to demonstrate the flexibility and robustness of our model, we purposefully initialize our model using a generic 3D sphere with radius $r$ in our experiments. Given the orientated point cloud, we apply our Poisson solver to obtain an indicator function grid, which can be converted to a mesh using Marching Cubes [35].

**Backward pass:** For every point $\mathbf{p}_{\text{mesh}}$ sampled from the mesh $\mathcal{M}$, we calculate a bi-directional L2 Chamfer Distance $\mathcal{L}_{\text{CD}}$ with respect to the input point cloud. To backpropagate the loss $\mathcal{L}_{\text{CD}}$ through $\mathbf{p}_{\text{mesh}}$ to point $\mathbf{p}$ in our source oriented point cloud, we decompose the gradient using the chain rule:

$$\frac{\partial \mathcal{L}_{\text{CD}}}{\partial \mathbf{p}} = \frac{\partial \mathcal{L}_{\text{CD}}}{\partial \mathbf{p}_{\text{mesh}}} \frac{\partial \mathbf{p}_{\text{mesh}}}{\partial \chi} \frac{\partial \chi}{\partial \mathbf{p}} \tag{5}$$

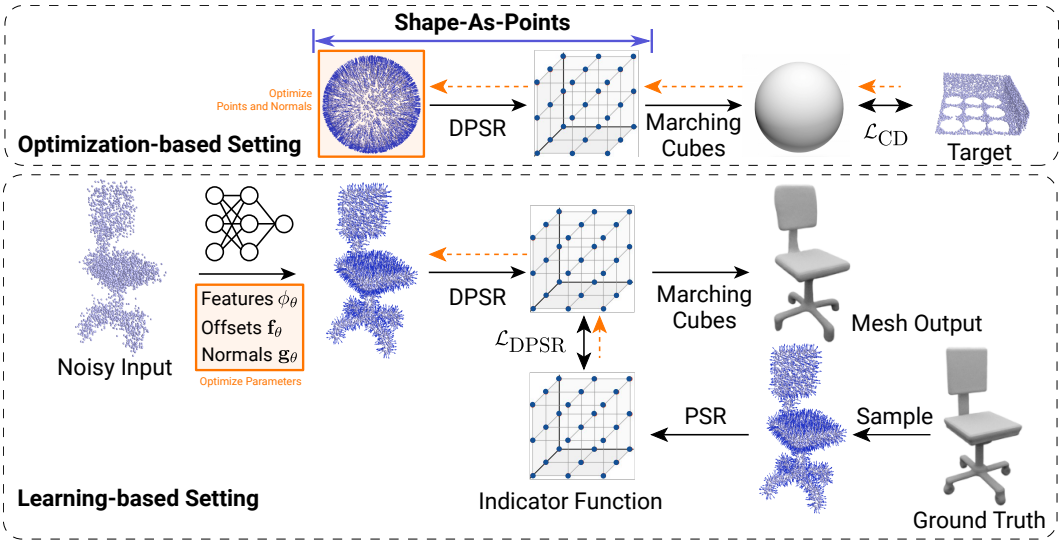


Figure 1: **Model Overview.** Top: Pipeline for optimization-based single object reconstruction. The Chamfer loss on the target point cloud is backpropagated to the source point cloud w/ normals for optimization. Bottom: Pipeline for learning-based surface reconstruction. Unlike the optimization-based setting, here we provide supervision at the indicator grid level, since we assume access to watertight meshes for supervision, as is common practice in learning-based single object reconstruction.

All terms in (5) are differentialable except for the middle one $\frac{\partial \mathbf{p}_{\text{mesh}}}{\partial \chi}$ which involves Marching Cubes. However, this gradient can be effectively approximated by the inverse surface normal [52]:

$$\frac{\partial \mathbf{p}_{\text{mesh}}}{\partial \chi} = -\mathbf{n}_{\text{mesh}} \tag{6}$$

where $\mathbf{n}_{\text{mesh}}$ is the normal of the point $\mathbf{p}_{\text{mesh}}$. Different from MeshSDF [52] that uses the gradients to update the latent code of a pretrained implicit shape representation, our method updates the source point cloud using the proposed differentiable Poisson solver.

**Resampling:** To increase the robustness of the optimization process, we uniformly resample points and normals from the largest mesh component every 200 iterations, and replace all points in the original point clouds with the resampled ones. This resampling strategy eliminates outlier points that drift away during the optimization, and enforces a more uniform distribution of points. We provide an ablation study in supplementary.

**Coarse-to-fine:** To further decrease run-time, we consider a coarse-to-fine strategy during optimization. More specifically, we start optimizing at an indicator grid resolution of $32^3$ for 1000 iterations, from which we obtain a coarse shape. Next, we sample from this coarse mesh and continue optimization at a resolution of $64^3$ for 1000 iterations. We repeat this process until we reach the target resolution ($256^3$) at which we acquire the final output mesh. See also supplementary.

### 3.3 SAP for Learning-based 3D Reconstruction

We now consider the learning-based 3D reconstruction setting in which we train a conditional model that takes a noisy, unoriented point cloud as input and outputs a 3D shape. More specifically, we train the model to predict a a clean oriented point cloud, from which we obtain a watertight mesh using our Poisson solver and Marching Cubes. We leverage the differentiability of our Poisson solver to learn the parameters of this conditional model. Following common practice, we assume watertight meshes as ground truth and consequently supervise directly with the ground truth indicator grid obtained from these meshes. Fig. 1 (bottom) illustrates the pipeline of our architecture for the learning-based surface reconstruction task.

**Architecture:** We first encode the unoriented input point cloud coordinates $\{\mathbf{c}\}$ into a feature $\phi$. The resulting feature should encapsulate both local and global information about the input point

cloud. We utilize the convolutional point encoder proposed in [49] for this purpose. Note that in the following, we will use $\phi_\theta(\mathbf{c})$ to denote the features at point $\mathbf{c}$, dropping the dependency of $\phi$ on the remaining points $\{\mathbf{c}\}$ for clarity. Also, we use $\theta$ to refer to network parameters in general.

Given their features, we aim to estimate both offsets and normals for every input point $\mathbf{c}$ in the point cloud $\{\mathbf{c}\}$. We use a shallow Multi-Layer Perceptron (MLP) $\mathbf{f}_\theta$ to predict the offset for $\mathbf{c}$:

$$\Delta\mathbf{c} = \mathbf{f}_\theta(\mathbf{c}, \phi_\theta(\mathbf{c})) \tag{7}$$

where $\phi(\mathbf{c})$ is obtained from the feature volume using trilinear interpolation. We predict $k$ offsets per input point, where $k \geq 1$. We add the offsets $\Delta\mathbf{c}$ to the input point position $\mathbf{c}$ and call the updated point position $\hat{\mathbf{c}}$. Additional offsets allow us to densify the point cloud, leading to enhanced reconstruction quality. We choose $k = 7$ for all learning-based reconstruction experiments (see ablation study in Table 4). For each updated point $\hat{\mathbf{c}}$, we use a second MLP $\mathbf{g}_\theta$ to predict its normal:

$$\hat{\mathbf{n}} = \mathbf{g}_\theta(\hat{\mathbf{c}}, \phi_\theta(\hat{\mathbf{c}})) \tag{8}$$

We use the same decoder architecture as in [49] for both $\mathbf{f}_\theta$ and $\mathbf{g}_\theta$. The network comprises 5 layers of ResNet blocks with a hidden dimension of 32.

**Training and Inference:** During training, we obtain the estimated indicator grid $\hat{\chi}$ from the predicted point clouds $(\hat{\mathbf{c}}, \hat{\mathbf{n}})$ using our differentiable Poisson solver. Since we assume watertight and noise-free meshes for supervision, we acquire the ground truth indicator grid by running PSR on a densely sampled point clouds of the ground truth meshes with the corresponding ground truth normals. This avoids running Marching Cubes at every iteration and accelerates training. We use the Mean Square Error (MSE) loss on the predicted and ground truth indicator grid:

$$\mathcal{L}_{\text{DPSR}} = \|\hat{\chi} - \chi\|^2 \tag{9}$$

We implement all models in PyTorch [48] and use the Adam optimizer [30] with a learning rate of 5e-4. During inference, we use our trained model to predict normals and offsets, use DPSR to solve for the indicator grid, and run Marching Cubes [35] to extract meshes.

# 4 Experiments

Following the exposition in the previous section, we conduct two types of experiments to evaluate our method. First, we perform single object reconstruction from unoriented point clouds. Next, we apply our method to learning-based surface reconstruction on ShapeNet [9], using noisy point clouds with or without outliers as inputs.

**Datasets:** We use the following datasets for optimization-based reconstruction: 1) Thingi10K [66], 2) Surface reconstruction benchmark (SRB) [60] and 3) D-FAUST [6]. Similar to prior works, we use 5 objects per dataset [19, 23, 60]. For learning-based object-level reconstruction, we consider all 13 classes of the ShapeNet [9] subset, using the train/val/test split from [11].

**Baselines:** In the optimization-based reconstruction setting, we compare against network-based methods IGR [19] and Point2Mesh [23], as well as Screened Poisson Surface Reconstruction[2] (SPSR) [29] on plane-fitted normals. To ensure that the predicted normals are consistently oriented for SPSR, we propagate the normal orientation using the minimum spanning tree [67]. For learning-based surface reconstruction, we compare against point-based Point Set Generation Networks (PSGN) [16], patch-based AtlasNet [20], voxel-based 3D-R2N2 [11], and ConvONet [49], which has recently reported state-of-the-art results on this task. We use ConvOnet in their best-performing setting (3-plane encoders). SPSR is also used as a baseline. In addition, to evaluate the importance of our differentiable PSR optimization, we design another point-based baseline. This baseline uses the same network architecture to predict points and normals. However, instead of passing them to our Poisson solver and calculate $\mathcal{L}_{\text{DPSR}}$ on the indicator grid, we directly supervise the point positions with a bi-directional Chamfer distance, and an L1 Loss on the normals as done in [36]. During inference, we also feed the predicted points and normals to our PSR solver and run Marching Cubes to obtain meshes.

**Metrics:** We consider Chamfer Distance, Normal Consistency and F-Score with the default threshold of $1\%$ for evaluation, and also report optimization & inference time.

[2] We use the official implementation https://github.com/mkazhdan/PoissonRecon.

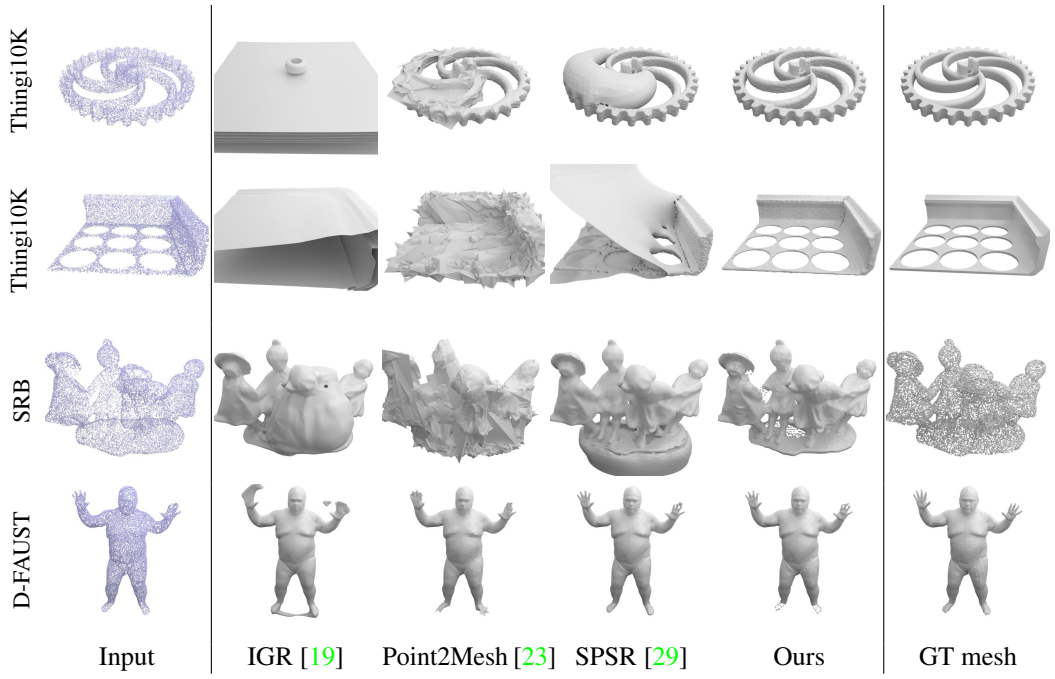


Figure 2: **Optimization-based 3D Reconstruction.** Input point clouds are downsampled for visualization. Note that the ground truth of SRB is provided as point clouds.

| Dataset | Method | Chamfer-$L_1$ (↓) | F-Score (↑) | Normal C. (↑) | Time (s) |
|---|---|---|---|---|---|
| Thingi10K | IGR [19] | 0.440 | 0.505 | 0.692 | 1842.3 |
| | Point2Mesh [23] | 0.109 | 0.656 | 0.806 | 3714.7 |
| | SPSR [29] | 0.223 | 0.787 | 0.896 | 9.3 |
| | **Ours** | **0.054** | **0.940** | **0.947** | 370.1 |
| SRB | IGR [19] | 0.178 | 0.755 | – | 1847.6 |
| | Point2Mesh [23] | 0.116 | 0.648 | – | 4707.9 |
| | SPSR [29] | 0.232 | 0.735 | – | 9.2 |
| | **Ours** | **0.076** | **0.830** | – | 326.0 |
| D-FAUST | IGR [19] | 0.235 | 0.805 | 0.911 | 1857.2 |
| | Point2Mesh [23] | 0.071 | 0.855 | 0.905 | 3678.7 |
| | SPSR [29] | 0.044 | **0.966** | **0.965** | 4.3 |
| | **Ours** | **0.043** | **0.966** | 0.959 | 379.9 |

Table 2: **Optimization-based 3D Reconstruction**. Quantitative comparison on 3 datasets. Normal Consistency cannot be evaluated on SRB as this dataset provides only unoriented point clouds. Optimization time is evaluated on a single GTX 1080Ti GPU for IGR, Point2Mesh and our method.

### 4.1 Optimization-based 3D Reconstruction

In this part, we investigate whether our method can be used for the single-object surface reconstruction task from unoriented point clouds or scans. We consider three different types of 3D inputs: point clouds sampled from synthetic meshes [66] with Gaussian noise, real-world scans [60], and high-resolution raw scans of humans with comparably little noise [6].

Fig. 2 and Table 2 show that our method achieves superior performance compared to both classical methods and network-based approaches. Note that the objects considered in this task are challenging due to their complex geometry, thin structures, noisy and incomplete observations. While some of the baseline methods fail completely on these challenging objects, our method achieves robust performance across all datasets.

In particular, Fig. 2 shows that IGR occasionally creates meshes in free space, as this is not penalized by its optimization objective when point clouds are unoriented. Both, Point2Mesh and our method alleviate this problem by optimizing for the Chamfer distance between the estimated mesh and the

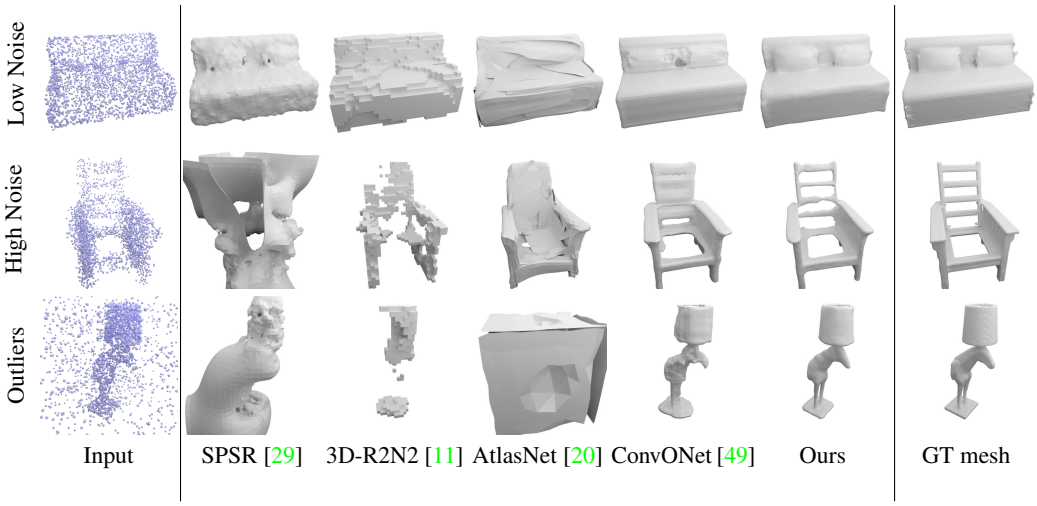


Figure 3: **3D Reconstruction from Point Clouds on ShapeNet.** Comparison of SAP to baselines on 3 different setups. More results can be found in supplementary.

| | (a) Noise=0.005 | | | (b) Noise=0.025 | | | (c) Noise=0.005, Outliers=50% | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Chamfer-$L_1$ | F-Score | Normal C. | Chamfer-$L_1$ | F-Score | Normal C. | Chamfer-$L_1$ | F-Score | Normal C. | Runtime |
| SPSR [29] | 0.298 | 0.612 | 0.772 | 0.499 | 0.324 | 0.604 | 1.317 | 0.164 | 0.636 | - |
| PSGN [16] | 0.147 | 0.259 | - | 0.151 | 0.247 | - | 0.736 | 0.007 | - | 0.010 s |
| 3D-R2N2 [11] | 0.172 | 0.400 | 0.715 | 0.173 | 0.418 | 0.710 | 0.202 | 0.387 | 0.709 | 0.015 s |
| AtlasNet [20] | 0.093 | 0.708 | 0.855 | 0.117 | 0.527 | 0.821 | 1.822 | 0.057 | 0.609 | 0.025 s |
| ConvONet [49] | 0.044 | 0.942 | 0.938 | 0.059 | 0.884 | **0.921** | 0.052 | 0.916 | 0.929 | 0.327 s |
| Ours (w/o $\mathcal{L}_{\text{DPSR}}$) | 0.044 | 0.942 | 0.935 | 0.067 | 0.841 | 0.907 | 0.085 | 0.819 | 0.903 | 0.068 s |
| **Ours** | **0.034** | **0.975** | **0.944** | **0.054** | **0.896** | 0.917 | **0.038** | **0.959** | **0.936** | 0.068 s |

Table 3: **3D Reconstruction from Point Clouds on ShapeNet.** Quantitative comparison between our method and baselines on the ShapeNet dataset (mean over 13 classes).

input point clouds. However, Point2Mesh requires an initial mesh as input of which the topology cannot be changed during optimization. Thus, it relies on SPSR to provide an initial mesh for objects with genus larger than 0 and suffers from inaccurate initialization [23]. Furthermore, compared to both IGR and Point2Mesh, our method converges faster.

While SPSR is even more efficient, it suffers from incorrect normal estimation on noisy input point clouds, which is a non-trivial task on its own. In contrast, our method demonstrates more robust behavior as we optimize points and normals guided by the Chamfer distance. Note that in this *single* object reconstruction task, our method is not able to complete large unobserved regions (e.g., the bottom of the person's feet in Fig. 2 is unobserved and hence not completed). This limitation can be addressed using learning-based object-level reconstruction as discussed next.

To analyze whether our proposed differentiable Poisson solver is also beneficial for learning-based reconstruction, we evaluate our method on the single object reconstruction task using noise and outlier-augmented point clouds from ShapeNet as input to our method. We investigate the performance for three different noise levels: (a) Gaussian noise with zero mean and standard deviation 0.005, (b) Gaussian noise with zero mean and standard deviation 0.025, (c) 50% points have the same noise as in a) and the other 50% points are outliers uniformly sampled inside the unit cube.

### 4.2 Learning-based Reconstruction on ShapeNet

Fig. 3 and Table 3 show our results. Compared to the baselines, our method achieves similar or better results on all three metrics. The results show that, in comparison to directly using Chamfer loss on point positions and L1 loss on point normals, our DPSR loss can produce better reconstructions in all settings as it directly supervises the indicator grid which implicitly determines the surface through the Poisson equation. SPSR fails when the noise level is high or when there are outliers in the input point cloud. We achieve significantly better performances than other representations such as point clouds,

| | $128^3$ | | | | $256^3$ | | | |
|---|---|---|---|---|---|---|---|---|
| | Enc. | Grid | MC | Total | Enc. | Grid | MC | Total |
| ConvONet | **0.010** | 0.280 | **0.037** | 0.327 | **0.010** | 3.798 | **0.299** | 4.107 |
| **Ours** | 0.013 | **0.012** | 0.039 | **0.064** | 0.019 | **0.140** | 0.374 | **0.533** |

| | Chamfer | F-Score | NormalC |
|---|---|---|---|
| Offset 1x | 0.041 | 0.952 | 0.928 |
| Offset 3x | 0.039 | 0.958 | 0.934 |
| Offset 5x | 0.039 | 0.957 | 0.934 |
| Offset 7x | **0.038** | **0.959** | **0.936** |
| 2D Enc. | 0.043 | 0.939 | 0.928 |
| 3D Enc. | **0.038** | **0.959** | **0.936** |

Table 4: **Ablation Study.** Left: Runtime breakdown (encoding, grid evaluation, marching cubes) for ConvONet vs. ours in seconds. Right: Ablation over number of offsets and 2D vs. 3D encoders.

meshes, voxel grids and patches. Moreover, we find that our method is robust to strong outliers. We refer to the supplementary for more detailed visualizations on how SAP handles outliers.

Table 3 also reports the runtime for setting (a) for all GPU-accelerated methods using a single NVIDIA GTX 1080Ti GPU, averaged over all objects of the ShapeNet test set. The baselines [11, 16, 20] demonstrate fast inference time but suffer in terms of reconstruction quality while the neural implicit model [49] attains high quality reconstructions but suffers from slow inference. In contrast, our method is able to produce competitive reconstruction results at reasonably fast inference time. In addition, since ConvONet and our method share a similar reconstruction pipeline, we provide a more detailed breakdown of the runtime at a resolution of $128^3$ and $256^3$ voxels in Table 4. We use the default setup from ConvONet[3]. As we can see from Table 4, the difference in terms of point encoding and Marching Cubes is marginal, but we gain more than $20\times$ speed-up over ConvONet in evaluating the indicator grid. In total, we are roughly $5\times$ and $8\times$ faster regarding the total inference time at a resolution of $128^3$ and $256^3$ voxels, respectively.

### 4.3 Ablation Study

In this section, we investigate different architecture choices in the context of learning-based reconstruction. We conduct our ablation experiments on ShapeNet for the third setup (most challenging).

**Number of Offsets:** From Table 4 we notice that predicting more offsets per input point leads to better performance. This can be explained by the fact that with more points near the object surface, geometric details can be better preserved.

**Point Cloud Encoder:** Here we compare two different point encoder architectures proposed in [49]: a 2D encoder using 3 canonical planes at a resolution of $64^2$ pixels and a 3D encoder using a feature volume with a resolution of $32^3$ voxels. We find that the 3D encoder works best in this setting and hypothesize that this is due to the representational alignment with the 3D indicator grid.

## 5 Conclusion

We introduce Shape-As-Points, a novel shape representation which is lightweight, interpretable and produces watertight meshes efficiently. We demonstrate its effectiveness for the task of surface reconstruction from unoriented point clouds in both optimization-based and learning-based settings. Our method is currently limited to small scenes due to the cubic memory requirements with respect to the indicator grid resolution. We believe that processing scenes in a sliding-window manner and space-adaptive data structures (e.g., octrees) will enable extending our method to larger scenes. Point cloud-based methods are broadly used in real-world applications ranging from household robots to self-driving cars, and hence share the same societal opportunities and risks as other learning-based 3D reconstruction techniques.

[3] To be consistent, we use the Marching Cubes implementation from [56] for both ConvONet and ours.

# References


[1] M. Atzmon and Y. Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[2] A. Badki, O. Gallo, J. Kautz, and P. Sen. Meshlet priors for 3d mesh reconstruction. *arXiv.org*, 2001.01744, 2020. 2, 3

[3] Y. Ben-Shabat and S. Gould. Deepfit: 3d surface fitting via neural network weighted least squares. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 3

[4] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer. Nesti-net: Normal estimation for unstructured 3d point clouds using convolutional neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[5] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. on Visualization and Computer Graphics (VCG)*, 1999. 3

[6] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black. Dynamic FAUST: registering human bodies in motion. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 7

[7] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral methods: fundamentals in single domains*. Springer Science & Business Media, 2007. 4

[8] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 3

[9] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 1512.03012, 2015. 6

[10] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3

[11] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 2, 6, 8, 9

[12] A. Dai, C. Diller, and M. Nießner. SG-NN: sparse generative neural networks for self-supervised scene completion of RGB-D scans. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[13] A. Dai, C. Diller, and M. Nießner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[14] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[15] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. on Graphics*, 1994. 3

[16] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. *arXiv.org*, abs/1612.00603, 2016. 2, 6, 8, 9

[17] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[18] K. Genova, F. Cole, A. Sud, A. Sarna, and T. A. Funkhouser. Local deep implicit functions for 3d shape. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[19] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman. Implicit geometric regularization for learning shapes. In *Proc. of the International Conf. on Machine learning (ICML)*, 2020. 3, 6, 7

[20] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. AtlasNet: A papier-mâché approach to learning 3d surface generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 6, 8, 9

[21] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra. Pcpnet learning local shape properties from raw point clouds. In *Computer Graphics Forum*, 2018. 3

[22] K. Gupta and M. Chandraker. Neural mesh flow: 3d manifold mesh generation via diffeomorphic flows. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[23] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or. Point2mesh: a self-prior for deformable meshes. In *ACM Trans. on Graphics*, 2020. 3, 6, 7, 8

[24] C. Jiang, J. Huang, A. Tagliasacchi, and L. J. Guibas. Shapeflow: Learnable deformation flows among 3d shapes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[25] C. Jiang, P. Marcus, et al. Hierarchical detail enhancing mesh-based shape generation with 3d generative adversarial network. *arXiv preprint arXiv:1709.07581*, 2017. 2

[26] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser. Local implicit grid representations for 3d scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3

[27] C. M. Jiang, J. Huang, K. Kashinath, Prabhat, P. Marcus, and M. Nießner. Spherical cnns on unstructured grids. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019. 2

[28] M. M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, 2006. 3, 4

[29] M. M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Trans. on Graphics*, 32(3):29, 2013. 3, 4, 6, 7, 8

[30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Machine learning (ICML)*, 2015. 6

[31] J. E. Lenssen, C. Osendorfer, and J. Masci. Deep iterative surface normal estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[32] J. Li and A. O. Hero. A spectral method for solving elliptic equations for surface reconstruction and 3d active contours. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 3, pages 1067–1070. IEEE, 2001. 4

[33] Y. Liao, S. Donne, and A. Geiger. Deep marching cubes: Learning explicit surface representations. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 3

[34] S. Lionar, D. Emtsev, D. Svilarkovic, and S. Peng. Dynamic plane convolutional occupancy networks. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021. 3

[35] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Trans. on Graphics*, 1987. 4, 6

[36] Q. Ma, S. Saito, J. Yang, S. Tang, and M. J. Black. Scale: Modeling clothed humans with a surface codec of articulated local elements. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 6

[37] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[38] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3

[39] M. Meshry, D. B. Goldman, S. Khamis, H. Hoppe, R. Pandey, N. Snavely, and R. Martin-Brualla. Neural rerendering in the wild. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[40] G. Metzer, R. Hanocka, D. Zorin, R. Giryes, D. Panozzo, and D. Cohen-Or. Orienting point clouds with dipole propagation. *ACM Trans. on Graphics*, 2021. 3

[41] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 1

[42] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv.org*, 2003.08934, 2020. 3

[43] M. Niemeyer and A. Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[44] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3

[45] M. Oechsle, L. Mescheder, M. Niemeyer, T. Strauss, and A. Geiger. Texture fields: Learning texture representations in function space. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1

[46] M. Oechsle, S. Peng, and A. Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. *arXiv preprint arXiv:2104.10078*, 2021. 1

[47] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 6

[49] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional occupancy networks. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 1, 3, 6, 8, 9

[50] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[51] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2

[52] E. Remelli, A. Lukoianov, S. R. Richter, B. Guillard, T. Bagautdinov, P. Baque, and P. Fua. Meshsdf: Differentiable iso-surface extraction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 5

[53] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

[54] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[55] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

[56] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu. scikit-image: image processing in python. *PeerJ*, 2014. 9

[57] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2

[58] W. Wang, Q. Xu, D. Ceylan, R. Mech, and U. Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 3

[59] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. on Graphics*, 2019. 2

[60] F. Williams, T. Schneider, C. Silva, D. Zorin, J. Bruna, and D. Panozzo. Deep geometric prior for surface reconstruction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 3, 6, 7

[61] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 2

[62] G. Yang, X. Huang, Z. Hao, M. Liu, S. J. Belongie, and B. Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2

[63] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[64] Z. Yang, Y. Chai, D. Anguelov, Y. Zhou, P. Sun, D. Erhan, S. Rafferty, and H. Kretzschmar. Surfelgan: Synthesizing realistic sensor data for autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[65] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, and Y. Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1

[66] Q. Zhou and A. Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 6, 7

[67] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 6

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We have performed experiments in Section 4 to study properties that we discussed in the abstract and introduction, for example high-quality shapes and fast inference.
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We have stated all assumptions for the theoretical results in Section 3 and Section 4.
   (b) Did you include complete proofs of all theoretical results? [Yes] Please refer to Section 3 as well as the supplementary material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We state in Section 1 that we will release our code upon publication, and provide implementation details in both main papers and supplementary.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We partially provide the training details in Section 3.3, and all other details in supplementary.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We don't require running experiments multiple times so do not consider the impact of random seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We provide that in Section 1 and Section 4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite or include links of the existing codes, data or models everywhere in paper.
   (b) Did you mention the license of the assets? [No] We only use assets that allow public usage.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We have provided URL for the baseline codes.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We only use publicly available datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]