DOI: 10.1111/cgf.70199
Eurographics Symposium on Geometry Processing 2025
M. Attene and S. Sellán
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 5

Anisotropic Gauss Reconstruction and Global Orientation with Octree-based Acceleration

Yueji Ma [1,2], Jialu Shen[1], Yanzun Meng[1,2], Dong Xiao[3,4], Zuoqiang Shi[2,5] † and Bin Wang[3] ‡
[1] Department of Mathematical Sciences, Tsinghua University, Beijing, China
[2] Yau Mathematical Sciences Center, Tsinghua University, Beijing, China
[3] School of Software, Tsinghua University, Beijing, China
[4] School of Mathematical Sciences, University of Science and Technology of China, Hefei, China
[5] Yanqi Lake Beijing Institute of Mathematical Sciences and Applications, Beijing, China

Abstract

Unoriented surface reconstruction is an important task in computer graphics. Recently, methods based on the Gauss formula or winding number have achieved state-of-the-art performance in both orientation and surface reconstruction. The Gauss formula or winding number, derived from the fundamental solution of the Laplace equation, initially found applications in calculating potentials in electromagnetism. Inspired by the practical necessity of calculating potentials in diverse electromagnetic media, we consider the anisotropic Laplace equation to derive the anisotropic Gauss formula and apply it to surface reconstruction, called "anisotropic Gauss reconstruction". By leveraging the flexibility of anisotropic coefficients, additional constraints can be introduced to the indicator function. This results in a stable linear system, eliminating the need for any artificial regularization. In addition, the oriented normals can be refined by computing the gradient of the indicator function, ultimately producing high-quality normals and surfaces. Regarding the space/time complexity, we propose an octree-based acceleration algorithm to achieve a space complexity of $O(N)$ and a time complexity of $O(NlogN)$. Our method can reconstruct ultra-large-scale models (exceeding 5 million points) within 4 minutes on an NVIDIA RTX 4090 GPU. Extensive experiments demonstrate that our method achieves state-of-the-art performance in both orientation and reconstruction, particularly for models with thin structures, small holes, or high genus. Both CuPy-based and CUDA-accelerated implementations are made publicly available at https://github.com/mayueji/AGR.

CCS Concepts

• ***Computing methodologies*** → *Mesh models;* ***Computer graphics; Shape modeling;***

1. Introduction

Surface reconstruction from point clouds is a fundamental challenge in computer graphics, with applications spanning geographic information systems, geometric modeling, and architectural visualization. While point clouds are easier to acquire than meshes, reconstructing surfaces from unoriented points—particularly those obtained from real-world scans—remains a complex task due to the lack of consistent normal information.

In recent years, several implicit reconstruction methods for unoriented points have been proposed. VIPSS [HCJ19] minimizes Duchon's energy using L-BFGS. Iterative Poisson surface reconstruction (iPSR) proposes the idea of a normal iteration manner to eliminate the dependence on inputting normals. LMCO [GH24] presents an algorithm by solving the least-squares solution of a mostly sparse linear system according to the Stokes' theorem. Based on SPSR [KH13], SNO [HFZ*24] introduces a signed uncertainty function that distinguishes the inside and outside of the underlying surface. Learning-based methods have also been gradually introduced, including supervised learning-based reconstruction (DeepSDF [PFS*19], P2S [EGO*20]) and fitting-based implicit neural representation (IGR [GYH*20], SIREN [SMB*20], NSH [WZX*23]). However, these global methods still suffer from high running time or handling noisy data. Surface reconstruction for unoriented points is still a challenging task.

An exciting and inspiring step in this field is the exploration of the Gauss formula for surface reconstruction. Based on GR [LSSW19], the parametric Gauss reconstruction (PGR) [LXSW22] introduces the idea of constructing a practical linear system through the isotropic Gauss formula and using the conjugate gradients (CG) method to calculate the indicator field. GCNO [XDW*23] proposes a smooth nonlinear objective function to characterize the requirements of an acceptable winding-number field and converts the problem into an unconstrained optimization problem. WNNC [LSL24]

† Co-corresponding author, author's address: zqshi@tsinghua.edu.cn
‡ Co-corresponding author, author's address: wangbins@tsinghua.edu.cn

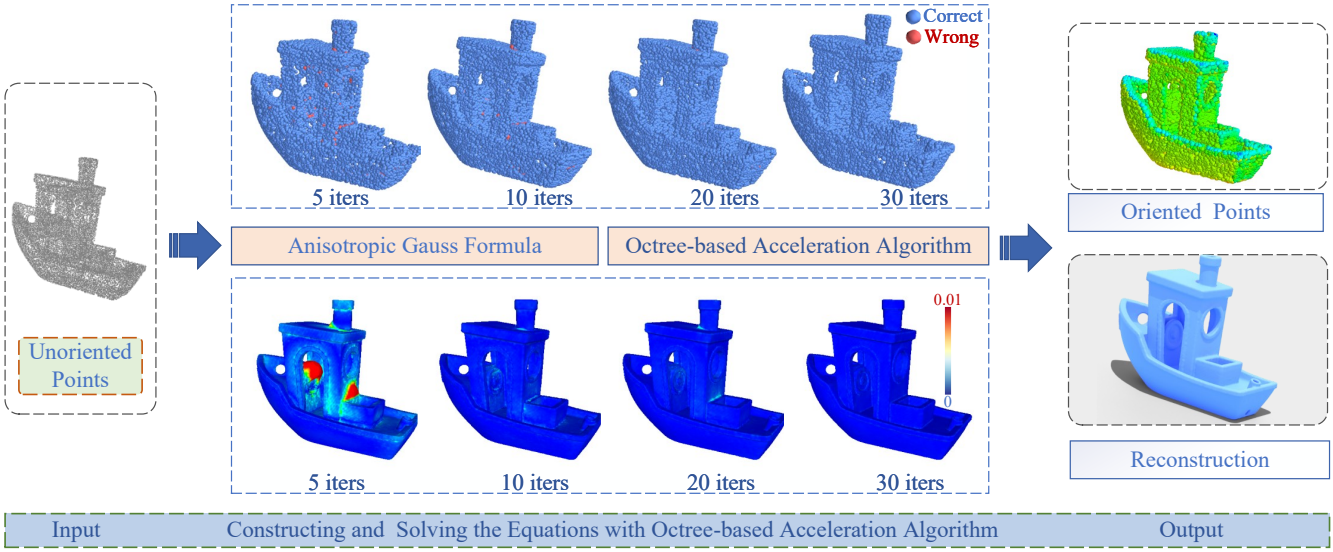


Figure 1: *Inspired by electromagnetism, we propose a novel anisotropic theory and apply it to surface reconstruction. Our method constructs more linear constraint by selecting different scaling matrices. Solving them through the preprocessed conjugate gradient, our method outputs both consistent orientation and reconstructed surface for unoriented point clouds.*

proposes the winding number normal consistency to orient points with low space and time complexity. Despite achieving high-quality results, the theoretical support for these methods is still insufficient. PGR needs to solve an under-determined linear equation system and exhibits significant sensitivity to regularization. The nonconvex nature of the objective function in GCNO often leads to convergence issues. WNNC remains an engineering solution without any theoretical proof of convergence.

To deal with these problems, we introduce a scaling matrix to the Laplace equation ($\Delta u = 0$) to make it anisotropic and apply it to surface reconstruction, called *anisotropic Gauss reconstruction* (AGR). We derive the anisotropic Gauss formula to calculate the indicator field. For N points, by selecting different scaling matrices, our method can construct $3N$ or even more linear equations with the same solution, which is treated as linearized surface elements (LSE). We adopt the preconditioned conjugate gradient (PCG) method to solve the linear system without requiring any regularization term during the solving process. Moreover, the negative gradients calculated from the indicator function should be co-directional to the LSE. Leveraging this observation, we introduce a normal updating operator to iteratively enhance the quality of normals.

Furthermore, we develop a GPU-accelerated and octree-based algorithm, which enables our method to complete the reconstruction of ultra-large-scale models with more than 5M points within 4 minutes on an NVIDIA RTX 4090 GPU. We conduct comprehensive comparisons with other state-of-the-art methods through comprehensive experiments. The results demonstrate that our method surpasses the baseline methods in the majority of cases across famous datasets. Our method also showcases robust orientation and reconstruction capabilities across diverse scenarios, encompassing nonuniform sampling, thin structures, deep holes, high genus, outliers, real-scanning, and noise. Our source codes are released on GitHub. Our main contributions can be summarized as follows.

- Method: derivation of the anisotropic Gauss formula and its application to surface reconstruction.
- Algorithm: a stable linear system formulation coupled with an octree-accelerated solver to achieve $O(N)$ space complexity and $O(N\log N)$ time complexity.
- Implementation: public release of GPU-accelerated implementations (CuPy / CUDA), enabling efficient processing of large-scale point clouds.

2. Related Work

The task of surface reconstruction has been extensively researched in the past decades. According to the dependence on normal, we classify existing methods into two categories: oriented surface reconstruction and unoriented surface reconstruction.

2.1. Surface Reconstruction for Oriented Point Clouds

As a popular choice, the radial basis function methods treat each input point as the center of a radial basis function and adjust the weights of these functions to approximate the signed distance function (SDF) of the target surface. To handle large-scale point clouds, [MYR*01] and [WCS06] propose to use compactly supported radial basis. To achieve fast evaluations, [CBC*01] applies the fast multipole method (FMM). To further improve the outputs and utilize normal information, [MGV11], [IYS*13], [LWBW16] propose to use Hermite RBFs to interpolate the positions of points and normals together.

In addition to methods based on approximation and fitting, many methods are inspired by the divergence theorem [Pfe86]. SMR

[Kaz05], WSR [MPS08], BWSR [RLH*18], and GR [LSSW19] respectively choose Fourier basis, orthogonal wavelet basis, biorthogonal wavelet basis, and isotropic fundamental solution of Laplace function, and calculate the corresponding coefficient to recover the indicator function. SMR [Kaz05] has a breakneck running speed at that time by using fast Fourier transforms (FFT) algorithm. WSR [MPS08] utilizes the orthogonal compact support property of wavelet basis functions to maintain linear space complexity and deal with multi-scale reconstruction. BWSR [RLH*18] further expands the selection from orthogonal wavelet basis to biorthogonal wavelet basis. GR [LSSW19] introduces the Gauss formula to surface reconstruction problems and achieves good results in both orientation and reconstruction.

Another popular choice is converting the reconstruction problem into a spatial Poisson problem. This novel idea is first proposed in PR [KBH06]. SSD [CT11], SPR [KH13], and EPR [KCRH20] respectively add different constraint conditions to fit the smoothed indicator function near the target surface and improve the performance of algorithm. To answer the statistical queries, SPSR [SJ22] introduces the stochastic Poisson surface reconstruction, a statistical derivation of PR as conditional probability distributions in a Gaussian process.

2.2. Surface Reconstruction for Unoriented Point Clouds

Given the inherent difficulty in acquiring consistent normals, numerous point clouds are unoriented, particularly those derived from real-world sources.

Reconstruction Based on Deep Learning With the powerful neural representation capabilities, deep learning methods have been intensively studied and widely used in surface reconstruction in recent years. Point2surf (P2S) [EGO*20] improves generalization performance and reconstruction accuracy by learning a prior over a combination of detailed local patches and coarse global information. IGR [GYH*20] proposes a new paradigm for computing high-fidelity implicit neural representations based on implicit geometric regularization. DiGS [BSKG22] incorporates second-order derivative constraints to guide the learning process of implicit neural representation, leading to better results. Neural-Singular-Hessian (NSH) [WZX*23] enforces the Hessian of the neural implicit function to have a zero determinant for points near the surface, which suppresses ghost geometry and recovers details from unoriented point clouds. Nevertheless, these deep learning methods tend to be extensive training and long running time.

Reconstruction Based on Propagation or Iteration An alternative strategy involves transforming unoriented point clouds into oriented ones and leveraging oriented surface reconstruction methods to accomplish the task. The key lies in the propagation of orientation. The original orientation propagation method traces its roots back to 1992. [HDD*92] constructs a graph for input points, whose edge costs represented orientation consistency. After that, minimum spanning tree (MST) is employed to find the solution. In recent years, many methods have been proposed to improve the robustness. Diople [MHZ*21] utilizes networks to predict consistently oriented normals for local patches, followed by dipole-based propagation to attain global consistent normals. IWSR [MMX*24] uses flipping-based iterative algorithms and proposed two novel strategies for flipping to solve the simplified problem. Iterative Poisson surface reconstruction (iPSR) [HWW*22] calculates the normals from the surface in the preceding iteration and generates a new surface with better quality. Nevertheless, these methods are highly sensitive to flipping strategies, graph weight selection, and algorithm hyper-parameters. Challenges persist in handling complex geometries effectively.

Reconstruction Based on Gauss Formula An inspiring step in this field is the exploration of the Gauss formula for unoriented surface reconstruction. Based on GR [LSSW19], parametric Gauss reconstruction (PGR) [LXSW22] constructs and solves the linear equations for linearized surface elements (LSE) to calculate the indicator field. Regarding the coefficient matrix be ill-conditioned or even singular, PGR applies proper regularization $(\alpha \cdot \mathrm{diag}(B_0))$ to improve it. The Gauss formula also offers a convenient approach to calculating the winding-number field and has attracted much attention. GCNO [XDW*23] proposes a smooth objective function to evaluate either 0 or 1 within the bounding box. WNNC [LSL24] iteratively updates normals by alternating between WNNC-based normal updates and PGR-based gradient descents to achieve the globally consistent normal. Despite achieving high-quality results, the theoretical support for these methods is still insufficient.

3. Method

Calculating the indicator function $\chi(\boldsymbol{x})$ of the target region Ω is a critical step for implicit surface reconstruction from unoriented point sets. In this section, we introduce a mathematical framework to calculate $\chi(\boldsymbol{x})$ in detail.

3.1. Anisotropic Laplace Equation

The Gauss formula (winding number) relates the indicator field to surface integrals and is widely used in surface reconstruction. The Laplace equation and its fundamental solution, with origins in both pure mathematics and electromagnetism, provide an effective framework for representing indicator functions.

In electromagnetic theory, charge distributions induce corresponding variations in the surrounding electric potential field. Electric field distributions in anisotropic materials are influenced by directional properties. The standard isotropic Laplace model is insufficient for such complexities. To address these directional effects, we propose incorporating a scaling matrix into the Laplace equation to introduce anisotropy. That is

$$\mathbf{div}(\boldsymbol{D} \cdot \nabla u) = 0, \tag{1}$$

where $\boldsymbol{D} = \mathrm{diag}(\boldsymbol{d}) = \begin{pmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{pmatrix}$, $\boldsymbol{d} = (d_1, d_2, d_3)^T \in \mathbb{R}^3$, $d_1, d_2, d_3 > 0$. To facilitate further exposition, we refer to the hyper-parameter $\boldsymbol{d}$ as the scaling vector.

This extension expands the applicability of the Laplace equation and allows for a more realistic description of potential distributions. Furthermore, by employing scale transformation and the delta function, the anisotropic fundamental solution can be derived, denoted as $\Phi_{\boldsymbol{d}}$, for Equation (1).

$$\Phi_{\boldsymbol{d}}(\boldsymbol{x}) = \frac{1}{4\pi\sqrt{d_1d_2d_3}} \frac{1}{\left(\frac{x_1^2}{d_1}+\frac{x_2^2}{d_2}+\frac{x_3^2}{d_3}\right)^{\frac{1}{2}}}, \tag{2}$$

with the gradient of it,

$$\nabla\Phi_{\boldsymbol{d}}(\boldsymbol{x}) = -\frac{1}{4\pi\sqrt{d_1d_2d_3}} \frac{\left(\frac{x_1}{d_1},\frac{x_2}{d_2},\frac{x_3}{d_3}\right)}{\left(\frac{x_1^2}{d_1}+\frac{x_2^2}{d_2}+\frac{x_3^2}{d_3}\right)^{\frac{3}{2}}}. \tag{3}$$

The complete and detailed derivation of Equations (2) and (3) is shown in the supplementary material of our paper.

For the geometric characteristics of real objects, introducing anisotropy is also crucial in surface reconstruction. Furthermore, for any point set $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^N$, the isotropic Gauss formula can only provide N non-homogeneous equations, while there are $3N$ unknown variables. As shown in [LXSW22], the constructed linear system is underdetermined and has non-unique solutions. The minimal norm solution may not align with the desired solution: linear surface element, thus lacking proper geometric meaning. Employing such solutions in subsequent reconstruction stages cannot guarantee an accurate estimation of the indicator function and the normals computed by the isotropic Gauss formula shows poor normal consistency. These limitations also make it particularly urgent to incorporate anisotropy into the surface reconstruction.

3.2. Anisotropic Gauss Formula

Firstly, the divergence theorem and double-layer potential theory are utilized to represent the indicator function as a boundary integral with an anisotropic kernel function, referred to as *anisotropic Gauss formula*.

Theorem 1 (Anisotropic Gauss Formula) Let $\Omega \subset \mathbb{R}^3$ be an open and bounded region with the smooth boundary $\partial\Omega$, then the indicator function $\chi(\boldsymbol{x})$ of the region Ω can be calculated through the anisotropic fundamental solution (2). In detail,

$$\chi(\boldsymbol{x}) = \int_{\partial\Omega} K_{\boldsymbol{d}}(\boldsymbol{x},\boldsymbol{y})\cdot\boldsymbol{n}(\boldsymbol{y})\mathrm{d}S(\boldsymbol{y}), \tag{4}$$

where

$$\begin{aligned} K_{\boldsymbol{d}}(\boldsymbol{x},\boldsymbol{y}) &= \boldsymbol{D}\cdot\nabla\Phi_{\boldsymbol{d}}(\boldsymbol{x}-\boldsymbol{y}) \\ &= -\frac{1}{4\pi\sqrt{d_1d_2d_3}} \frac{(x_1-y_1,x_2-y_2,x_3-y_3)}{\left(\frac{(x_1-y_1)^2}{d_1}+\frac{(x_2-y_2)^2}{d_2}+\frac{(x_3-y_3)^2}{d_3}\right)^{\frac{3}{2}}}, \end{aligned} \tag{5}$$

$\boldsymbol{n}(\boldsymbol{y})$ represents the outward unit normal at any point $\boldsymbol{y}\in\partial\Omega$, and $\mathrm{d}S(\boldsymbol{y})$ denotes the area element of the point $\boldsymbol{y}$.

The complete and detailed theoretical derivation of the Theorem 1 is provided in the supplementary material of our paper. When $\boldsymbol{d} = (1,1,1)^T$, the anisotropic Gauss formula degenerates to be the isotropic Gauss formula. In terms of innovation and practicality, we can construct as many equations as we want by leveraging the flexibility of anisotropic coefficients.

3.3. Constructing the Linear Equations

After deriving the theoretical formula, a detailed introduction to constructing the linear equations is provided.

First, we discretize the integral in Equation (4) by the input point cloud $\mathcal{P}\in\partial\Omega$.

$$\begin{aligned} \chi(\boldsymbol{x}) &= \int_{\partial\Omega} K_{\boldsymbol{d}}(\boldsymbol{x},\boldsymbol{y})\cdot\boldsymbol{n}(\boldsymbol{y})\mathrm{d}S(\boldsymbol{y}) \\ &\approx \sum_{j=1}^N \varphi_j^{\boldsymbol{d}}(\boldsymbol{x})\cdot\boldsymbol{n}_{\boldsymbol{p}_j}\sigma_{\boldsymbol{p}_j}, \end{aligned} \tag{6}$$

where

$$\varphi_j^{\boldsymbol{d}}(\boldsymbol{x}) = -\frac{1}{4\pi\sqrt{d_1d_2d_3}} \frac{(x_1-p_{j,1},x_2-p_{j,2},x_3-p_{j,3})}{\left(\frac{(x_1-p_{j,1})^2}{d_1}+\frac{(x_2-p_{j,2})^2}{d_2}+\frac{(x_3-p_{j,3})^2}{d_3}\right)^{\frac{3}{2}}},$$

and $\boldsymbol{n}_{\boldsymbol{p}_j}$ represents the outward unit normal at $\boldsymbol{p}_j = (p_{j,1},p_{j,2},p_{j,3})$, $\sigma_{\boldsymbol{p}_j}$ denotes the discrete area element of the point $\boldsymbol{p}_j$. The only unknowns are the normal $\boldsymbol{n}_{\boldsymbol{p}_j}$ and area elements $\sigma_{\boldsymbol{p}_j}$. Similar to PGR [LXSW22], we discretize the theorem and construct the linear systems. We denote the 3-dimensional vector

$$\boldsymbol{\mu}_j = (\mu_{j,1},\mu_{j,2},\mu_{j,3}) \triangleq \boldsymbol{n}_{\boldsymbol{p}_j}\sigma_{\boldsymbol{p}_j} \tag{7}$$

to estimate area element and normal of $\boldsymbol{p}_j$, called linearized surface element (LSE). Then, the anisotropic Gauss formula has a new form

$$\chi(\boldsymbol{x}) \approx \sum_{j=1}^N\sum_{k=1}^3 \varphi_{j,k}^{\boldsymbol{d}}(\boldsymbol{x})\mu_{j,k} \tag{8}$$

to calculate the indicator function of Ω. Although the lack of outward normals prevents a direct estimation of $\mu_{j,k}$, we can take $\mu_{j,k}$ as unknown variables.

For any fixed scaling vector $\boldsymbol{d}$, we select the input points $\boldsymbol{p}_i$ lying on the surface as the query point $\boldsymbol{x}$ in Equation (8) sequentially to construct constraints. According to Theorem 1, the indicator function is $\frac{1}{2}$ at input points: $\chi(\boldsymbol{p}_i) = \frac{1}{2}, i = 1,2,\cdots,N$. That is

$$\sum_{j=1}^N\sum_{k=1}^3 \varphi_{j,k}^{\boldsymbol{d}}(\boldsymbol{p}_i)\mu_{j,k} = \frac{1}{2}, \quad i = 1,2,3,\cdots,N.$$

For brevity, we denote it as

$$A_i^{\boldsymbol{d}}(\boldsymbol{p}_i;\mathcal{P})\boldsymbol{\mu} = \frac{1}{2}, \quad i = 1,2,3,\cdots,N, \tag{9}$$

where $\boldsymbol{\mu}\in\mathbb{R}^{3N\times 1}$ denotes the flattened vector of $\mu_{j,k}$, and $A_i^{\boldsymbol{d}}\in\mathbb{R}^{1\times 3N}$ is a row vector to represent the flattened vector of $\varphi_{j,k}^{\boldsymbol{d}}(\boldsymbol{p}_i)$. In case where $\boldsymbol{x} = \boldsymbol{y} = \boldsymbol{p}_j$, the denominator of Equation (5) becomes 0, leading to a singular equation that cannot be disregarded. The singularity could introduce numerical instability and reduce the stability of method. Hence, we modify the distance function as

$$\widetilde{d}(\boldsymbol{x},\boldsymbol{p}_j) = \max\left\{|\boldsymbol{x}-\boldsymbol{p}_j|, w(\boldsymbol{x})\right\},$$

where $w(\boldsymbol{x})$ is the width function to set the threshold of truncation. It is related to the average distance of the k closest points to the point cloud and can be calculated as

$$w(\boldsymbol{x}) = \min\left\{\max\left\{w_{\min}, \sqrt{\frac{1}{k}\sum_{\boldsymbol{p}\in\mathrm{kNN}(\boldsymbol{x};\mathcal{P})}|\boldsymbol{x}-\boldsymbol{p}|^2}\right\}, w_{\max}\right\},$$

where $\mathrm{kNN}(\boldsymbol{x};\mathcal{P})$ represents the k nearest neighbors of $\boldsymbol{x}$ in input point cloud $\mathcal{P}$, and $|\cdot|$ means the L_2 norm of vector. $w_{\min}$, $w_{\max}$, k are hyperparameters. $w_{\min}$ and $w_{\max}$ are suggested to be set as 0.002, 0.016 for clean points respectively. The default value for k is 7. We also provide its recommended value for points with different levels of noise in the code. We denote Equation (9) after modifying the distance as

$$A_{\boldsymbol{d},i}(\boldsymbol{p}_i;\mathcal{P})\boldsymbol{\mu} = \frac{1}{2}, \quad i = 1,2,3,\cdots,N. \tag{10}$$

3.4. Scaling Vector d

We formulate N equations for each scaling vector $\boldsymbol{d}$ for uniformity, and the corresponding matrix form of Equation (10) is

$$A_{\boldsymbol{d}}(\mathcal{P};\mathcal{P})\boldsymbol{\mu} = \frac{\mathbf{1}}{2}, \tag{11}$$

where $A_{\boldsymbol{d}} \in \mathbb{R}^{N\times 3N}$, $\boldsymbol{\mu} \in \mathbb{R}^{3N\times 1}$, $\frac{\mathbf{1}}{2} \in \mathbb{R}^{N\times 1}$. There is no restriction on the number of scaling vectors. For 3-dimensional point clouds $\mathcal{P}$ with N points, the number of unknown variables is $3N$. However, a single scaling vector can construct at most N non-homogeneous equations. Hence, to avoid solving underdetermined linear systems, we opt to select three or more sets of different scaling vectors. We denote m as the number of selected scaling vectors and analyze the cases of $m = 3$ and $m > 3$ separately.

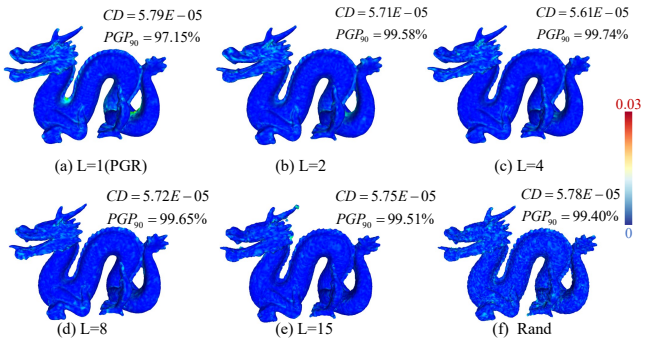


Figure 2: *The qualitative and quantitative comparison of different scaling parameters L. The colors, accompanied by a color bar, indicate the Hausdorff distance between the reconstructed surface and the ground truth.*

When $m = 3$, the coefficient matrix is a square matrix. The linear system becomes

$$A\boldsymbol{\mu} = \begin{bmatrix} A_{\boldsymbol{d}_1}(\mathcal{P};\mathcal{P}) \\ A_{\boldsymbol{d}_2}(\mathcal{P};\mathcal{P}) \\ A_{\boldsymbol{d}_3}(\mathcal{P};\mathcal{P}) \end{bmatrix} \boldsymbol{\mu} = \boldsymbol{h}, \tag{12}$$

where $A \in \mathbb{R}^{3N\times 3N}$, $\boldsymbol{\mu} \in \mathbb{R}^{3N\times 1}$, $\boldsymbol{h} = \frac{\mathbf{1}}{2} \in \mathbb{R}^{3N\times 1}$. Similar to PGR [LXSW22], it can be computed as

$$\boldsymbol{\mu} = A^T\boldsymbol{\xi}, \quad \text{with} \quad B\boldsymbol{\xi} = \boldsymbol{h}, \tag{13}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{3N\times 1}$ is an intermediate variable and $B = AA^T$.

The key improvement lies in that our method no longer needs any regularization term (especially $(\alpha \cdot \mathrm{diag}(B_0))$ in PGR [LXSW22]) during the process of solving linear systems.

Given that $\boldsymbol{d}$ represents the scaling transformation, we implement a recommended selection strategy for scaling vectors. We denote the set of values for scaling vectors as $\mathcal{D}_L$ with hyperparameters L. In detail,

$$\begin{aligned} &\mathcal{D}_L = \{\boldsymbol{d}_{L,1}, \boldsymbol{d}_{L,2}, \boldsymbol{d}_{L,3}\}, \\ &\boldsymbol{d}_{L,1} = (L,1,1)^T, \boldsymbol{d}_{L,2} = (1,L,1)^T, \boldsymbol{d}_{L,3} = (1,1,L)^T. \end{aligned} \tag{14}$$

This strategy scales along the x, y, and z axes independently. We can also choose different L to fine-tune the scaling magnitude. As shown in Figure 2, our proposed selection strategy for scaling vectors with a suitable L can significantly improve the quality of the output normals and reconstructions. Too small L can only introduce weak anisotropy, while too large L leads to larger numerical errors in calculations, making it unsuitable. In addition, the above is just one recommended selection strategy. $\boldsymbol{d}$ can be independently chosen without grouping any three scaling vectors together, and the chosen anisotropies do not need to be axis-aligned.

From the fact in mathematical algebra, it is known that a matrix with more large singular values contains a greater number of relatively independent equations. To this end, we compare the distribution of singular values in the coefficient matrix under the isotropic case with that of our proposed strategy. As illustrated in Figure 3, our strategy significantly increases the number of independent equations. For the model with 2K points, the number of valid constraints rises from approximately 1990 to around 5906.

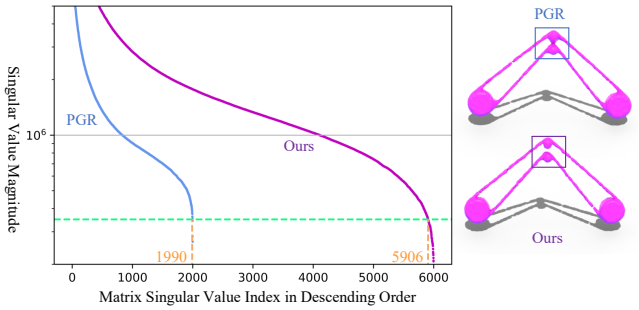


Figure 3: *The left subgraph shows the comparison of the singular values from the coefficient matrix for the isotropic case versus our strategy. Our strategy enhances the number of independent equations, increasing it from approximately 1990 to 5906. Meanwhile, the right subgraph shows the comparison of the reconstructions.*

We conduct a detailed numerical analysis for the rank of constructed matrix. We randomly sample 5, 10, 50, 100, 500, 1000, and 5000 points from Bunny and 3D spheres respectively (10 times per scale), and adopt $m = 3$ with $L = 3$ and strategy (14) to construct the linear equation systems. Python-based analysis confirmed that all systems are full rank. We show an example of constructed matrix in the supplementary material.

When $m > 3$, the equations become an overdetermined system and we seek the least squares solution of it. That is

$$\min_{\boldsymbol{\mu}} ||A\boldsymbol{\mu} - \boldsymbol{h}_m||_2^2,$$

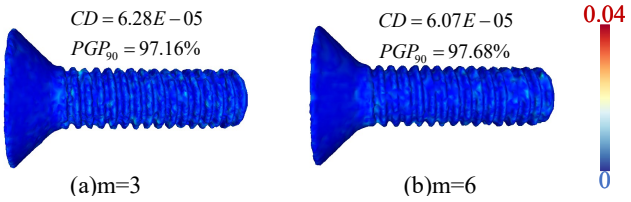


Figure 4: *The qualitative and quantitative comparison of reconstruction with different number of scaling vectors ($m = 3$ and 6). The colors, accompanied by a color bar, indicate the Hausdorff distance between the reconstructed surface and the ground truth. Increasing the number of scaling vectors leads to a slight improvement.*

where $\boldsymbol{h}_m = \frac{1}{2} \in \mathbb{R}^{mN\times 1}$, which has an equivalent form with

$$\min_{\boldsymbol{\mu}} \sum_{i=1}^{m} ||A_{\boldsymbol{d}_i}\boldsymbol{\mu} - \boldsymbol{h}_1||_2^2, \tag{15}$$

where $\boldsymbol{h}_1 = \frac{1}{2} \in \mathbb{R}^{N\times 1}$. Then, it can be computed as $H\boldsymbol{\mu} = \left(\sum_{i=1}^{m} A_{\boldsymbol{d}_i}^T \boldsymbol{h}_1\right)$, with

$$H = \begin{bmatrix} A_{\boldsymbol{d}_1}^T(\mathcal{P};\mathcal{P}) & \cdots & A_{\boldsymbol{d}_m}^T(\mathcal{P};\mathcal{P}) \end{bmatrix} \begin{bmatrix} A_{\boldsymbol{d}_1}(\mathcal{P};\mathcal{P}) \\ \vdots \\ A_{\boldsymbol{d}_m}(\mathcal{P};\mathcal{P}) \end{bmatrix} = \sum_{i=1}^{m} A_{\boldsymbol{d}_i}^T A_{\boldsymbol{d}_i}. \tag{16}$$

Although our method theoretically allows for the construction of an infinite number of equations, increasing the number of equations also raises the running time. Additionally, given that the equations only involve $3N$ unknowns, constructing too many equations may introduce linear correlations to degrade the quality of the matrix. We select $m = 6$ as a typical example of $m > 3$ by choosing $L = 3$, $L = 6$ as two sets of parameters in the experiment.

The qualitative and quantitative comparison between $m = 3$ and $m = 6$ is shown in Figure 4. The improvement by choosing $m = 6$ in reconstruction quality is slight, and it comes with a significant increase in computing time. When A is a square matrix, the algorithm outlined in Equation (13) and in Equation (16) are completely equivalent. To promote consistency and uniformity, we use the least squares method in Equation (16) to construct linear systems in experiments. Since the eigenvalues of AA^T and A^TA are identical, the previous singular value analysis of the coefficient matrix remains applicable. Consequently, we adopt $m = 3$ with $L = 3$ (the average value of the optimal parameters in Figure 2) and strategy (14) to construct $3N$ equations uniformly in all following experiments with an interface for $m > 3$ case in the provided code.

3.5. Solving the Linear System and Reconstructing the Surface

After determining the strategy for constructing equations, the following subsection explains how to obtain the oriented normals of input points and reconstruct the mesh.

Preconditioned Conjugate Gradient (PCG) Algorithm The preconditioned conjugate gradient algorithm [Kaa88] avoids matrix inversion calculations, and has been proved to converge quickly. It is suitable for solving symmetric linear equations and large-scale problems. Since H in Equation (16) is a symmetric matrix, we use the PCG to solve linear equations, which includes using the steepest descent method to estimate a high-quality initial value for the standard conjugate gradient algorithm [Naz09]. In addition, we use the matrix blocking algorithm to obtain A by calculating $A_{\boldsymbol{d}_1}, A_{\boldsymbol{d}_2}, A_{\boldsymbol{d}_3}$ separately.

Algorithm 1: Anisotropic Gauss Reconstruction

Data: Unoriented point cloud $\mathcal{P}$, the maximum depth of octree $D_{\max}$, width function parameters $w_{\min}$, $w_{\max}$, the scaling parameter L, and the steps N_1, N_2, N_3 for SD, CG and normal updating.
Result: Oriented point cloud $(\mathcal{P},\mathcal{N})$, a watertight surface M.

Set up an octree O with a maximum depth $D_{\max}$ by $\mathcal{P}$ and select the corner points of the octree as query points $\mathcal{Q}$;
Calculate the width of the input points as [LSSW19] ;
Let $\boldsymbol{\mu} = \boldsymbol{0}, \boldsymbol{b} = \frac{1}{2}, \boldsymbol{d}_1 = (L,1,1)^T, \boldsymbol{d}_2 = (1,L,1)^T, \boldsymbol{d}_3 = (1,1,L)^T$;
// Steepest Descent (SD) for an initial solution;
for $j = 1$ **to** N_1 **do**
  Calculate $\boldsymbol{p}_i = A_{\boldsymbol{d}_i}\boldsymbol{\mu}, i = 1,2,3$;
  Calculate $\boldsymbol{q} = A_{\boldsymbol{d}_1}^T\boldsymbol{p}_1 + A_{\boldsymbol{d}_2}^T\boldsymbol{p}_2 + A_{\boldsymbol{d}_3}^T\boldsymbol{p}_3$;
  Calculate $\boldsymbol{r} = A_{\boldsymbol{d}_1}^T\boldsymbol{b} + A_{\boldsymbol{d}_2}^T\boldsymbol{b} + A_{\boldsymbol{d}_3}^T\boldsymbol{b} - \boldsymbol{q}$;
  Calculate $A\boldsymbol{r} = A_{\boldsymbol{d}_1}\boldsymbol{r} + A_{\boldsymbol{d}_2}\boldsymbol{r} + A_{\boldsymbol{d}_3}\boldsymbol{r}$;
  Calculate $\beta = \dfrac{\langle \boldsymbol{r},\boldsymbol{r}\rangle}{\langle A\boldsymbol{r},A\boldsymbol{r}\rangle}$ and update $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta\cdot\boldsymbol{r}$;
end
// Conjugate Gradient (CG) for the LSE;
Initialize $\boldsymbol{p}_{CG} \leftarrow \boldsymbol{r}$ for the Conjugate Gradient (CG) algorithm;
for $j = 1$ **to** N_2 **do**
  Calculate $\boldsymbol{p}_i = A_{\boldsymbol{d}_i}\boldsymbol{p}_{CG}, i = 1,2,3$;
  Calculate $\boldsymbol{q} = A_{\boldsymbol{d}_1}^T\boldsymbol{p}_1 + A_{\boldsymbol{d}_2}^T\boldsymbol{p}_2 + A_{\boldsymbol{d}_3}^T\boldsymbol{p}_3$;
  Calculate $\beta = \dfrac{\langle \boldsymbol{r},\boldsymbol{r}\rangle}{\langle \boldsymbol{p}_1,\boldsymbol{p}_1\rangle + \langle \boldsymbol{p}_2,\boldsymbol{p}_2\rangle + \langle \boldsymbol{p}_3,\boldsymbol{p}_3\rangle}$;
  Update $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \beta\cdot\boldsymbol{p}_{CG}$;
  Calculate $\boldsymbol{r}_2 = \boldsymbol{r} - \beta\cdot\boldsymbol{q}$;
  Calculate $\gamma = \dfrac{\langle \boldsymbol{r}_2,\boldsymbol{r}_2\rangle}{\langle \boldsymbol{r},\boldsymbol{r}\rangle}$;
  Update $\boldsymbol{p}_{CG} \leftarrow \boldsymbol{r}_2 + \gamma\cdot\boldsymbol{p}_{CG}, \boldsymbol{r} \leftarrow \boldsymbol{r}_2$;
end
Calculate the surface area element s_i of $\mathcal{P}$;
// Normal updating for the LSE ($\boldsymbol{\mu} \mapsto U(\boldsymbol{\mu})$);
for $j = 1$ **to** N_3 **do**
  Update the normal $\boldsymbol{n} \leftarrow (H\Phi)\boldsymbol{\mu}$;
  Rescale normals to achieve the updated LSE $\boldsymbol{\mu}_i \leftarrow s_i \cdot \dfrac{\boldsymbol{n}_i}{||\boldsymbol{n}_i||_2}$;
end
Calculate normals $\boldsymbol{n}_i = \dfrac{\boldsymbol{\mu}_i}{||\boldsymbol{\mu}_i||_2}$, and output oriented points $(\mathcal{P},\mathcal{N})$;
Calculate the average coefficient matrix $\overline{A} = (\sum_{k=1}^{m} A_{\boldsymbol{d}_k})/m$;
Calculate the iso-value as $v_{\text{iso}} = \text{average}\left(\overline{A}(\mathcal{P};\mathcal{P})\boldsymbol{\mu}\right)$;
Obtain the values of query points as $\mathcal{Q}.\text{value} = \overline{A}(\mathcal{Q};\mathcal{P})\boldsymbol{\mu}$;
Reconstruct surface M using marching cubes by $\mathcal{Q}$.value and v_{iso} ;

Normal Update and Orientation The core of the unoriented surface reconstruction is to obtain the oriented normal of input points. We deal with the orientation task first.

We reshape $\boldsymbol{\mu}$ into the matrix with the size of $N \times 3$, where each

row represents one of the oriented normal of input points. That is

$$\boldsymbol{\mu} \in \mathbb{R}^{3N\times 1} \to \widehat{\boldsymbol{\mu}} = \begin{bmatrix} \mu_1 & \mu_2 & \mu_3 \\ \vdots & \vdots & \vdots \\ \mu_{3N-2} & \mu_{3N-1} & \mu_{3N} \end{bmatrix} \in \mathbb{R}^{N\times 3},$$

and let $(s_{\boldsymbol{p}_i}, \widehat{\boldsymbol{n}}_{\boldsymbol{p}_i})$ be the estimated area element and the estimated outward unit normal for point $\boldsymbol{p}_i \in \mathcal{P}$,

$$\begin{aligned} s_{\boldsymbol{p}_i} &= \sqrt{\mu_{3i+1}^2 + \mu_{3i+2}^2 + \mu_{3i+3}^2}, \\ \widehat{\boldsymbol{n}}_{\boldsymbol{p}_i} &= \frac{(\mu_{3i+1}, \mu_{3i+2}, \mu_{3i+3})}{s_{\boldsymbol{p}_i}}. \end{aligned} \tag{17}$$

While the linear system of our method incorporates more nonhomogeneous equations through the introduction of anisotropy, discretizing the anisotropic Gauss formula may introduce a minor numerical error, resulting in a slight deviation between the solved LSE and the ground truth. Therefore, a normal updating operator is proposed to improve the quality of output normals.

In detail, the negative gradient direction of the indicator function at the boundary of the target region is the outward normal. We can take the gradient on both sides of the anisotropic Gauss formula (4) to calculate a new normal. In other words,

$$\begin{aligned} \nabla\chi(\boldsymbol{x}) &= \nabla\left(\int_{\partial\Omega} K_{\boldsymbol{d}}(\boldsymbol{x},\boldsymbol{y})\cdot\boldsymbol{n}(\boldsymbol{y})\mathrm{d}S(\boldsymbol{y})\right) \\ &= \int_{\partial\Omega} \nabla_{\boldsymbol{x}} K_{\boldsymbol{d}}(\boldsymbol{x},\boldsymbol{y})\cdot\boldsymbol{n}(\boldsymbol{y})\mathrm{d}S(\boldsymbol{y}). \end{aligned} \tag{18}$$

We choose $\boldsymbol{d} = (1,1,1)^T$ in Equation (18), then

$$\nabla\chi(\boldsymbol{x}) = \sum_{j=1}^{N} H_{\boldsymbol{x}}\Phi(\boldsymbol{x},\boldsymbol{p}_j)\cdot\boldsymbol{\mu}_j,$$

where $H_{\boldsymbol{x}}\Phi$ is the Hessian of Φ about $\boldsymbol{x}$ and let $\overline{\Phi}(\boldsymbol{x}-\boldsymbol{p}_j) \triangleq \Phi(\boldsymbol{x},\boldsymbol{p}_j)$ for simplicity. Then, we have $H_{\boldsymbol{x}}\Phi = H\overline{\Phi}$ and

$$H\overline{\Phi}(\boldsymbol{y}) = \frac{1}{4\pi|\boldsymbol{y}|^5}\left(\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} - |\boldsymbol{y}|^2\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}\right),$$

where $|\cdot|$ means the L_2 norm of vector.

Since the calculated area elements have been already extracted in Equation (17), we can obtain the updated LSE

$$\boldsymbol{\mu}_j = s_{\boldsymbol{p}_j}\cdot\frac{\nabla\chi(\boldsymbol{p}_j)}{|\nabla\chi(\boldsymbol{p}_j)|}.$$

Namely, we construct a self-updating operator for LSE, referred to as U. Multiple such updating operators can be sequentially interconnected to improve the normal quality and keep the fast speed of our method. Figure 17 shows a qualitative comparison of using different steps in our method.

The ground truth normal is the fixed point of the operator U. Disregarding discrete errors, the theoretical solution of our method corresponds to the ground truth of the oriented normal and area elements of input points.

Figure 5 presents a qualitative comparison of normal estimation

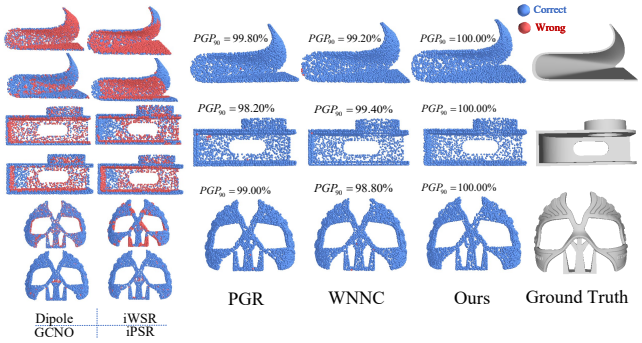


Figure 5: *Qualitative and quantitative comparison of our method and baselines on orientation. The red points indicate the wrong orientation. Our method demonstrates superior performances on orientation compared to baselines.*

with other famous methods. Our method demonstrates superior performance, showcasing the high-quality normals with minimal angular deviation from the ground truth. The results outperform other famous methods.

Iso-surface Extraction After completing the normal estimation task, we calculate the value of the indicator function on the query points by averaging the results of the matrix multiplications to reconstruct the surface. We select the set of all corner points on the octree $\mathcal{Q} = \{\boldsymbol{q}_s\}_{s=1}^{N_Q}$ as the query point set. That is

$$\mathcal{Q}.\text{value} = \frac{1}{m}\sum_{i=1}^{m} A_{\boldsymbol{d}_i}(\mathcal{Q};\mathcal{P})\boldsymbol{\mu}.$$

Note that the numerical calculation error may cause the slight shift of iso-value from $\frac{1}{2}$. We update the iso-value v_{iso} with

$$\begin{aligned} v_{\text{iso}} &= \text{average}\left(\frac{1}{m}\sum_{i=1}^{m} A_{\boldsymbol{d}_i}(\mathcal{P};\mathcal{P})\boldsymbol{\mu}\right) \\ &= \frac{1}{mN}\sum_{i=1}^{m}\sum_{j=1}^{N} A_{\boldsymbol{d}_i}(\boldsymbol{p}_j;\mathcal{P})\boldsymbol{\mu}. \end{aligned}$$

Finally, we extract surfaces with marching cubes [LC98] on octree by v_{iso} and $\mathcal{Q}$.value. A detailed description of our method is provided in Algorithm 1. In Algorithm 1, $\langle\cdot,\cdot\rangle$ means the L_2 inner product of vectors, e.g. $\langle\boldsymbol{a},\boldsymbol{b}\rangle = \boldsymbol{b}^T\boldsymbol{a}$.

Our method demonstrates the state-of-the-art performance of reconstruction, as shown in Figures 6, 8, and 10.

3.6. Octree-based Acceleration on GPU

Although the PCG algorithm helps our method avoid the matrix inversion calculation, it does not eliminate the need to store the coefficient matrix. Since the kernel function $K_{\boldsymbol{d}}(\boldsymbol{x},\boldsymbol{y})$ in Equation (5) is not compact in the three-dimensional space, our coefficient matrix A is dense. The time and space complexity of constructing a dense matrix element by element are both $O(N^2)$, limiting the applicability of method in large-scale point clouds. For example,

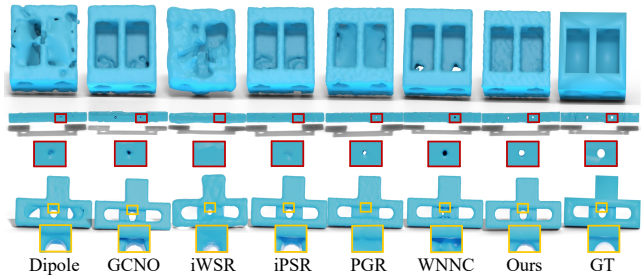


Figure 6: *Qualitative comparison of our method and baselines on reconstruction for models with 5K points. The zoom-in view is also provided for the reconstruction details. Our method outperforms the baselines.*

PGR [LXSW22] suffers from this problem seriously. It can only handle point clouds below 50K with an average running speed. To address this issue, we leverage the distinctive structure of octree to accelerate and save running memory.

Algorithm 2: Calculate $\boldsymbol{l}_o$

Data: The representative attribute $\boldsymbol{w}_i$ for every point $\boldsymbol{p}_i \in \mathcal{P}$,the built octree $\mathcal{T}$, the current tree node o.
Result: Updated $\boldsymbol{w}-$represented octree $(\boldsymbol{l}_{\mathcal{T}}, \boldsymbol{w}_{\mathcal{T}})$.

Set $\boldsymbol{l}_o \leftarrow 0$;
if o is leaf node **then**
  for each point $\boldsymbol{p}_j \in o$ **do**
    $\boldsymbol{l}_o \leftarrow \boldsymbol{l}_o + \frac{|\boldsymbol{w}_{\boldsymbol{p}_j}|}{\sum_{\boldsymbol{p}_j \in o} |\boldsymbol{w}_{\boldsymbol{p}_j}|} \boldsymbol{p}_j$;
  end
end
else
  for each child node o_c of o **do**
    $\boldsymbol{l}_o \leftarrow \boldsymbol{l}_o + \frac{|\boldsymbol{w}_{o_c}|}{\sum_{o_{c_i}} |\boldsymbol{w}_{o_{c_i}}|} \boldsymbol{l}(\mathcal{P}, o_c)$;
    // o_{c_i}: all child nodes of o;
  end
end
return $\boldsymbol{l}_o$;

The core solving process can be divided into three linear operators: (1) $\boldsymbol{w} \mapsto A\boldsymbol{w}$, (2) $\boldsymbol{w} \mapsto A^T\boldsymbol{w}$, and (3) $\boldsymbol{w} \mapsto U(\boldsymbol{w})$. The key to acceleration lies in the correspondence between these operators and the octree structure. We regard each operator as the summing up calculation of the attribute of distance between the query points $\boldsymbol{x}$ and the input points $\boldsymbol{p}_j$ with the octree $\mathcal{T}$. For any scaling vector $\boldsymbol{d}$, the kernel $K_{\boldsymbol{d}}(\boldsymbol{x} - \boldsymbol{p}_i)$ rapidly diminishes as $|\boldsymbol{x} - \boldsymbol{p}_i|$ increases. Hence, far-field interaction of neighboring points can be approximated accordingly by the new representation $(\boldsymbol{l}_{\mathcal{T}}, \boldsymbol{w}_{\mathcal{T}})$. The former represents position information, and the latter represents weight information. It is important to note that the octree subdivision extends to the deepest level or to the node containing only one input point, which includes the distance information. The division of neighboring points can correspond one-to-one with the octree node o.

In detail, such operators from kernel functions all depend on the distance between the target point $(\boldsymbol{y})$ and the source point $(\boldsymbol{x})$. Therefore, the positions of representative points need to be computed during the approximate accumulation process. Let $\boldsymbol{l}_o$ represent the weighted location of all points in node o, and $\boldsymbol{w}_o$ represent the accumulated attributes of points in node o. $(\boldsymbol{l}_o, \boldsymbol{w}_o)$ are all the attributes stored with the octree mode o. We adopt a fine-to-coarse strategy to construct the correspondence between the accumulated attributes $\boldsymbol{w}$ of neighboring points and the octree node o. For any octree node o, its position of $\boldsymbol{w}-$representative point can be calculated by

$$\boldsymbol{l}_{o,\boldsymbol{w}} = \sum_{\boldsymbol{p}_i \in o} \frac{|\boldsymbol{w}_i|}{\sum_{\boldsymbol{p}_j \in o} |\boldsymbol{w}_j|} \boldsymbol{p}_i,$$

where $\boldsymbol{p}_j \in o$ means that the input point $\boldsymbol{p}_j$ is located in the octree node o. The $\boldsymbol{w}-$representative point's accumulated attributes can be calculated by

$$\boldsymbol{w}_o = \sum_{\boldsymbol{p}_i \in o} \boldsymbol{w}_i.$$

It is important to note that the input point cloud uniquely determines its octree. The fine-to-coarse recursive algorithm described above ensures that each input point is processed at most once when constructing $(\boldsymbol{l}, \boldsymbol{w})$. In other words, the time complexity of this construction is $O(N)$. Algorithm 2 shows the detailed fine-to-coarse and recursive algorithm by taking $\boldsymbol{l}$ as the example.

Following the construction of the $(\boldsymbol{l}, \boldsymbol{w})$-represented octree, we replace the original input points with representative points. The linear operators can be calculated through a coarse-to-fine tree traversal method. For any query point $\boldsymbol{q}$, we start from the root node of octree and sum up the accumulated attributes of the nodes that do not contain $\boldsymbol{q}$ level by level until reaching the leaf node that contains $\boldsymbol{q}$.

The octree-based acceleration algorithm exhibits a time complexity of $O(N\log N)$ and a space complexity of $O(N)$. Section 4.2 provides the detailed analysis of the running time and space complexity. We have implemented the octree-based acceleration using CUDA and NVCC compilers. The provided code integrates this GPU acceleration algorithm into PyTorch, enabling efficient computation and handling ultra-large-scale point clouds (at least 10M).

4. Experiments

Experimental Setup Experiments are conducted on an NVIDIA GeForce RTX 4090 graphics card with 24GB memory and an Intel i7-10870H CPU. We use PyTorch with CUDA-based acceleration algorithms to orient and reconstruct the surface on GPU.

Evaluating Indicator The proportion of good points (PGP) is the ratio of points for which the angle between the estimated normal and the ground truth normal is smaller than a specified threshold. A higher value of PGP indicates the better accuracy of the orientation.

$$\text{PGP}_{90}(P) = |\text{correct}.P|/|P|,$$
$$\text{correct}.P = \{\boldsymbol{p}_i \in P |\ \boldsymbol{n}_{\boldsymbol{p}_{i,\text{out}}} \cdot \boldsymbol{n}_{\boldsymbol{p}_{i,\text{true}}} > 0\}.$$

The Chamfer distance (CD) penalizes both false negatives (missing parts) and false positives (excess parts) to evaluate the recon-

Table 1: Quantitative comparisons of our method with other state-of-the-art methods on the orientation and reconstruction in sparse point clouds. GCNO is only tested on small or medium-sized datasets due to its slow computation speed. The Chamfer Distance (CD) values are multiplied by 10^5. The best results are in bold. Our method outperforms the baselines in the statistical sense.

		5K,noise:0%					5K,noise:0.5%				
		Realworld	Famous	ABC	Thingi10K	Thin	Realworld	Famous	ABC	Thingi10K	Thin
PGP_{90} ↑	Dipole [MHZ*21]	0.9060	0.9192	0.9287	0.9428	0.7167	0.8454	0.8886	0.8275	0.8727	0.6654
	iWSR [MMX*24]	0.8815	0.9555	0.8797	0.9366	0.7248	0.8795	0.9013	0.8683	0.9112	0.6755
	GCNO [XDW*23]	0.9537	0.9454	-	-	-	0.8864	0.9001	-	-	-
	iPSR [HWW*22]	0.9747	0.9772	0.9128	0.9685	0.9268	0.9411	0.9097	0.8587	0.9509	0.8950
	PGR [LXSW22]	0.9894	0.9776	0.9592	0.9759	0.9735	0.9412	0.9346	0.9180	0.9487	0.8933
	WNNC [LSL24]	0.9976	0.9827	0.9571	0.9803	0.9717	0.9757	0.9556	0.9364	0.9713	0.8975
	Ours	**0.9987**	**0.9833**	**0.9621**	**0.9805**	**0.9742**	**0.9774**	**0.9558**	**0.9397**	**0.9717**	**0.9132**
NC_p ↑	Dipole [MHZ*21]	0.7850	0.7942	0.7622	0.8441	0.4266	0.7468	0.7670	0.7103	0.8024	0.3254
	iWSR [MMX*24]	0.8143	0.8336	0.7390	0.8792	0.4553	0.8234	0.8194	0.7156	0.7965	0.3454
	GCNO [XDW*23]	0.7842	0.8196	-	-	-	0.7201	0.6875	-	-	-
	iPSR [HWW*22]	0.8488	0.8447	0.7614	0.8646	0.7262	0.7566	0.6908	0.6427	0.8015	0.7129
	PGR [LXSW22]	0.8893	0.8440	0.8370	0.8986	0.8867	0.7343	0.7203	0.7109	0.7644	0.7184
	WNNC [LSL24]	0.9486	**0.9019**	0.8677	0.9316	0.8984	0.8583	0.8005	0.7918	0.8627	0.7065
	Ours	**0.9494**	0.9003	**0.8749**	**0.9330**	**0.9011**	**0.8597**	**0.8155**	**0.7925**	**0.8747**	**0.7282**
NC_s ↑	Dipole [MHZ*21]	0.8421	0.8242	0.7832	0.8594	0.4572	0.7468	0.7670	0.7103	0.8024	0.4233
	iWSR [MMX*24]	0.8347	0.8668	0.7564	0.8792	0.5295	0.7855	0.8447	0.7454	0.8492	0.4828
	GCNO [XDW*23]	0.8244	0.8447	-	-	-	0.7795	0.8335	-	-	-
	iPSR [HWW*22]	0.8852	0.8970	0.7883	0.8983	0.7810	0.8587	0.8674	0.7586	0.8765	0.7349
	PGR [LXSW22]	0.9038	0.8779	0.8319	0.9122	0.8583	0.9011	0.8773	0.7964	0.8766	**0.7794**
	WNNC [LSL24]	0.9210	**0.9085**	0.8656	0.9249	0.8601	0.8922	0.8830	0.8343	0.8953	0.6949
	Ours	**0.9235**	0.9059	**0.8751**	**0.9279**	**0.8624**	**0.9257**	**0.9086**	**0.8429**	**0.8959**	0.7699
CD ↓	Dipole [MHZ*21]	89.40	50.42	85.74	69.53	1240.73	179.47	70.42	135.78	96.34	1936.17
	iWSR [MMX*24]	73.41	25.82	66.76	41.52	1075.28	148.88	32.85	123.76	74.07	1805.58
	GCNO [XDW*23]	41.38	36.83	-	-	-	61.38	39.68	-	-	-
	iPSR [HWW*22]	30.43	9.46	40.40	18.24	53.14	32.45	13.01	77.01	19.63	62.77
	PGR [LXSW22]	28.02	21.77	45.18	18.10	28.23	29.97	23.75	46.28	17.93	**39.53**
	WNNC [LSL24]	25.11	8.12	15.42	11.64	26.88	26.32	9.62	17.14	13.12	70.78
	Ours	**24.63**	**8.06**	**14.19**	**10.71**	**24.58**	**25.69**	**9.27**	**14.18**	**11.24**	46.36

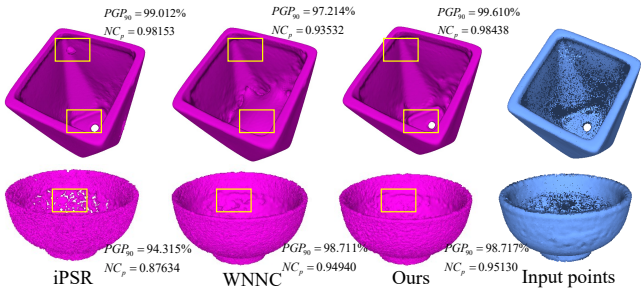


Figure 7: *Qualitative results of the reconstruction and Quantitative results of orientation on real-scan models with 200K points. Our method outputs high-quality surfaces with accurate normal.*

struction error.

$$\mathrm{CD}(S_1,S_2)=\frac{1}{|S_1|}\sum_{\boldsymbol{x}\in S_1}\min_{\boldsymbol{y}\in S_2}||\boldsymbol{x}-\boldsymbol{y}||_2^2+\frac{1}{|S_2|}\sum_{\boldsymbol{y}\in S_2}\min_{\boldsymbol{x}\in S_1}||\boldsymbol{x}-\boldsymbol{y}||_2^2.$$

S_1 and S_2 denote the reconstructed and ground truth surfaces, respectively. $|\cdot|$ denotes the cardinality of a set, representing the number of elements it contains. To evaluate the quality of the reconstruction, we adopt 20K sample points for either surface.

Normal consistency (expressed as a percentage and abbreviated as 'NC') reflects the normal consistency between two point clouds. The NC value is computed as follows:

$$\mathrm{NC}(P_1,P_2)=\frac{1}{2|P_1|}\sum_{\boldsymbol{p}_1\in P_1}\boldsymbol{n}_{\boldsymbol{p}_1}\boldsymbol{n}_{\mathrm{cl}(\boldsymbol{p}_1,P_2)}+\frac{1}{2|P_2|}\sum_{\boldsymbol{p}_2\in P_2}\boldsymbol{n}_{\boldsymbol{p}_2}\boldsymbol{n}_{\mathrm{cl}(\boldsymbol{p}_2,P_1)}$$
$$\mathrm{cl}(\boldsymbol{p},P)=\arg\min_{\boldsymbol{p}'\in P}d(\boldsymbol{p},\boldsymbol{p}'). \quad (19)$$

NC_p indicates that P_1 is chosen as the input points with its ground truth normal, and P_2 represents the output oriented points of algorithms. Serving as a supplement to PGP_{90}, NC_p is used to evaluate the quality of orientation and normal estimation. NC_s denotes that P_1 is sampled from the ground truth surface, while P_2 is sampled from the output surface of algorithms. Serving as an addition to CD, NC_s is used to evaluate the quality of the reconstructed surface.

Parameters We adopt the parameter setting $L=3.0$, $w_{\min}=0.002$, $w_{\max}=0.016$ for clean point clouds, and $w_{\min}=0.04$, $w_{\max}=0.12$ for point clouds with 0.5% Gaussian noise. We run 40 iterations for the PCG algorithm to solve the linear system with 4 steps for normal updating.

Baselines We include several famous methods (Dipole [MHZ*21], iWSR [MMX*24], PGR [LXSW22], GCNO [XDW*23], iPSR

[HWW*22], WNNC [LSL24]) for comparison. For Dipole and GCNO, we follow the default setting and match it with SPR [KH13] for reconstruction. For iWSR, despite selecting numerous parameters, its efficiency significantly decreases when dealing with sparse point cloud datasets. Due to the slow running speed of GCNO, we cannot test its performance on large datasets. For iPSR, PGR, and WNNC, we have tried our best to find their optimal parameters and show the optimal results in this paper.

4.1. Experimental Effect

Famous, ABC and Thingi10K Datasets The Famous [EGO*20] dataset includes a variety of classic shapes such as bunny, dragon, and armadillo. The ABC [KB14] dataset comprises a diverse collection of CAD meshes, while the Thingi10K [ZML*22] dataset contains a variety of shapes with complex geometric details. We randomly sample 5K points from each model to test our method capabilities on sparse point clouds. The quantitative comparison of orientation and surface reconstruction on these datasets is shown in Table 1. We also evaluate our method with dense point clouds to demonstrate its ability to reconstruct complex structural details accurately. Table 3 shows the results with dense point clouds. The qualitative comparison of surface reconstruction on these datasets is shown in Figures 6 and 8. In addition, the qualitative comparison of normal estimation and orientation on these datasets is provided in Figure 5. Experimental results demonstrate the good performance of our method on these datasets.

Real-world and Real-scan Datasets The models in Real-world [EGO*20] dataset contains noise and outliers. In addition, the ground truth mesh of such models exhibits lower smoothness. To verify the robustness of our method, we utilize the Real-world dataset to generate point clouds with 5K or 50K points. The quantitative comparison results of the methods are shown in Tables 1 and 3. Our method outperforms the baselines in most cases.

Table 2: Quantitative comparison results of different methods on Real-scan dataset [EGO*20]. The best results are in bold. Our method demonstrates good performance.

	PGP_{90} ↑			NC_p ↑		
	iPSR	WNNC	Ours	iPSR	WNNC	Ours
flower_pot	0.9974	**1.0000**	**1.0000**	0.9739	0.9899	**0.9900**
flower_pot2	0.9901	0.9721	**0.9961**	0.9815	0.9353	**0.9840**
toy_duck	0.9867	0.9999	**1.0000**	0.9763	0.9891	**0.9896**
bowl	0.9432	0.9871	**0.9872**	0.8763	0.9494	**0.9513**
lock	0.9845	0.9934	**0.9958**	0.9687	0.9700	**0.9736**
cup	0.9878	**1.0000**	**1.0000**	0.9731	0.9864	**0.9898**
Whole Dataset	0.9732	0.9907	**0.9929**	0.9201	0.9477	**0.9538**

Due to the limited number of models in the Real-world dataset, we further validate our method using the Real-scan dataset [HWW*24]. This dataset contains 20 real-scanned models with 200K points, including commodities, instruments, and art wares. The materials include metal, plastic, ceramic, and cloth, with various sensing drawbacks. The visualization comparison of our method and qualitative is shown in Figure 7. The quantitative comparison is shown in Table 2. Numerous experiments showcase the state-of-the-art performance of our method in handling real-world models.

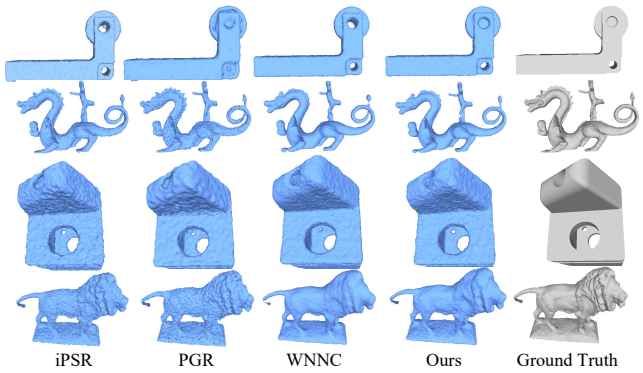


Figure 8: *Qualitative results of the reconstruction of noisy point clouds. Our method outputs the smoother mesh than iPSR. In comparison to PGR and WNNC, our method performs better in preserving holes and thin structures.*

Noisy Datasets Evaluating performance on noisy datasets is crucial for demonstrating the robustness of methods. However, most non-learning methods cannot handle noisy point clouds well, especially in sparse point clouds. We introduce a uniform Gaussian noise to the Famous [EGO*20], ABC [KB14], Thingi10K [ZML*22], and Real-world [EGO*20] datasets with 5K, 50K or 500K points as input. The standard deviation of Gaussian noise is 0.5% the length of the bounding box. The provided code has integrated recommended parameters (provided in Section 4.3) for models with various noise levels ($L_0 - L_5$, varying noise from 0% to 1%). The quantitative comparison of our method and other well-known methods is shown in Tables 1 and 3. Figure 8 displays the reconstructions from noisy point clouds. Our method outputs globally consistent orientations and achieves state-of-the-art performance in denoising and preserving detail.

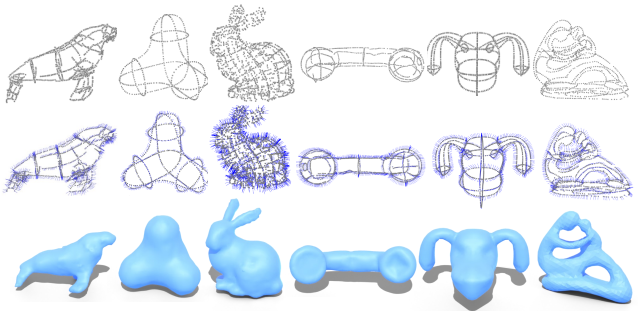


Figure 9: *The reconstruction and orientation results of our method for wireframe models. All models are sourced from VIPSS [HCJ19]. The first row shows the input unoriented point clouds (1K). The second and third rows showcase the output oriented point clouds, along with the reconstructed surfaces.*

Sparse Points and Wireframe Sparse points can be another challenge due to the lack of detailed information. Even slight deviations in normals can result in severe reconstruction errors. Figure 3 illustrates the reconstructions of our method compared to PGR on a sparse model with 2K points. Our method demonstrates better

Table 3: Quantitative comparisons of our method with other state-of-the-art methods on the orientation and reconstruction in the large-scale models. PGR can only handle about 50K point clouds on an NVIDIA GeForce RTX 4090. IPSR takes about 100 minutes to complete the reconstruction on a 500K point cloud with 0.5% Gaussian noise. The Chamfer Distance (CD) values are multiplied by 10^5. The best results are in bold. In most cases, our method gives the best results.

		50K,noise:0%					50K,noise:0.5%					500K,noise:0.5%	
		Realworld	Famous	ABC	Thingi10K	Thin	Realworld	Famous	ABC	Thingi10K	Thin	Realworld	Thin
PGP_{90} ↑	iPSR [HWW*22]	0.9984	0.9922	0.9639	0.9872	0.9817	0.9589	0.9109	0.8822	0.9639	0.8950	0.9478	-
	PGR [LXSW22]	0.9903	0.9778	0.9703	0.9859	0.9915	0.9113	0.9047	0.8949	0.9062	0.8945	-	-
	WNNC [LSL24]	**0.9999**	0.9961	**0.9872**	0.9939	0.9933	0.9801	0.9584	0.9232	0.9708	0.8986	0.9765	0.8898
	Ours	**0.9999**	**0.9962**	0.9844	**0.9940**	**0.9934**	**0.9807**	**0.9590**	**0.9306**	**0.9713**	**0.9225**	**0.9767**	**0.9239**
NC_p ↑	iPSR [HWW*22]	0.9586	0.9550	0.8914	0.9484	0.9238	0.7926	0.7035	0.7034	0.8892	0.7311	0.8373	-
	PGR [LXSW22]	0.9015	0.8664	0.8763	0.9169	0.9321	0.7271	0.7084	0.6892	0.7465	0.6724	-	-
	WNNC [LSL24]	**0.9841**	0.9712	**0.9715**	0.9791	0.9764	0.8715	0.8298	0.7928	0.8885	0.7157	0.8702	0.7027
	Ours	**0.9841**	**0.9720**	0.9584	**0.9796**	**0.9765**	**0.8851**	**0.8301**	**0.8031**	**0.8936**	**0.7598**	**0.8704**	**0.7652**
NC_s ↑	iPSR [HWW*22]	0.9406	0.9488	0.8881	0.9419	0.9120	0.8985	0.8800	0.8671	0.9171	0.8370	0.8436	-
	PGR [LXSW22]	0.9413	0.9339	0.9015	0.9505	0.9390	0.8311	0.8544	0.8432	0.9055	0.8585	-	-
	WNNC [LSL24]	0.9539	0.9511	**0.9328**	0.9539	0.9401	0.8958	**0.8906**	0.8691	0.9176	0.7686	0.9003	0.7629
	Ours	**0.9574**	**0.9523**	0.9103	**0.9548**	**0.9402**	**0.8999**	0.8867	**0.8723**	**0.9219**	**0.8699**	**0.9005**	**0.8313**
CD ↓	iPSR [HWW*22]	25.12	16.57	13.67	15.95	28.44	24.55	14.71	27.56	57.33	51.78	33.37	-
	PGR [LXSW22]	25.08	7.34	14.64	10.46	5.11	31.67	13.22	21.88	11.61	28.36	-	-
	WNNC [LSL24]	23.44	7.41	**12.42**	9.93	7.50	27.38	8.93	58.54	11.57	42.81	28.33	367.33
	Ours	**23.40**	**7.17**	13.21	**9.70**	**4.94**	**23.62**	**8.54**	**21.48**	**11.32**	**26.95**	**28.24**	**72.12**

performance in normal estimation and reconstruction, particularly in preserving thin structures. Additional qualitative comparisons of sparse point cloud reconstructions in the ABC, Real-world, and Thingi10K datasets can be found in the supplementary material of our paper. Furthermore, we evaluate our method in wireframe models with 1K points. The results are shown in Figure 9. Our method outputs high-quality normals and reconstruction for all point clouds.

Large Scale Point Clouds Handling large-scale point clouds plays a vital role in surface reconstruction. However, the current well-known methods are troubled by the scale of input point clouds. GCNO [XDW*23] needs to solve nonlinear problem optimization. PGR [LXSW22] requires storing and solving dense linear equation systems. IPSR [HWW*22] needs iterative application of SPR. When dealing with large-scale point clouds, these constraints often result in insufficient memory or long running times (more than 10 hours).

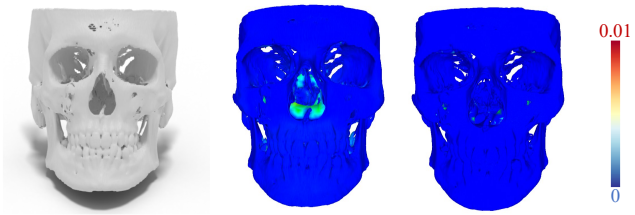


Figure 10: *Qualitative comparisons of reconstruction for models with 10M points. The colors, accompanied by a color bar, indicate the Hausdorff distance between the reconstructed surface and the ground truth. Our method has a smaller reconstruction error for the target skull.*

Based on octree-based acceleration, our method reduces the space complexity from $O(N^2)$ to $O(N)$. Notably, our method demonstrates the ability to handle ultra-large-scale models with 10M points in Figure 10 and models with 5M points in the supplementary material.

The ground truth of Figure 10 is a skull featuring numerous thin, hollow, and discontinuous structures. The figure shows a detailed comparison of our method against WNNC [LSL24] on large-scale and complex models. Our method is superior in capturing details.

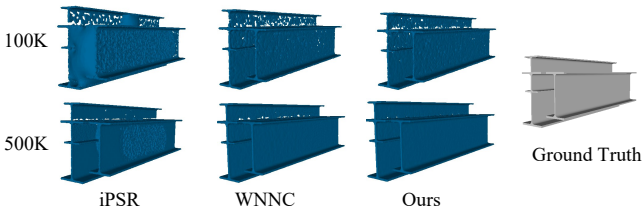


Figure 11: *Qualitative comparison of our method with iPSR and WNNC for thin structure point clouds (100K ∼ 500K). Our method reconstruct the good and complete surface.*

In addition, Figure 11 shows the reconstructed meshes with the 100K ∼ 500K input points. Table 3 shows the quantitative results on datasets that contain models with 500K points. Our method achieves state-of-the-art performance in these experiments.

Models with Holes Models with holes present higher complexity and pose significant challenges, especially for narrow and deep ones. Existing methods still cannot avoid the wrong orientation and sealing holes incorrectly, especially for small and deep holes.

Figures 6, 7, and 8 show the qualitative comparisons of reconstruction for clean, real-scan, and noisy inputs with holes. Additional experiments can be found in the supplementary material. With the introduction of anisotropy, our method successfully preserves these precious details and outputs high-quality reconstruction.

Thin Structures Thin structures typically consist of closely positioned surfaces with opposing normals, which is common in real-scan data and CAD models. Existing methods may erroneously interpret this as a single surface normal, leading to misalignments

among surfaces, resulting in low-quality orientation and reconstruction. In contrast, our method introduces anisotropy to construct more equations, enabling accurate normal estimation even for extremely thin structures. Figure 11 illustrates the reconstruction of our method and WNNC for I-beams. Additional experiments for I-beams can be found in the supplementary material. Other state-of-the-art methods struggle to handle large-scale point clouds with thin structures. Our method overcomes the constraints of existing methods that are susceptible to surface damage and improves the quality of reconstructions involving thin structures.

Furthermore, we select a subset of 70 models with thin structures from the comprehensive datasets of ABC [KB14] and Thingi10K [ZML*22] to compose the Thin dataset. Subsequently, we conduct quantitative experiments on our method and other state-of-the-art methods by sampling 5K and 50K point clouds from the Thin dataset, respectively. As shown in Tables 1 and 3, our method performs well in orientation and reconstruction tasks, especially for noisy and dense point clouds with thin structures.

High-genus Model Given their intricate topology, high-genus models pose substantial challenges in terms of accurate orientation and reconstruction. To tackle this issue, we specifically choose all 98 models with a genus greater than 50 from the official Thingi10K dataset [ZML*22] as inputs. As illustrated in Figure 12, our method accurately captures the topological features, demonstrating superior orientation and reconstruction capabilities for models with curved and fine skeletal elements.

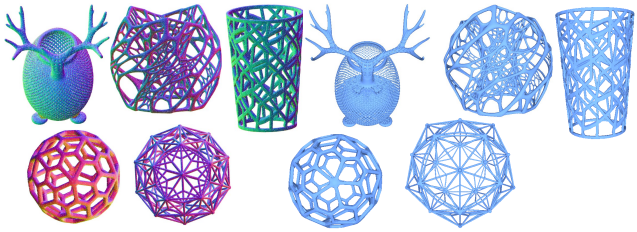
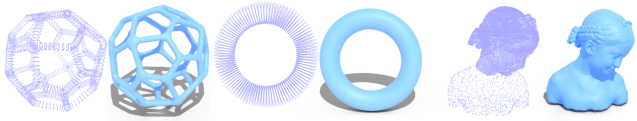


Figure 12: *Qualitative results of output normals (rendered with RGB to show the orientation) and high-quality reconstruction (in blue) of our method for high genus models.*

Parameterized Sampling Model and Nonuniform Sampling Model Compared to random sampling, the point cloud from parameterized sampling exhibits apparent regularity rather than randomly selecting on surfaces and imposes novel demands on orientation. Figure 13(a) shows the reconstruction of our method on the parameterized sampling models. Furthermore, we conduct experiments on the non-uniform sampling models to evaluate the capabilities of our method for models with variations in point cloud densities. As illustrated in Figure 13(b), the experiments demonstrate that our method can handle nonuniform sampling point clouds well and output high-quality reconstructions.

4.2. Space and Time complexity Analysis

The space and time complexity of algorithm are related to the processing potential of large-scale models and practical applications.



(a) Parameterized sampling (b) Nonuniform sampling

Figure 13: *Reconstruction results of (a) Parameterized sampling models, and (b) Nonuniform sampling models. All models are sourced from VIPSS [HCJ19]. Our method can handle both cases and output high-quality reconstructions.*

Our method makes full use of the octree structure, thus achieving a one-to-one correspondence between stored information and octree nodes. For the input model with N points, the space complexity of our method is $O(N)$ and time complexity of our method is $O(N\log N)$, both of which are state-of-the-art. Then, we analyze the complexity of our method in octree construction, solving linear equations, normal updating, and surface reconstruction in detail.

Firstly, we set $D_{\max}$ as the deepest level of the octree. Then, each input point belongs to at most $D_{\max}$ octree nodes, so the number of octree nodes is $O(N)$. With the input points normalized to the bounding box $[0,1]^3$, the establishment and pruning of an octree exhibit a time complexity of $O(N)$ and a space complexity of $O(N)$.

After adopting acceleration strategy, only three linear operators: $\boldsymbol{w} \mapsto A\boldsymbol{w}$, $\boldsymbol{w} \mapsto A^T\boldsymbol{w}$, $\boldsymbol{w} \mapsto U(\boldsymbol{w})$ need to be calculated during the PCG and normal updating process. These operators can be calculated through the $\boldsymbol{w}$-representation of all octree nodes in Algorithm 2. The $\boldsymbol{w}$-representative accumulated attributes and weighted positions of points are still one-to-one correspondence with octree nodes, so the space complexity maintains $O(N)$. As demonstrated in the Fast Multipole Method (FMM) [BH86], the time complexity for computing each linear operator for N points mentioned above is $O(N\log N)$.

In addition, only a $O(1)$ steps steepest descent for the high-quality initial value with a $O(1)$ steps conjugate gradient descent is used to solve the linear equation. A $O(1)$ steps normal updating is used to improve the quality of orientation further. As a result, the space and time complexity of our method for orientation is $O(N)$ and $O(N\log N)$, respectively.

Finally, we select the vertices of the octree leaf nodes as query points for reconstruction, resulting in a query point scale of $O(N)$. In addition, we also have completed the octree-based acceleration for the anisotropic oriented Gauss reconstruction. Hence, the complexity of space and time remains consistent with the previous analysis.

We highlight the advantages of our running speed and memory usage over the other methods based on the Gauss formula: GCNO and PGR. GCNO requires repeated evaluations of winding numbers. The balance term $f_B(n)$ needs to calculate the winding-number score for the vertex of Voronoi cells. It causes at least $O(N^2)$ space and time complexity. Hence, the running speed of GCNO is notably slow, requiring several hours to complete the orientation for a model with 20K points . PGR requires storing dense matrices and calculating dense matrix multiplication, so its time and space complexity is at least $O(N^2)$. In experiments, PGR is severely limited by memory requirements, with a maximum processing capacity of about 50K

points. In contrast, our method can orient and reconstruct ultra-large-scale models with more than 10M points in minutes. In previous

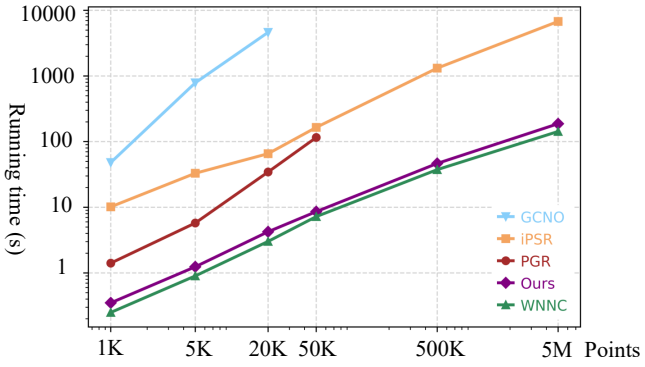


Figure 14: *Qualitative comparison of the running time of our method and the baselines. GCNO takes hours to converge on a 20K point cloud. PGR can only handle the models with at most 50K points within 24GB memory.*

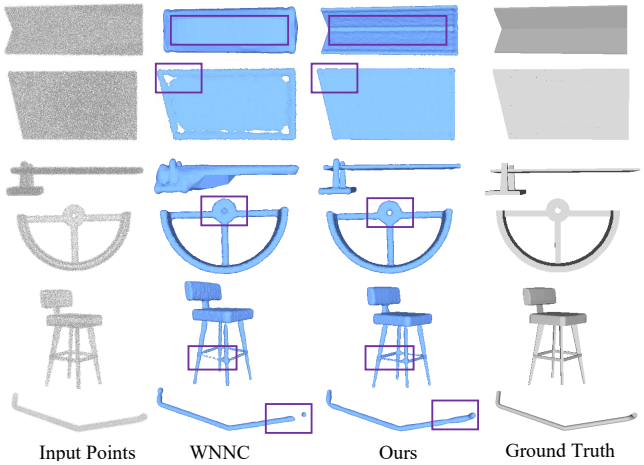


Figure 15: *Qualitative comparisons of reconstruction on large-scale noisy point clouds with thin structures. Our method outputs better results.*

experiments, we divided the models into sparse point clouds (5K), dense point clouds (50K), large-scale point clouds (100K $\sim$ 500K), and ultra-large-scale point clouds (1M $\sim$ 10M). Most existing methods can only handle sparse or dense point cloud data. Figure 14 compares different reconstruction methods in running time. By using octree-based acceleration, our method can output better orientation and reconstruction than baselines with a comparable running speed to WNNC. Due to the iteration of SPR, the computational cost of iPSR increases significantly as the number of point clouds increases. Additionally, due to its resampling of input points, especially for noisy and large-scale models, the running speed of iPSR for large-scale point clouds is even worse than the value shown in Figure 14 and supplementary materials.

AGR vs. PGR Based on the isotropic Gauss formula, PGR is a currently recognized state-of-the-art method. However, it still has much room for improvement.

Firstly, PGR has to solve an underdetermined linear system with non-unique solutions. To obtain a solution, PGR selects the minimum norm solution. However, it may not align with the desired LSE and lack geometric meaning. There is no theoretical support for the solution to accurately calculate the values at query points and recover the indicator function.

Secondly, as a chain reaction of lack of equations, the coefficient matrix of PGR has strong singularity and needs artificial regularization terms to solve it. PGR is highly sensitive to the regularization terms , which reduces the robustness of the overall algorithm.

Finally, PGR requires storing dense matrices and calculating dense matrix multiplication. It is severely limited by memory requirements, with a maximum capacity of about 50K points.

By incorporating anisotropy and integrating it with an acceleration algorithm, our method (AGR) effectively addresses the issues encountered in PGR. By leveraging the flexibility of anisotropic coefficients, we can choose three or more distinct scale vectors $\boldsymbol{d}_i$ to construct the constraint. The linear system in AGR is no longer underdetermined. Moreover, due to the augmented number of equations, AGR does not necessitate any regularization terms when solving linear systems. Based on the octree-based acceleration algorithm, AGR has good performance on time and space complexity with the processing capacity of models with more than 10M points.

The qualitative and quantitative comparison of AGR and PGR is shown in Figures 5, 6, 8, and Tables 1, 3. Additional qualitative comparisons are shown in the supplementary material. Moreover, compatibility with PGR is ensured by providing CuPy-based code.

AGR vs. WNNC WNNC iteratively update normals by alternating between WNNC-based normal updates and PGR-based gradient descents to calculate a globally consistent normal vector field.

To demonstrate the improvement of our method in reconstruction, additional experiments are conducted on handling large-scale noisy point clouds with thin structures, as shown in Figure 15. The input of first three lines is the 500K point cloud with 0.5% Gaussian noise, and the input of last three lines is the 50K point cloud with 0.5% Gaussian noise. The difficulty of solving for the large scale and complex structure significantly increases. WNNC only relies on the steepest descent method, which leads to a decrease in the convergence speed in the later stage, thereby reducing the quality of the final reconstructions.

In addition, WNNC remains solving the underdetermined system. Our method has more linearly independent constraints and the ground truth is a fixed point of both preprocessed conjugate gradient method and self-updating operator in our method.

4.3. Evaluation of Individual Parameters

Smoothing Widths It is crucial to implement smoothing adjustments to address unavoidable singularity in the solving process. Setting the proper smoothing widths to the distance function can improve the robustness, especially for noisy inputs.

A substantial smoothing width results in a uniform orientation and

smooth surface, but it may sacrifice structural details. Conversely, a narrower smoothing width preserves more details but could introduce roughness in the reconstruction process. The first row and the second row in Figure 16 compare different smoothing widths for clean and noisy inputs, respectively. This indicates that an excessively large width can result in over-smoothed reconstruction. Selecting $w_{\min}$ and $w_{\max}$ with significant differences may lead to the wrong calculations of the area element and the error of detail reconstruction. Small smoothing widths lead to more accurate calculating LSE in dense points and surfaces, preserving more details.

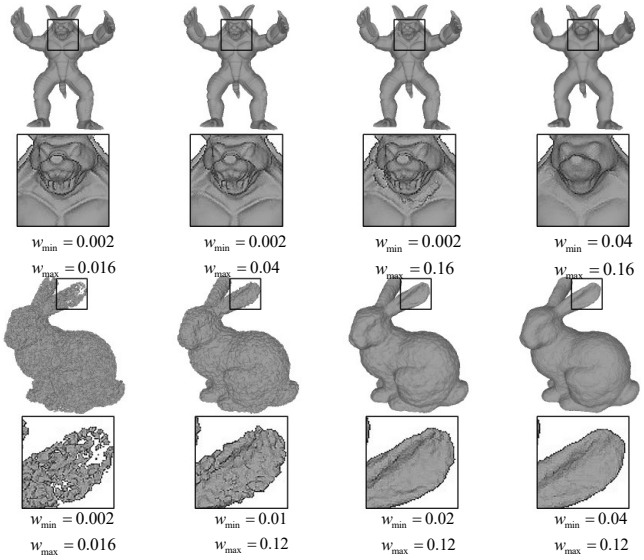


Figure 16: *Qualitative comparisons of different smoothing widths. The first and second rows are for clean and noisy points, respectively. The zoom-in view is also provided for the reconstruction details.*

We provide the recommended values for $w_{\min}$ and $w_{\max}$ for clean points and noisy models. That is: (1) $w_{\min} = 0.002, w_{\max} = 0.016$ for clean dense points, (2) $w_{\min} = 0.01, w_{\max} = 0.04$ for real-scan points, (3) $w_{\min} = 0.04, w_{\max} = 0.12$ for noisy points with 0.5% Gaussian noise, and (4) $w_{\min} = 0.05, w_{\max} = 0.2$ for sparse point clouds. We use linear scheduling and do not adopt any early termination strategy for the smoothing width during the PCG iteration process.

Steps in Normal Updating After solving the linear system through PCG, we add a normal updating linear operator U to improve the normal quality in our method. All prior experimental results contain the normal updating process. As previously discussed, this serves as a self-updating operator for μ, allowing for using in series. We investigate and discuss the optimal number of operators U.

We separately select the steps of normal updating operators to be 0, 1, 2, 4, and 8, and present the orientation results in Figure 17. Employing the normal updating operator U substantially improves the quality of normals. Nevertheless, as the number of iterations increases, the operator progressively converges to a fixed point, leading to diminishing returns in improvement until further enhancement

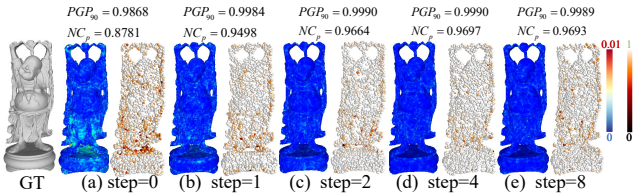


Figure 17: *Qualitative and quantitative comparison of our method's reconstruction and normal estimation with different steps of normal updating. Normal updating can significantly improve normal consistency and show marginal effects.*

becomes negligible. Hence, unless stated otherwise, we employ four consecutive normal consistency update operators in our experiments.

5. Limintation and Future Work

Non-manifold Surfaces or Open Scans With the motivation from electromagnetism, we leverage the Laplace partial differential equation, Gauss formula, and indicator function field to propose a new anisotropic solution framework. However, our method necessitates the target area Ω to be a bounded region, implying that the surface $\partial\Omega$ must be closed. Similar to other implicit methods, our method is unable to accommodate 3D shapes that do not conform to this condition, such as non-manifold surfaces and open scans.

As shown in Figure 18, our method underperforms on the open surface models. Although any other implicit field-based is also unsuitable for these cases, open scans are common in the real world, e.g., street scenes and interior decoration. Further exploration is needed for future comprehensive orientation and reconstruction of open surfaces. We are actively working towards integrating the Gauss formula or winding number with open surfaces in our future research efforts, which is meaningful.

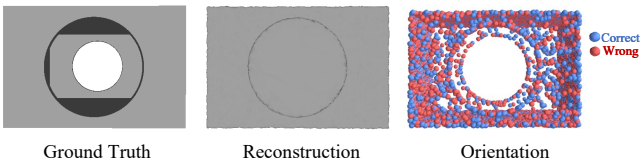


Figure 18: *The performance of our method is relatively limited for open surfaces, similar to other implicit methods.*

Solid Proof of Convergence The convergence of current state-of-the-art methods based on the isotropic Gauss formula or winding numbers still has significant shortcomings.

Our method partially alleviates this problem in comparison to PGR, GCNO, and WNNC. By introducing anisotropy and leveraging the flexibility of anisotropic coefficients, We construct additional linear equations with the ground truth as the solution, disregarding discrete errors. We employ the preconditioned conjugate gradient (PCG) algorithm to solve equations, which guarantees convergence from any starting point for N-order symmetric matrices within N steps. The normal updating operator can further improve orientation, and ground truth normal is a fixed point for this operator U.

However, without taking the discretization error into account, we admit the proof of convergence of our method is still not solid enough. Although it doesn't affect the good performance and contribution of our method, we still make efforts to provide the proof under continuous conditions by the double-layer potential theory [Blo78].

6. Conclusion

This paper presents our new research on the anisotropic Laplace equation and applies it to orientation and surface reconstruction by deriving the anisotropic Gauss formula.

By leveraging the flexibility of anisotropic coefficients, additional constraints are introduced to the indicator function. This allows us to construct more linear equations and solve them without the need for any regularization terms. After solving the equation, a normal updating operator U is proposed to further improve the oriented normal's quality. Regarding the space/time complexity, we propose the octree-based acceleration with $O(N)$ space complexity and $O(N\log N)$ time complexity. Our method can complete the reconstruction of point clouds below 500K within a few seconds.

Extensive experiments demonstrate that our method achieves state-of-the-art performance in both orientation and reconstruction, especially for models with thin structures, small holes or high genus.

7. Acknowledgments

We would like to thank all the anonymous reviewers for their comments and suggestions. This research was supported by the National Natural Science Foundation of China (92370125).

References

[BH86] BARNES J., HUT P.: A hierarchical o (n log n) force-calculation algorithm. *nature 324*, 6096 (1986), 446–449. 12

[Blo78] BLOCK L. P.: A double layer review. *Astrophysics and Space Science 55* (1978), 59–83. 15

[BSKG22] BEN-SHABAT Y., KONEPUTUGODAGE C. H., GOULD S.: Digs: Divergence guided shape implicit neural representation for unoriented point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 19323–19332. 3

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, Association for Computing Machinery, p. 67–76. URL: https://doi.org/10.1145/383259.383266, doi:10.1145/383259.383266. 2

[CT11] CALAKLI F., TAUBIN G.: SSD: Smooth signed distance surface reconstruction. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 1993–2002. 3

[EGO*20] ERLER P., GUERRERO P., OHRHALLINGER S., MITRA N. J., WIMMER M.: Points2surf learning implicit surfaces from point clouds. In *European Conference on Computer Vision* (2020), Springer, pp. 108–124. 1, 3, 10

[GH24] GOTSMAN C., HORMANN K.: A linear method to consistently orient normals of a 3d point cloud. In *ACM SIGGRAPH 2024 Conference Papers* (New York, NY, USA, 2024), SIGGRAPH '24, Association for Computing Machinery. URL: https://doi.org/10.1145/3641519.3657429, doi:10.1145/3641519.3657429. 1

[GYH*20] GROPP A., YARIV L., HAIM N., ATZMON M., LIPMAN Y.: Implicit geometric regularization for learning shapes. In *Proceedings of the 37th International Conference on Machine Learning* (13–18 Jul 2020), III H. D., Singh A., (Eds.), vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 3789–3799. URL: https://proceedings.mlr.press/v119/gropp20a.html. 1, 3

[HCJ19] HUANG Z., CARR N., JU T.: Variational implicit point set surfaces. *ACM Trans. Graph. 38*, 4 (July 2019). URL: https://doi.org/10.1145/3306346.3322994, doi:10.1145/3306346.3322994. 1, 10, 12

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on computer graphics and interactive techniques* (1992), pp. 71–78. 3

[HFZ*24] HUANG G., FANG Q., ZHANG Z., LIU L., FU X.-M.: Stochastic normal orientation for point clouds. *ACM Trans. Graph. 43*, 6 (Nov. 2024). URL: https://doi.org/10.1145/3687944, doi:10.1145/3687944. 1

[HWW*22] HOU F., WANG C., WANG W., QIN H., QIAN C., HE Y.: Iterative poisson surface reconstruction (ipsr) for unoriented points. *ACM Trans. Graph. 41*, 4 (July 2022). URL: https://doi.org/10.1145/3528223.3530096, doi:10.1145/3528223.3530096. 3, 9, 10, 11

[HWW*24] HUANG Z., WEN Y., WANG Z., REN J., JIA K.: Surface reconstruction from point clouds: A survey and a benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024). 10

[IYS*13] IJIRI T., YOSHIZAWA S., SATO Y., ITO M., YOKOTA H.: Bilateral hermite radial basis functions for contour-based volume segmentation. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 123–132. 2

[Kaa88] KAASSCHIETER E. F.: Preconditioned conjugate gradients for solving singular systems. *Journal of Computational and Applied mathematics 24*, 1-2 (1988), 265–275. 6

[Kaz05] KAZHDAN M.: Reconstruction of solid models from oriented point sets. In *Proceedings of the third Eurographics symposium on Geometry processing* (2005), pp. 73–es. 3

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *Computer Science* (2014). 10, 12

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (2006), vol. 7, p. 0. 3

[KCRH20] KAZHDAN M., CHUANG M., RUSINKIEWICZ S., HOPPE H.: Poisson surface reconstruction with envelope constraints. In *Computer graphics forum* (2020), vol. 39, Wiley Online Library, pp. 173–182. 3

[KH13] KAZHDAN M., HOPPE H.: Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG) 32*, 3 (2013), 1–13. 1, 3, 10

[LC98] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*. 1998, pp. 347–353. 7

[LSL24] LIN S., SHI Z., LIU Y.: Fast and globally consistent normal orientation based on the winding number normal consistency. *ACM Trans. Graph. 43*, 6 (Nov. 2024). URL: https://doi.org/10.1145/3687895, doi:10.1145/3687895. 1, 3, 9, 10, 11

[LSSW19] LU W., SHI Z., SUN J., WANG B.: Surface reconstruction based on the modified gauss formula. *ACM Trans. Graph. 38*, 1 (2019), 2:1–2:18. URL: https://doi.org/10.1145/3233984, doi:10.1145/3233984. 1, 3, 6

[LWBW16] LIU S., WANG C. C., BRUNNETT G., WANG J.: A closed-form formulation of hrbf-based surface reconstruction by approximate solution. *Computer-Aided Design 78* (2016), 147–157. 2

[LXSW22] LIN S., XIAO D., SHI Z., WANG B.: Surface reconstruction from point clouds without normals by parametrizing the gauss formula.

ACM Trans. Graph. 42, 2 (Oct. 2022). URL: https://doi.org/10.1145/3554730, doi:10.1145/3554730. 1, 3, 4, 5, 8, 9, 11

[MGV11] MACÊDO I., GOIS J. P., VELHO L.: Hermite radial basis functions implicits. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 27–42. 2

[MHZ*21] METZER G., HANOCKA R., ZORIN D., GIRYES R., PANOZZO D., COHEN-OR D.: Orienting point clouds with dipole propagation. *ACM Trans. Graph. 40*, 4 (July 2021). URL: https://doi.org/10.1145/3450626.3459835, doi:10.1145/3450626.3459835. 3, 9

[MMX*24] MA Y., MENG Y., XIAO D., SHI Z., WANG B.: Flipping-based iterative surface reconstruction for unoriented points. *Computer Aided Geometric Design 111* (2024), 102315. URL: https://www.sciencedirect.com/science/article/pii/S0167839624000499, doi:https://doi.org/10.1016/j.cagd.2024.102315. 3, 9

[MPS08] MANSON J., PETROVA G., SCHAEFER S.: Streaming surface reconstruction using wavelets. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 1411–1420. 3

[MYR*01] MORSE B., YOO T., RHEINGANS P., CHEN D., SUBRAMANIAN K.: Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Proceedings International Conference on Shape Modeling and Applications* (2001), pp. 89–98. doi:10.1109/SMA.2001.923379. 2

[Naz09] NAZARETH J. L.: Conjugate gradient method. *Wiley Interdisciplinary Reviews: Computational Statistics 1*, 3 (2009), 348–353. 6

[Pfe86] PFEFFER W.: The divergence theorem. *Transactions of the American Mathematical Society 295*, 2 (1986), 665–685. 2

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). 1

[RLH*18] REN X., LYU L., HE X., CAO W., YANG Z., SHENG B., ZHANG Y., WU E.: Biorthogonal wavelet surface reconstruction using partial integrations. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 13–24. 3

[SJ22] SELLÁN S., JACOBSON A.: Stochastic poisson surface reconstruction. *ACM Trans. Graph. 41*, 6 (Nov. 2022). URL: https://doi.org/10.1145/3550454.3555441, doi:10.1145/3550454.3555441. 3

[SMB*20] SITZMANN V., MARTEL J., BERGMAN A., LINDELL D., WETZSTEIN G.: Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems* (2020), Larochelle H., Ranzato M., Hadsell R., Balcan M., Lin H., (Eds.), vol. 33, Curran Associates, Inc., pp. 7462–7473. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf. 1

[WCS06] WALDER C., CHAPELLE O., SCHÖLKOPF B.: Implicit surfaces with globally regularised and compactly supported basis functions. *Advances in Neural Information Processing Systems 19* (2006). 2

[WZX*23] WANG Z., ZHANG Y., XU R., ZHANG F., WANG P.-S., CHEN S., XIN S., WANG W., TU C.: Neural-singular-hessian: Implicit neural representation of unoriented point clouds by enforcing singular hessian. *ACM Transactions on Graphics (TOG) 42*, 6 (2023), 1–14. URL: https://doi.org/10.1145/3618311, doi:10.1145/3618311. 1, 3

[XDW*23] XU R., DOU Z., WANG N., XIN S., CHEN S., JIANG M., GUO X., WANG W., TU C.: Globally consistent normal orientation for point clouds by regularizing the winding-number field. *ACM Trans. Graph. 42*, 4 (2023), 111:1–111:15. URL: https://doi.org/10.1145/3592129, doi:10.1145/3592129. 1, 3, 9, 11

[ZML*22] ZHOU J., MA B., LIU Y.-S., FANG Y., HAN Z.: Learning consistency-aware unsigned distance functions progressively from raw point clouds. *Advances in Neural Information Processing Systems 35* (2022), 16481–16494. 10, 12