Small Steps and Level Sets: Fitting Neural Surface Models with Point Guidance

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

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

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

Abstract

A neural signed distance function (SDF) is a convenient shape representation for many tasks, such as surface reconstruction, editing and generation. However, neural SDFs are difficult to fit to raw point clouds, such as those sampled from the surface of a shape by a scanner. A major issue occurs when the shape's geometry is very different from the structural biases implicit in the network's initialization. In this case, we observe that the standard loss formulation does not guide the network towards the correct SDF values. We circumvent this problem by introducing guiding points, and use them to steer the optimization towards the true shape via small incremental changes for which the loss formulation has a good descent direction. We show that this point-guided homotopy-based optimization scheme facilitates a deformation from an easy problem to the difficult reconstruction problem. We also propose a metric to quantify the difference in surface geometry between a target shape and an initial surface, which helps indicate whether the standard loss formulation is guiding towards the target shape. Our method outperforms previous state-of-the-art approaches, with large improvements on shapes identified by this metric as particularly challenging.

1. Introduction

Point cloud reconstruction using a neural signed distance function (SDF) is a popular approach due to its simplicity and flexibility. Such methods show impressive performance improvements for cases with significant noise, outliers, missing regions or misalignment [20]. However, they are still unable to outperform classical methods on clean and relatively dense input data [20, 25], especially when the shape exhibits challenging geometry. For example, thin surfaces near each other (see Fig. 1 bottom) or large concave regions with small openings (see Fig. 1 top).

We address the task of reconstruction from raw point clouds, i.e., without ground-truth normal vectors. We observe that the standard loss formulation used for the task does not give a good descent direction[1] towards the target SDF values, especially when the initialization does not align with the target geometry. Since most methods initialize to a sphere to bias the optimization result towards a bounded shape, and rely on this bias to find a good descent direction, their performance can be poor if the input shape has very different geometry to a sphere (see Fig. 1).

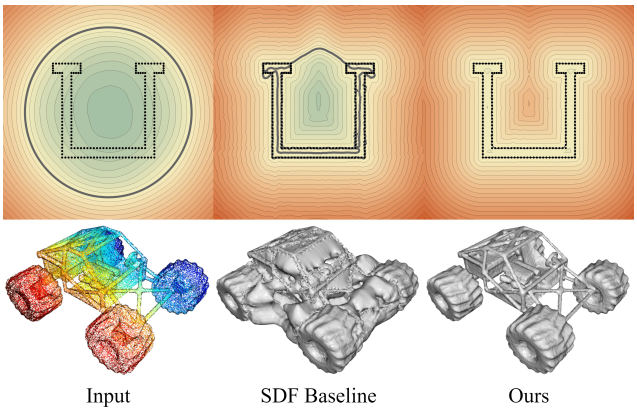

Figure 1. Given a point cloud without ground-truth normals (left), neural SDF optimization for surface fitting frequently gets trapped in local minima (middle). Note that these two shapes have geometry very different to a sphere, which causes the loss right after spherical initialization to not provide a good descent direction towards the target shape (see Sec. 3.3). Our point-guided homotopy-based approach provides a principled way to avoid this issue and improve reconstruction quality (right).

Instead we propose a point-guided SDF (PG-SDF) approach, where we introduce guiding points to represent an intermediate surface for the optimization to target. By incrementally moving the guiding points towards the sampled input and optimizing the network to their new location, we ensure that the loss formulation gives a good descent direction during optimization. We eventually converge the guiding points to the input points, leading the optimization stably towards the target SDF. We also propose

[1]To avoid confusion, in this paper we use the term *normal direction* to refer to vectors in the normalized gradient field of the SDF and *descent direction* for the negative gradient of the loss function.

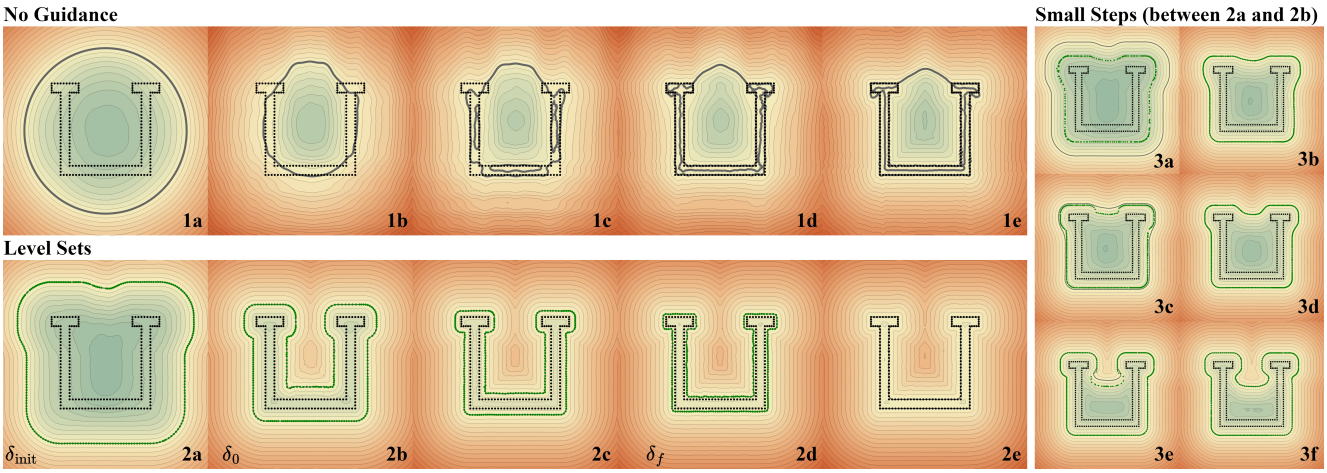

Figure 2. Reconstructing a 2D surface using a neural SDF, without (**1a–e**) and with (**2a–e,3a–f**) our proposed point guidance. Note that our point guidance has two parts, converging to level sets (**2**) and taking small steps in between (**3**). The input points $\mathcal{X}$ are black, the guiding points $\mathcal{Y}$ are green, and the network's level sets are drawn as contour lines with the zero level set $\mathcal{R}$ in bold. **(1a–b)** The network deforms from the sphere initialization to one whose zero level set is close to $\mathcal{X}$. **(1c)** The network only makes local changes near $\mathcal{X}$, gradually interpolating $\mathcal{X}$ by adding complexity. **(1d–e)** $\mathcal{X}$ is fully interpolated, but the result is unsatisfactory. **(2a)** Our initialization. **(2b–d)** Our method after convergence to each intermediate level set. **(2e)** Our method after optimizing on the input points. **(3a–f)** The small steps taken by our method while reaching the first intermediate level set of radius δ_0. Each row shows a single step, where the guiding points are first moved a short distance (left) and then the network is optimized to match this (right).

a metric—the percentage of self-intersecting outward normals (SION%)—to quantify the difference between the surface geometry of a target shape and that of another surface, such as the sphere initialization. We use this metric to determine which shapes are likely to be difficult for neural SDF methods, and why the optimization steps in our method are robust. Our contributions are:

1. a point-guided optimization approach for surface reconstruction from point clouds without normals;
2. a metric to quantify differences between surfaces, which can be used to identify the most challenging shapes to reconstruct; and
3. an interpretation of the algorithm as a homotopy method, providing a justification for the design decisions and insight into the failure modes of existing approaches.

2. Related Work

Surface reconstruction from point clouds is a fundamental problem in computer vision. We restrict our attention to methods that optimize using generic priors, rather than methods that learn reconstruction priors from a dataset [19, 45] and are sensitive to domain shift.

Early approaches [1, 7, 10, 24] directly triangulated point clouds, with provable guarantees given dense input points with no defects. However, the limitation of vertices being input points make them impractical for even small levels of imperfection [6], and the result is often not watertight. A less restrictive class of approaches is implicit field methods [9, 22, 23, 32, 34], where an implicit function is defined over the domain to represent the reconstructed surface. Such methods often use a carefully defined set of basis functions, and optimize for an energy function with respect to the basis function parameters. However they usually rely on normal vectors as inputs, allowing efficient optimization with linear solvers, such as Poisson Surface Reconstruction [23] and its screened version [22].

Neural implicit methods, which were popularised by DeepSDF [35] and Occupancy Networks [29], are promising as they provide more modeling and optimization flexibility. Rather than creating basis functions tailored to the input, the network is fixed and can adapt its representational ability through optimization. Furthermore, as they are optimized by gradient descent, they can work with any differentiable loss. This flexibility has led to learned shape spaces that allow for strong learned biases and interpolation [21, 26, 35, 47]. However, while many neural methods are able to perform better than non-neural approaches when the input has defects, they often perform worse when the data is clean and dense [20]. A major issue is that gradient descent optimization is not guaranteed to converge to a global optimum. Furthermore, many methods for the surface reconstruction task [2–4, 15, 41] use losses that are difficult for gradient descent to optimize. As the reason these losses are hard to optimize is fundamental to our method, we examine this issue and related loss formulations in Sec. 3.3.

Our method follows a line of work that attempts to

improve upon these issues for reconstruction from points clouds without normals. There are three major methods that are competitive for this task, DiGS [5], SAP [39] and OG-INR [25]. DiGS and OG-INR are neural SDF methods. In order to avoid the difficulties of SDF optimization, DiGS [5] changes the optimization landscape by adding strong regularization at the start, and slowly annealing the regularization to gently change the optimization landscape back to the correct one. OG-INR [25] avoids local minima by first discretizing the problem and solving for inside–outside labels. It then uses this discrete solution to get an approximate SDF that makes it easier to optimize the original SDF. SAP [39] does not use neural methods or SDFs, and instead minimizes a chamfer loss on points and normals directly. This is related to our approach as we also use points and SDF-derived normals to guide our SDF optimization. However, rather than using a chamfer loss, which we find gets trapped in bad local minima, we use a more nuanced strategy to move our guiding points, outlined in Sec. 4.1.

Another highly related approach is Point2Mesh [17], which deforms an enclosing mesh onto the point cloud. Like SAP, they use a chamfer loss to guide the deformation, however they note that this loss does not easily enter concavities. To address this, they introduce a beam gap loss, which guides mesh vertices towards input points in the inward normal direction if they are not already well matched by chamfer distance. Our deformation strategy, in contrast, is based purely on explicitly matching points in the inward normal direction. Our approach also avoids a major shortcoming of Point2Mesh, the reconstruction output must have the same genus as the initial mesh.

3. Preliminaries

3.1. Notation and Definitions

We denote the Euclidean inner product as $\langle p, q \rangle = p^\mathsf{T} q$, the Euclidean distance as $d(p, q) = \|p - q\|_2$, a unit vector as $\hat{v} = v/\|v\|$, the cosine similarity between two vectors as $s(p, q) = \langle \hat{p}, \hat{q} \rangle$, and the distance from a point p to a set A by $d(p, A) = d_A(p) = \inf_{q \in A} d(p, q)$.

The chamfer distance d_C is a common measure of the distance between two discrete point sets P and Q, defined by the average of two one-sided chamfer distances

$$d_C(P, Q) = \frac{1}{2}\left(d_{\vec{C}}(P, Q) + d_{\vec{C}}(Q, P)\right) \tag{1}$$

each averaging the distance over one set to the other set

$$d_{\vec{C}}(P, Q) = \frac{1}{|P|} \sum_{p \in P} d(p, Q). \tag{2}$$

The chamfer distance can also be used between two surfaces, A and B, in which case the chamfer distance is taken over uniform samplings, $U_A \sim U(A)$ and $U_B \sim U(B)$.

A signed distance function (SDF) $F_\mathcal{V} : \mathcal{D} \to \mathbb{R}$ of a bounded, open volume $\mathcal{V} \subset \mathcal{D}$ is a scalar function defined everywhere on $\mathcal{D}$. $F_\mathcal{V}(z)$ is the distance from z to the closest point on the boundary $\partial\mathcal{V}$, with a negative sign if $z \in \mathcal{V}$

$$F_\mathcal{V}(z) = (-1)^{[\![z \in \mathcal{V}]\!]} d(z, \partial\mathcal{V}) \tag{3}$$

where the Iverson bracket $[\![X]\!]$ is 1 if X is true and 0 otherwise. The level set L_a is the set of points that have a signed distance of a from $\partial\mathcal{V}$, i.e.,

$$L_a = \{z \in \mathcal{D} \mid F_\mathcal{V}(z) = a\}. \tag{4}$$

The exterior level set Ω_a ($a > 0$) is the subset of L_a such that for each $p \in L_a$ there is a connected path from a point on the boundary of $\mathcal{D}$ to p that only intersects L_a at p.

3.2. Problem Formulation

Let $\mathcal{X} = \{x_i\}_{i \in I}$ be a point cloud sampled from a surface $\mathcal{S} \subset \mathcal{D}$ where $\mathcal{D} \subset \mathbb{R}^3$ is compact. We often refer to $\mathcal{S}$ as the target shape, and assume that it is watertight, i.e., $\mathcal{S} = \partial\mathcal{V}$ of some open volume $\mathcal{V} \subset \mathcal{D}$. We wish to reconstruct $\mathcal{S}$ from the point cloud $\mathcal{X} = \{x_i\}$. Note that no per-point surface normals or scanner information is provided.

We choose to represent our reconstructed surface $\mathcal{R} \subset \mathcal{D}$ using an implicit neural representation Φ. Specifically we define $\mathcal{R} = \{z \in \mathcal{D} \mid \Phi(z, \theta) = 0\}$ where θ are the parameters of the neural network Φ. Thus we want to minimize the chamfer distance between $\mathcal{S}$ and $\mathcal{R}$. Since we do not know $\mathcal{S}$, we instead compute the distance to our input points $d_C(\mathcal{X}, \mathcal{R}) = d_C(\mathcal{X}, U_\mathcal{R})$, where $U_\mathcal{R}$ is a uniform sampling from $\mathcal{R}$. Furthermore we choose to optimize the implicit function Φ to approximate the true signed distance function $F_\mathcal{V}$ of the volume $\mathcal{V}$. As $\mathcal{V}$ is unknown, during optimization we care about ensuring that Φ is a valid SDF to some open volume $\mathcal{U}$ for which $\mathcal{X}$ is an approximate sampling, i.e., $\Phi \approx F_\mathcal{U}$ and $d_C(\mathcal{X}, \partial\mathcal{U})$ is small (note $\partial\mathcal{U} = \mathcal{R}$). Thus our overall optimization problem is

$$\begin{aligned} \text{minimize}_\theta \quad & d_C(\mathcal{X}, \partial\mathcal{U}) \\ \text{subject to} \quad & \Phi(x, \theta) = F_\mathcal{U}(x). \end{aligned} \tag{5}$$

Why SDFs? We choose SDFs over other popular implicit functions like indicator or occupancy functions [22, 23, 29, 38] or unsigned distance functions (UDFs) [12, 16, 49] for several reasons. First, SDFs and UDFs constrain the possibilities for Φ, and are fairly smooth (unlike indicator functions). This helps with regularizing $\mathcal{R}$ and ensures that they can be represented by neural networks, which are biased to be smooth. Previous works have shown that neural SDFs empirically perform better than neural occupancy functions [51]. Second, SDFs allow for distance and direction to the surface queries, which we make use of in our algorithm. Last, the signed aspect of SDFs guarantee the watertight criteria compared to UDF, and make it easier to mesh using algorithms like marching cubes [27].

3.3. Neural SDF Formulation

Since neural SDFs are coordinate networks, each coordinate is processed independently. During gradient descent each coordinate seeks to minimize its loss locally, and is only affected by changes in nearby coordinates due to the network's inherent smoothness bias. Thus information propagation through the domain (from the smoothness bias) is slow, and optimization is only effective if the loss provides clear guidance. Specifically, if the descent direction of the loss with respect to its SDF input value at a coordinate gives an approximate descent direction towards the target SDF value at that coordinate. However whether or not the loss provides a good descent direction depends on whether the geometry of the current state of the network aligns with the geometry of the target shape (see Fig. 2 top left for when they do not align and thus optimization gets stuck).

We now discuss this observation in relation to currently used neural SDF losses. When optimizing neural SDFs for point clouds without normals, the main losses are

$$\mathcal{L}_{\text{zls}}(\theta, \mathcal{X}) = \sum_{x \in \mathcal{X}} |\Phi(x, \theta)| \tag{6}$$

$$\mathcal{L}_{\text{eik}}(\theta) = \sum_{z \in \mathcal{D}} |\|\nabla_z \Phi(z, \theta)\|_2 - 1| \tag{7}$$

$$\mathcal{L}_{\text{pull}}(\theta, \mathcal{X}) = \sum_{z \in \mathcal{D}} \|x^\star_{\mathcal{R}}(z) - x^\star_{\mathcal{X}}(z)\|_2 \tag{8}$$

where $x^\star_{\mathcal{R}}(z) = z - \Phi(z)\widehat{\nabla_z \Phi(z)}$ and $x^\star_{\mathcal{X}}(z) = \arg\min_{x \in \mathcal{X}} d(z, x)$. The first two losses were proposed by IGR [15] and the last one was proposed by Neural Pull [4] (further details in the supplement). The zero level set loss $\mathcal{L}_{\text{zls}}$ gives a perfect descent direction to the target SDF values for the points in $\mathcal{X}$. However, for other points in the domain, the loss formulation may not give a descent direction towards the target SDF value. The Eikonal loss $\mathcal{L}_{\text{eik}}$ guides towards scaling the SDF gradient, and does not change its direction unless forced to by the optimization of nearby points. Thus for a point $z \in \mathcal{D}$, if the network's current normal directions in the region around z is in the opposite direction to what the target SDF normals are around z, then at z this loss does not guide towards the target SDF values. The neural pull loss $\mathcal{L}_{\text{pull}}$ has two value–gradient pairs that will minimize its loss (one where the normal points towards $x^\star_{\mathcal{X}}$ and one where it points away), so the loss will guide towards the closer one, which may not be the target.

As a result, an important part of these methods is spherical initialization [2], where the network is initialized roughly to the SDF of a sphere. If the shape's geometry is similar to a sphere, then the initialized normal directions in the network are mostly in the correct direction, so $\mathcal{L}_{\text{eik}}$ and $\mathcal{L}_{\text{pull}}$ guide towards the correct SDF value. When the shape's geometry is very different to a sphere resulting in large regions where the initialized normal directions are very different to the target normal directions, then the loss does not provide a good descent direction in those regions. As a result, optimization often converges to a local minimum close to the initialization, such as in Fig. 2 (top left).

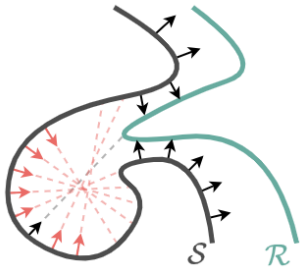

Figure 3. Surface $\mathcal{S}$ and surrounding surface $\mathcal{R}$, with outward normal directions for some points on $\mathcal{S}$ shown, and which ones are SIONs relative to $\mathcal{R}$ shown in red.

3.4. Shape Reachability

From the analysis in the previous subsection, we are interested in whether the current state of a network provides similar normal directions to the SDF of the target shape, so that the loss provides a good descent direction towards the target SDF values. Let us assume that the current state of the network is roughly the SDF to some surface $\mathcal{R}$ surrounding the target shape's surface $\mathcal{S}$. One way of quantifying whether the normal directions are similar is to determine if points on $\mathcal{S}$ can "reach" $\mathcal{R}$ by travelling in their outward normal direction (see Fig. 3). If $\mathcal{S}$ has large concavities and the outward surface normal of most points in that concavity does not reach $\mathcal{R}$ before it self-intersects, then the current normal directions of the network are likely to be poor.

More formally, if $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ are samplings of $\mathcal{S}$ and $\mathcal{R}$ and for each $p \in P_{\mathcal{S}}$ we have a known outward unit normal n_p, then for a point $p \in P_{\mathcal{S}}$ we can construct around the outward normal a thin cylinder

$$\mathcal{C}_p = \{z \in \mathcal{D} \mid \|p + tn_p - z\| \leq \varepsilon, \langle z - p, n_p \rangle \geq 0\}. \tag{9}$$

By the assumption of $\mathcal{R}$ surrounding $\mathcal{S}$ and $P_{\mathcal{R}}$ being a dense sampling of $\mathcal{R}$, we note that $\mathcal{C}_p \cap P_{\mathcal{R}} \neq \emptyset$. Then we consider p to have a *self-intersecting outward normal (SION)* with respect to $P_{\mathcal{R}}$ if its normal direction intersects its own shape before it intersects $\mathcal{R}$: $\exists p' \in P_{\mathcal{S}}$ such that $p' \in \mathcal{C}_p$ and $\forall q \in \mathcal{C} \cap P_{\mathcal{R}}$ we have that $\|p - p'\|_2 < \|p - q\|$. We then use the percentage of points in $p_{\mathcal{S}}$ that have SIONs with respect to $P_{\mathcal{R}}$ as a measure of how dissimilar the geometry of $\mathcal{S}$ is to $\mathcal{R}$, which we denote SION$(\mathcal{S}; \mathcal{R})$, where a high value indicates dissimilarity.

When $\mathcal{R}$ is not specified and $\mathcal{S}$ is clear from context, we will use SION as a shorthand for SION$(\mathcal{S}; \mathcal{R})$, with $\mathcal{R}$ assumed to be any sphere surrounding $\mathcal{S}$. The two point clouds in Fig. 1 have a high SION, which indicates that their geometry is very different to a sphere and that optimization

from a spherical initialization is likely to be difficult.

4. Point-Guided SDF

The key idea of our method is to slowly converge the surface of reconstruction $\mathcal{R}$ towards the input points $\mathcal{X}$ until it interpolates $\mathcal{X}$, while maintaining Φ as a valid SDF to the volume enclosed by $\mathcal{R}$ (see Fig. 2). We do this by guiding $\mathcal{R}$ towards level sets (Fig. 2:2a–e) and taking small steps towards reaching each level set (Fig. 2:3a–f, Fig. 4). At each step we move guiding points $\mathcal{Y}$, which represent the surface $\mathcal{R}$, a small distance towards $\mathcal{X}$ (Fig. 4c) and then optimize Φ to interpolate the new $\mathcal{Y}$ positions, thus defining the new surface of reconstruction $\mathcal{R}$ over which $\mathcal{Y}$ is resampled (Fig. 4d). Our full algorithm is shown in Algorithm 1.

Since each movement of the guiding points is small, the geometry of the moved guiding points $\mathcal{Y}$ should be similar to the network's current surface $\mathcal{R}$, and SION($\mathcal{Y}$; $\mathcal{R}$) should be low. Thus optimizing the network to reconstruct the updated $\mathcal{Y}$ should be robust (unlikely to converge to local minima) as the optimization is likely to have a clear descent direction from the loss at most points.

In Sec. 4.1 we outline how to move the guiding points $\mathcal{Y}$ one step closer to the input points $\mathcal{X}$ (Fig. 4a–c). In Sec. 4.2 we outline how to optimize the network Φ to interpolate the new guiding point locations (Fig. 4d). Finally, in Sec. 4.3 we analyze our method within the homotopy framework.

Preprocessing $\mathcal{X}$. Following standard practice, we first center and (isotropically) scale the shape to fit in a centered ball. After scaling, we calculate the point cloud's approximate sampling radius $\delta_{\mathcal{X}}$ conservatively by calculating the distance to the four closest neighbors for each $x \in \mathcal{X}$ and averaging the highest 5% of all such distances. We use this radius extensively in our method.

Initialization. We first initialize our network to the exterior level set of radius $\delta_{\text{init}} = 16\delta_{\mathcal{X}}$. We provide intuition into why we can do this robustly in Sec. 4.3, and provide details on how to make this more efficient in the supplement.

4.1. Moving the Guiding Points

We now describe how to move the guiding points $y \in \mathcal{Y}$ towards the input points $\mathcal{X}$. At a high level, each step we move $\mathcal{Y}$ closer to an exterior level set of radius δ, Ω_δ, and when $\mathcal{R}$ has converged to Ω_δ we reduce δ (see Algorithm 1).

Since we are moving $y \in \mathcal{Y}$ to guide the surface $\mathcal{R}$, we only move each y in the inwards normal direction of $\mathcal{R}$. Next, since we are moving towards an exterior level set Ω_δ of $\mathcal{X}$, for each $y \in \mathcal{Y}$ we find the closest input point in the inwards normal direction $x^\star(y) \in \mathcal{X}$, and use this to determine the target point $\omega^\star(y)$ on Ω_δ that we move towards. Finally, since we want to move the surface $\mathcal{R}$ slowly, we have a maximum step size for each movement.

Closest point in the inward direction. We first define $x^\star(y)$ (see Fig. 4a–b). As we are moving in the inward normal direction, i.e., $-n_y$ where $n_y = \widehat{\nabla_y \Phi(y)}$, we want to move y based on the closest point $x \in \mathcal{X}$ in the inward normal direction $-n_y$. We define the inward normal definition by setting a cosine similarity threshold $a_y \geq 0$ to define a cone C_y supported at y emanating in the direction $-n_y$,

$$C_y = \{z \mid s(-n_y, z - y) \geq a_y\}. \tag{10}$$

However as y gets very close to points in $\mathcal{X}$, it is unlikely that any of those nearby points stay within this cone. Thus we also consider the half ball H_y of radius $2\delta_{\mathcal{X}}$ containing points close to y

$$H_y = \{z \mid \|x - y\|_2 \leq 2\delta_{\mathcal{X}} \wedge s(-n_y, z - y) \geq 0\}. \tag{11}$$

Thus our target point $x^\star(y)$ is given by

$$x^\star(y) = \underset{x \in \mathcal{X} \cap (C_y \cup H_y)}{\arg\min} ||x - y||_2. \tag{12}$$

Moving towards Ω_δ. As $x^\star(y)$ is our closest (inward) point, a reasonable location for a point in Ω_δ can be computed by moving δ away from $x^\star$ towards y

$$\omega^\star(y) = (\|x^\star - y\|_2 - \delta)(\widehat{x^\star - y}). \tag{13}$$

We move towards $\omega^\star$ with a maximum step size of s_m so that the movement step is small and easy to optimize. Thus our final point y' (see Fig. 4b) is

$$y' = y - \text{clip}(\langle \omega^\star(y), -n_y \rangle, s_m) n_y \tag{14}$$

where $\text{clip}(a, \varepsilon)$ projects a to the interval $[-\varepsilon, \varepsilon]$.

Reducing δ. After initialization, we set $\delta = \delta_0$ as our next target level set. Once $\mathcal{Y}$ has finished moving towards Ω_δ and $\mathcal{R}$ has converged, we decrease the current δ by half and start moving $\mathcal{Y}$ towards the new exterior level set. We do this until δ reaches δ_f. We use $\delta_0 = 4\delta_{\mathcal{X}}$, $\delta_f = \delta_{\mathcal{X}}$. We then abandon $\mathcal{Y}$ and optimize directly on $\mathcal{X}$.

4.2. Optimizing the Network

We now have the points $\mathcal{Y}$ to guide where our new surface should be. Furthermore, we have an approximate normal n_y for each $y \in \mathcal{Y}$ given by Φ, whose direction should be useful as the SION between the new position of the points relative to the previous positions is low. We use this to estimate an approximate SDF

$$\tilde{F_{\mathcal{V}}}(z) = (-1)^{[\![s(z - y(z), n_{y(z)}) \geq 0]\!]} d(z, y(z)) \tag{15}$$

where $y(z) = \arg\min_{y \in \mathcal{Y}} d(z, y)$ and $s(p, q) = \langle \hat{p}, \hat{q} \rangle$ is cosine similarity. We guide towards these values using

$$\mathcal{L}_{\widetilde{\text{sdf}}}(\theta, \mathcal{X}) = \sum_{z \in \mathcal{D}} \|\Phi(z, \theta) - \tilde{F_{\mathcal{V}}}(z)\|_2. \tag{16}$$

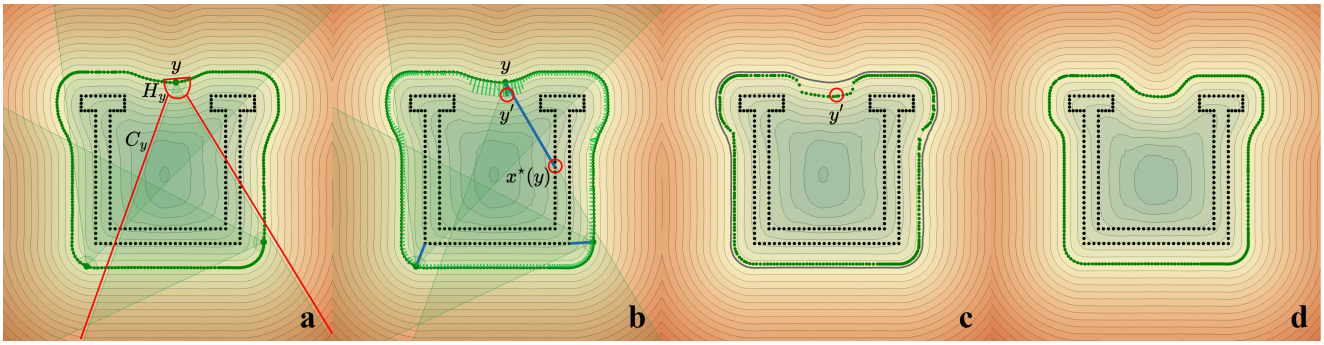

Figure 4. One iteration of Algorithm 1, steps 6–8. **(a)** Given the current value of our network Φ (the contours) and our guiding points $\mathcal{Y}$ (the green points), for each $y \in \mathcal{Y}$ we calculate a half ball H_y and a cone C_y in the inside normal direction to find an input points $x \in \mathcal{X}$ (black points) to move towards (shown for 3 points, with one highlighted in red). **(b)** We find the closest input point $x^\star(y) \in \mathcal{X}$ within $H_y \cap C_y$, and use this to calculate the new position y' for each $y \in \mathcal{Y}$ (shown for all points). **(c)** The result after each $y \in \mathcal{Y}$ has been moved. **(d)** The network Φ is optimized on the new guiding points, and $\mathcal{Y}$ is resampled to densely cover the new zero level set $\mathcal{R}$.

Furthermore, our input points $\mathcal{X}$ should be at distance δ from $\mathcal{Y}$ and should be inside, so we add a loss term

$$\mathcal{L}_{\text{input}}(\theta, \mathcal{X}, \delta) = \sum_{x \in \mathcal{X}} |\Phi(x, \theta) + \delta|. \tag{17}$$

Thus our overall loss term formulation is

$$\mathcal{L}(\theta, \mathcal{X}, \mathcal{Y}, \delta) = \mathcal{L}_{\text{zls}}(\theta, \mathcal{Y}) + \mathcal{L}_{\text{input}}(\theta, \mathcal{X}, \delta) + \mathcal{L}_{\text{eik}}(\theta) + \mathcal{L}_{\widetilde{\text{sdf}}}(\theta, \mathcal{Y}) + \mathcal{L}_{\text{pull}}(\theta, \mathcal{Y}). \tag{18}$$

Relevant details (e.g., loss term weightings, domain sampling procedure, optimizer settings) are in the supplement.

Finally, we resample $\mathcal{Y}$ in the new zero level set $\mathcal{R}$ (Fig. 4f) so that it densely samples it. After this process has finished (i.e., when Φ has converged for $\delta = \delta_f$), we optimize Φ on the input points $\mathcal{X}$ using the loss in Eq. (18). Here we do not use the approximate SDF term, as $\mathcal{X}$ is likely to have more imperfections that our guiding points.

Algorithm 1 Point-Guided SDF

procedure PG-SDF($\mathcal{X}$)
  Center $\mathcal{X}$, scale $\mathcal{X}$ and compute sampling radius $\delta_{\mathcal{X}}$
  $\mathcal{Y}, \mathcal{N}_{\mathcal{Y}}, \Phi \leftarrow$ Initialization($\mathcal{X}$, δ_{init})
  $\delta \leftarrow \delta_0$
  while $\delta \geq \delta_f$ **do**
    Move $\mathcal{Y}$ using Eq. (14) ▷*See Sec. 4.1*
    Optimize Φ to match $\mathcal{Y}$ ▷*See Sec. 4.2*
    Resample $\mathcal{Y}$ on new $\mathcal{R}$
    Query Φ for normals $\mathcal{N}_{\mathcal{Y}}$
    if Φ's zero level set has not changed **then**
      $\delta \leftarrow \frac{1}{2}\delta$
    end if
  end while
  Optimize Φ on $\mathcal{X}$ ▷*See Sec. 4.2*
  return Φ
end procedure

4.3. Interpretation as a Homotopy Method

One way to analyze this method is in the context of a homotopy method [33]. Here, we wish to solve a difficult problem $M(\theta) = 0$ for θ, and instead consider the homotopy map $H(\theta, t) = tM(\theta) + (1-t)N(\theta)$ where $N(\theta) = 0$ is easy to solve and $t \in [0, 1]$. This forms a homotopy between the problems $M(\theta) = H(\theta, 1)$ and $N(\theta) = H(\theta, 0)$ as we can now continuously deform one problem into the other by varying t. By taking small steps along the *zero path*, the trajectory of points (θ, t) for which $H(\theta, t) = 0$, we can reach the solution to $M(\theta)$.

For our method, we start by solving the easier problem $\mathcal{L}(\theta, \mathcal{Y}_0) = 0$ where $\mathcal{Y}_0$ is our initial set of guiding points. Our goal is to solve the hard problem $\mathcal{L}(\theta, \mathcal{X}) = 0$. To see that our method is a homotopy method, note that our guiding points $\mathcal{Y}$ are facilitating a topological homotopy from $\mathcal{Y}_0$ to our input points $\mathcal{X}$. Thus we can parameterize our zero path using $\mathcal{Y}$. It is given by $(\theta(\mathcal{Y}), t(\mathcal{Y}))$ where

$$\theta(\mathcal{Y}) \text{ satisfies } \mathcal{L}(\theta(\mathcal{Y}), \mathcal{Y}) = 0 \tag{19}$$

$$t(\mathcal{Y}) = \frac{d_C(\mathcal{Y}_0, \mathcal{X}) - d_C(\mathcal{Y}, \mathcal{X})}{d_C(\mathcal{Y}_0, \mathcal{X})}. \tag{20}$$

Note that at initialization $\mathcal{Y} = \mathcal{Y}_0$ and we have $t(\mathcal{Y}_0) = 0$, while at termination $\mathcal{Y} = \mathcal{X}$ and we have $t(\mathcal{X}) = 1$. Each time we update $\mathcal{Y}$, we are traversing the zero path. If we take n steps to reach the final exterior level set of radius $\delta_{\mathcal{X}}$, then we visit $n+2$ points along the zero path parameterized by the sequence $\mathcal{Y}_0, \mathcal{Y}_1, \ldots, \mathcal{Y}_n, \mathcal{X}$.

To see why this works well in practice, we need to show that solving $H(\theta(\mathcal{Y}_{k+1}), t(\mathcal{Y}_{k+1}))$ for $\theta(\mathcal{Y}_{k+1})$ given $\theta(\mathcal{Y}_k)$ is easy, and solving $H(\theta(\mathcal{Y}_0), t(\mathcal{Y}_0))$ for $\theta(\mathcal{Y}_0)$ is easy. To see the former, note that we move our guiding points $\mathcal{Y}$ only a small distance from $\mathcal{Y}_k$ to $\mathcal{Y}_{k+1}$, and that SION$(\mathcal{Y}_{k+1}; \mathcal{Y}_k)$ is small by construction. Thus, by the analysis in Secs. 3.3 and 3.4, the relative optimization problem is easy. To see the

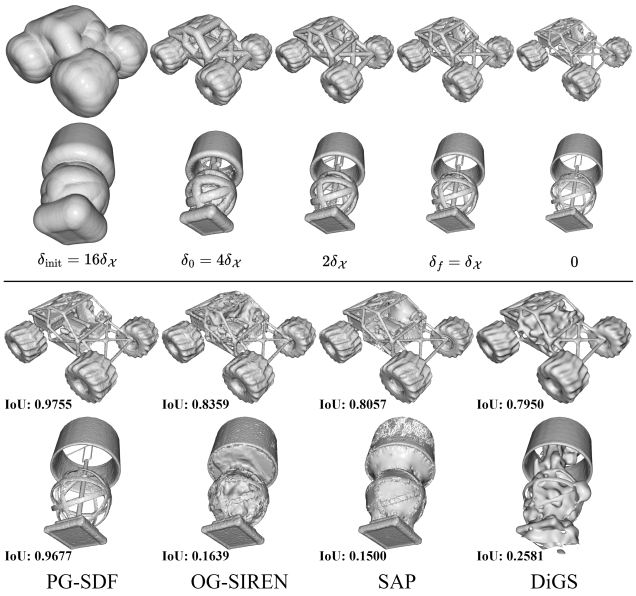

Figure 5. Qualitative results on ShapeNet. Top: Intermediate results as our method converges to different level sets Ω_δ. Bottom: Comparison with other methods, for challenging high SION shapes (0.40 for the car and 0.42 for the lamp).

latter, when $\delta_{\text{init}} \gg \delta_{\mathcal{X}}$ we can estimate good normals for $\mathcal{Y}_0$ by taking the displacement from the closest input point. With good normal estimates, $\mathcal{L}_{\widetilde{\text{sdf}}}$ is very effective, and so optimization is easy. Furthermore, we note that higher radius exterior level sets are less complex, with smaller SION to an enclosing sphere. We formalize and prove this observation in the supplement.

5. Results

Dataset. We evaluate the performance of our method on ShapeNet [11], a large dataset of 3D CAD models. In this dataset, there are many shapes with complex geometries that are challenging to reconstruct, including extremely thin surfaces (sometimes string-like), hollow regions (often with small openings), and shapes with high genus. Previous works [5, 25] have shown that many neural methods fail on these shapes, even when using the ground-truth normals. We use the preprocessed watertight subset of Williams et al. [46], which consists of 13 classes with 20 shapes in each class. For each shape, 100k points are uniformly sampled from the surface and ground-truth normals are provided for each point, which are only used for those methods that require normals. We use these points as the input to our method, and for evaluating the squared chamfer metric. An additional 100k points are uniformly sampled within the domain with ground-truth labels that indicate whether the point is inside or outside the shape, allowing for the interior intersection-over-union (IoU) metric to be calculated. Further experimental details are given in the supplement.

Method	NN	n_x	Squared Chamfer ↓ Mean	Median	Std	IoU ↑ Mean	Median	Std
SPSR [22]	×	T	5.44e-5	**1.97e-5**	5.06e-4	0.9926	**0.9956**	0.0105
IGR [15]	✓	T	5.12e-4	1.13e-4	2.15e-3	0.8102	0.8480	0.1519
SIREN [41]	✓	T	1.03e-4	5.28e-5	1.93e-4	0.8268	0.9097	0.2329
FFN [43]	✓	T	9.12e-5	8.65e-5	3.36e-5	0.8218	0.8396	0.0989
NSP [46]	×	T	5.36e-5	4.06e-5	3.64e-5	0.8973	0.9230	0.0871
DiGS+N [5]	✓	T	2.74e-4	2.32e-5	9.90e-4	0.9200	0.9774	0.1992
SDF+N†	✓	T	3.97e-5	2.22e-5	8.98e-5	0.9915	0.9951	0.0141
Ours+N	✓	T	**2.43e-5**	2.15e-5	1.66e-5	**0.9932**	0.9955	0.0076
SPSR [22]	×	E	3.76e-3	4.37e-5	1.14e-2	0.7187	0.9761	0.3767
SIREN [41]	✓	×	3.08e-4	2.58e-4	3.26e-4	0.3085	0.2952	0.2014
SAL [2]	✓	×	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 [39]	×	×	4.09e-4	2.46e-5	2.60e-3	0.9118	**0.9923**	0.2002
OG-S [25]	✓	×	3.75e-5	**2.17e-5**	8.24e-5	0.9615	0.9871	0.1048
OG-N [25]	✓	×	5.07e-5	2.57e-5	9.39e-5	0.9593	0.9870	0.1057
SDF†	✓	×	1.84e-4	8.15e-4	2.95e-3	0.5490	0.5245	0.3572
SDF+N†	✓	E	2.48e-2	5.89e-5	5.63e-3	0.6817	0.9671	0.3990
Ours+N	✓	E	2.79e-2	4.79e-5	6.79e-2	0.7017	0.9721	0.3876
Ours	✓	×	**3.72e-5**	2.40e-5	5.39e-5	**0.9729**	0.9896	0.0676

Table 1. Surface reconstruction results on ShapeNet [11]. NN: uses a neural network. n_x: uses ground-truth (T), estimated (E) or no (×) input surface normals. Suffix '+N' denotes normal information added to the underlying method. †Our implementation.

Implementation and Baselines. Our baseline methods use the same implementation framework as our approach (see full details in the supplement) with standard loss formulations. 'SDF' uses $\mathcal{L}_{\text{zls}}$, $\mathcal{L}_{\text{eik}}$, and $\mathcal{L}_{\text{pull}}$ while 'SDF+N' additionally uses $\mathcal{L}_{\widetilde{\text{sdf}}}$ and $\mathcal{L}_{\text{normal}}$ [15]. We also evaluate our methods with estimated normals (using Open3D [50]).

5.1. Surface Reconstruction on ShapeNet

Results on the full ShapeNet subset are given in Tab. 1, with qualitative results in Fig. 5. Our proposed method outperforms all prior normal-free methods and even demonstrates improvement when ground-truth normals are available. In particular, our approach gets a significant improvement in mean IoU, with much smaller variance, indicating that our method performs reliably well.

5.2. Challenging High-SION Shapes

In this section, we analyze the performance of surface reconstruction methods with respect to the SION metric. Fig. 6 shows that the mean IoU degrades significantly as we progressively consider shapes with higher SION scores. While our method also follows this trend, performance degradation is considerably slower, indicating that our method is much more robust to challenging shapes. We report the reconstruction performance on the subset of 20 shapes with the highest SION in Tab. 2. On these most challenging shapes, our approach has a decisive advantage. This demonstrates that our structured optimization approach is able to achieve good reconstructions of the most difficult shapes, without sacrificing performance on easy shapes.

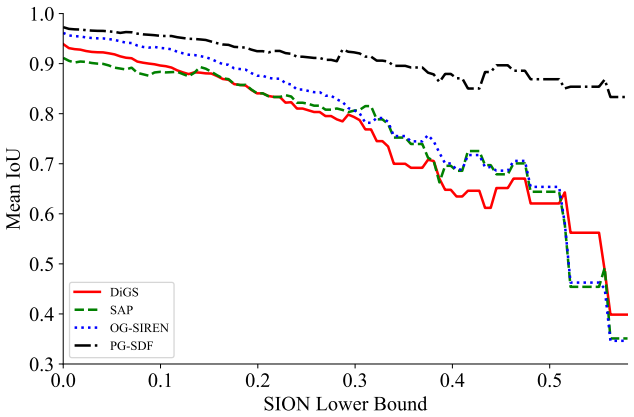

Figure 6. Mean IoU for all shapes with a SION higher than a lower bound. The analysis from Secs. 3.3 and 3.4 predicts a significant negative correlation for existing neural SDF methods. Our method weakens this trend considerably.

Method	NN	n_x	Squared Chamfer ↓ Mean	Median	Std	IoU ↑ Mean	Median	Std
SPSR [22]	×	T	**3.98e-5**	3.92e-5	1.80e-5	0.9798	0.9848	0.0222
DiGS+N [5]	✓	T	2.09e-4	7.42e-5	2.91e-4	0.8160	0.8764	0.2119
SDF+N†	✓	T	1.11e-4	4.84e-5	1.95e-4	0.9733	0.9854	0.0409
Ours+N	✓	T	4.16e-5	**3.89e-5**	1.86e-5	**0.9822**	**0.9865**	0.0148
SPSR [22]	×	E	5.83e-3	1.25e-3	1.16e-2	0.3290	0.1802	0.3445
DiGS [5]	✓	×	4.34e-4	1.71e-4	5.34e-4	0.7108	0.8196	0.2536
SAP [39]	×	×	1.28e-4	5.84e-5	1.99e-4	0.7616	0.8872	0.2665
OG-S [25]	✓	×	1.31e-4	6.50e-5	1.79e-4	0.7651	0.9219	0.2512
SDF†	✓	×	1.29e-3	7.75e -4	1.43e-3	0.3413	0.2617	0.2292
SDF+N†	✓	E	5.32e-2	4.51e-3	7.03e-2	0.2896	0.1977	0.3288
Ours+N	✓	E	4.89e-2	7.32e-3	6.25e-2	0.3173	0.2362	0.3259
Ours	✓	×	**5.51e-5**	**4.87e-5**	3.12e-5	**0.9153**	**0.9598**	0.0906

Table 2. Surface reconstruction results on the 20 highest SION ShapeNet shapes. NN: uses a neural network. n_x: uses ground-truth (T), estimated (E) or no (×) input surface normals. Suffix '+N' denotes adding normal information to the underlying method. †Our implementation.

Guidance $\mathcal{Y}_0$	$\mathcal{Y}$	δ_0	δ_f	Squared Chamfer ↓ Mean	Median	Std	IoU ↑ Mean	Median	Std
×	×	–	–	2.41e-3	1.18e-3	2.65e-3	0.2795	0.2255	0.2078
✓	×	–	–	2.63e-4	1.91e-4	2.31e-4	0.5694	0.6091	0.2278
✓	✓	4	2	9.89e-5	7.70e-5	7.95e-5	0.8657	0.9163	0.1219
✓	✓	1	1	8.08e-5	7.52e-5	5.66e-5	0.8538	0.9386	0.1602
✓	✓	4	1	**5.51e-5**	**4.87e-5**	**3.12e-5**	**0.9153**	**0.9598**	**0.0906**

Table 3. Ablation results on the 20 highest SION ShapeNet shapes. $\mathcal{Y}_0$: uses our SDF initialization strategy instead of spherical initialization [15]. $\mathcal{Y}$: uses our point guidance strategy. δ_0/δ_f: starting and final level set radii, given in multiples of $\delta_{\mathcal{X}}$.

5.3. Ablation Study and Runtime

An ablation study, on the 20 highest-SION shapes of ShapeNet, is reported in Tab. 3. The performance when using a spherical initialization and optimizing with respect to the input points (row 1) is poor. It improves significantly when replaced with our high exterior level set initialization (row 2). When point guidance is added (rows 3–5), the performance increases considerably. In particular, stopping point guidance early (when $\delta_f = 2\delta_{\mathcal{X}}$) harms performance, as does guiding straight to the final level set target ($\delta_0 = \delta_{\mathcal{X}}$) without first converging to intermediate level set targets. Our gradual homotopy approach performs best by a significant margin, justifying the strategy.

An unoptimized implementation of our method takes 10 mins on average per ShapeNet shape (a median of 7.4 mins). Convergence time depends strongly on the difficulty of the shape, as shown in Fig. 7. Our implementation has 17.5M parameters: a small neural network with Fourier feature embeddings [43] and parametric embeddings [31, 42] computed using a dense feature grid ($> 99\%$ of the parameters).

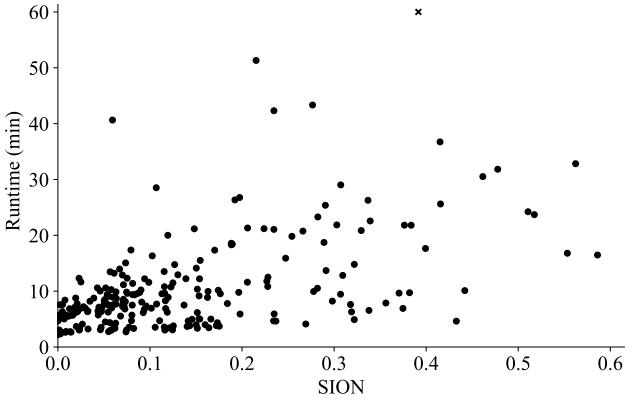

Figure 7. Runtime vs SION on ShapeNet for our method. Challenging shapes with high SION take more time to optimize. One outlier (82 min) is truncated for clarity and marked as ×.

6. Discussion and Conclusion

In this paper, we proposed a point-guided optimization method for neural SDF reconstruction from raw point clouds. We also examined why neural SDF losses can be detrimental for some shapes, proposed a metric to identify these shapes, and used this analysis to justify our algorithm design. Our method outperforms prior work, with large improvements for the most challenging shapes.

Like many methods, our approach is not robust to high outlier fractions and large noise scales, behaving similarly to the compared methods. In addition, our approach does not handle large occlusions gracefully, since it treats these regions like concavities to be explored. This could be alleviated by defining anisotropic per-point sampling radii δ_i.

Acknowledgements. Y. Ben-Shabat is supported by 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. We would also like to thank Mingda Xu for discussions about homotopy methods.

References

[1] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. In *Computational Geometry*, 2001. 2

[2] Matan Atzmon and Yaron Lipman. SAL: Sign Agnostic Learning of shapes from raw data. In *CVPR*, 2020. 2, 4, 7

[3] Matan Atzmon and Yaron Lipman. SALD: Sign Agnostic Learning with Derivatives. In *ICLR*, 2021.

[4] Ma Baorui, Han Zhizhong, Liu Yu-Shen, and Zwicker Matthias. Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces. In *ICML*, 2021. 2, 4

[5] Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. Digs: Divergence guided shape implicit neural representation for unoriented point clouds. In *CVPR*, 2022. 3, 7, 8

[6] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. In *Comput. Graph. Forum*, 2017. 2

[7] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. In *IEEE TVCG*, 1999. 2

[8] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. 10

[9] 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*, 2001. 2

[10] Frédéric Cazals and Joachim Giesen. Delaunay triangulation based surface reconstruction. In *Effective Computational Geometry for Curves and Surfaces*. Springer, 2006. 2

[11] 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:1512.03012*, 2015. 7

[12] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *NeurIPS*, 2020. 3

[13] M.P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976. 8

[14] Jean Feydy, Joan Glaunès, Benjamin Charlier, and Michael Bronstein. Fast geometric learning with symbolic matrices. In *NeurIPS*, 2020. 3

[15] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit Geometric Regularization for learning shapes. In *ICML*, 2020. 2, 4, 7, 8

[16] Benoit Guillard, Federico Stella, and Pascal Fua. Meshudf: Fast and differentiable meshing of unsigned distance field networks. In *ECCV*, 2022. 3

[17] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. In *ACM TOG*, 2020. 3

[18] Erich Hartmann. Geometry and algorithms for computer aided design, 2003. 8, 9

[19] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural kernel surface reconstruction. In *CVPR*, 2023. 2

[20] Zhangjin Huang, Yuxin Wen, Zihao Wang, Jinjuan Ren, and Kui Jia. Surface reconstruction from point clouds: A survey and a benchmark. *arXiv:2205.02413*, 2022. 1, 2

[21] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3D scenes. In *CVPR*, 2020. 2

[22] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. In *ACM TOG*, 2013. 2, 3, 7, 8

[23] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *SGP*, 2006. 2, 3

[24] Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F O'Brien. Spectral surface reconstruction from noisy point clouds. In *SGP*, 2004. 2

[25] Chamin Hewa Koneputugodage, Yizhak Ben-Shabat, and Stephen Gould. Octree guided unoriented surface reconstruction. In *CVPR*, 2023. 1, 3, 7, 8, 6, 14

[26] Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. Learning smooth neural functions via lipschitz regularization. In *SIGGRAPH*, 2022. 2

[27] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH*, 1987. 3

[28] Daniel Mayost. *Applications Of The Signed Distance Function To Surface Geometry*. PhD thesis, University of Toronto, 2014. 8

[29] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019. 2, 3

[30] Thomas Müller. tiny-cuda-nn, 2021. 6

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. In *ACM TOG*, 2022. 8, 6

[32] Yukie Nagai, Yutaka Ohtake, and Hiromasa Suzuki. Smoothing of partition of unity implicit surfaces for noise robust surface reconstruction. In *Comput. Graph. Forum*, 2009. 2

[33] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999. 6, 3

[34] Yutaka Ohtake, Alexander Belyaev, and Marc Alexa. Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. In *SGP*, 2005. 2

[35] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner,

Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. 3

[37] N.M. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer, 2002. 8

[38] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 3

[39] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape As Points: A differentiable poisson solver. In *NeurIPS*, 2021. 3, 7, 8

[40] Ralph Tyrell Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, 1970. 10

[41] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020. 2, 7

[42] 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 *CVPR*, 2021. 8, 3, 6

[43] 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. In *NeurIPS*, 2020. 7, 8, 6

[44] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *SIGGRAPH*, 2023. 6

[45] Benjamin Ummenhofer and Vladlen Koltun. Adaptive surface reconstruction with multiscale convolutional kernels. In *ICCV*, 2021. 2

[46] Francis Williams, Matthew Trager, Joan Bruna, and Denis Zorin. Neural Splines: Fitting 3D surfaces with infinitely-wide neural networks. In *CVPR*, 2021. 7

[47] Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. Geometry processing with neural fields. In *NeurIPS*, 2021. 2

[48] Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. Sdfstudio: A unified framework for surface reconstruction, 2022. 6

[49] Junsheng Zhou, Baorui Ma, Yu-Shen Liu, Yi Fang, and Zhizhong Han. Learning consistency-aware unsigned distance functions progressively from raw point clouds. In *NeurIPS*, 2022. 3

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

[51] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. In *International Conference on 3D Vision (3DV)*, 2024. 3