Eurographics Symposium on Geometry Processing 2025
M. Attene and S. Sellán
(Guest Editors)

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

Symmetrized Poisson Reconstruction

M. Kohlbrenner[1] H. Liu[2] M. Alexa[1] M. Kazhdan[2]

[1]TU Berlin, Germany [2]Johns Hopkins University, USA

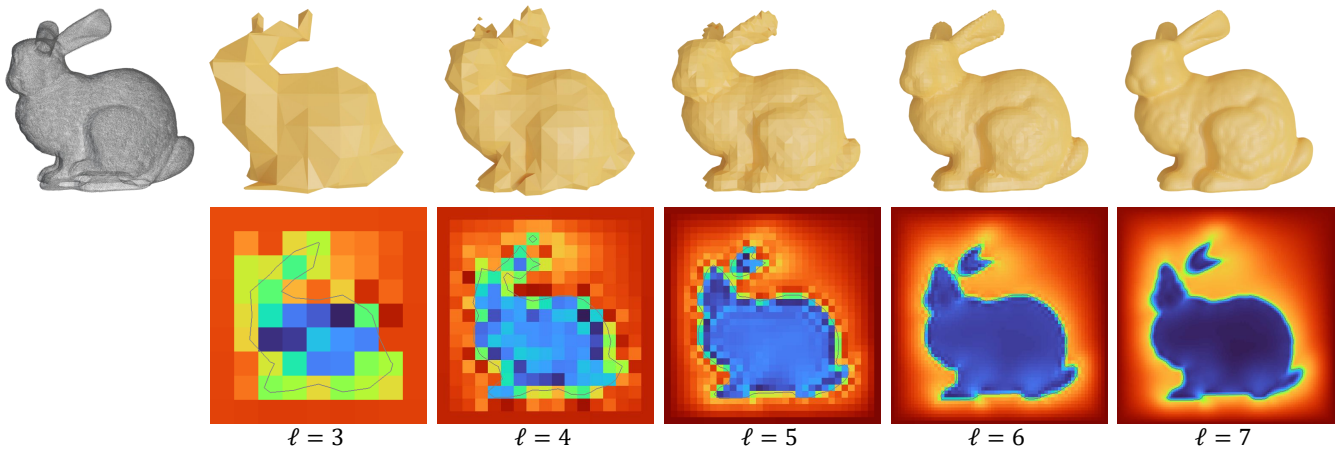

Figure 1: *Reconstructions of the Stanford* BUNNY *from raw scanner data at successively finer levels of the hierarchy, showing the surfaces and visualizing the restriction of the implicit function to a 2D plane, for the different levels* $3 \leq \ell \leq 7$.

Abstract

Many common approaches for reconstructing surfaces from point clouds leverage normal information to fit an implicit function to the points. Normals typically play two roles: the direction provides a planar approximation to the surface and the sign distinguishes inside from outside. When the sign is missing, reconstructing a surface with globally consistent sidedness is challenging.

In this work, we investigate the idea of squaring the Poisson Surface Reconstruction, replacing the normals with their outer products, making the approach agnostic to the signs of the input/estimated normals. Squaring results in a quartic optimization problem, for which we develop an iterative and hierarchical solver, based on setting the cubic partial derivatives to zero. We show that this technique significantly outperforms standard L-BFGS solver and demonstrate reconstruction of surfaces from unoriented noisy input in linear time.

Keywords: curve and surface reconstruction, outer product, polynomial optimization

CCS Concepts

• **Computing methodologies** → **Shape modeling;** • **Mathematics of computing** → *Nonlinear equations; Numerical analysis;*

1. Introduction

Reconstructing a surface from a point cloud sampling an unknown surface remains a classical problem in geometry processing. For robust and efficient reconstruction, most recent approaches represent the surface using an implicit formulation – a function $f : \mathbb{R}^3 \to \mathbb{R}$ whose level-set is the desired surface. The implicit function itself is typically defined as the minimizer of an energy that balances between fitting the input data and satisfying regularization constraints such as smoothness. While such approaches have shown a remarkable combination of efficiency and resilience to noise and outliers, they often require the normals of the surface as input – information that may not be provided with the data.

When no normal information is provided, it is straightforward to assign *unsigned* normals to the samples (e.g. via covariance analysis). However, defining a globally consistent sign is challenging. Previous implicit reconstruction approaches have addressed this problem, broadly, in one of three ways: (1) Signing the normals in a pre-processing step and then leveraging existing reconstruction techniques; (2) explicitly introducing signed normals as optimization parameters within the reconstruction framework; and (3) formulating the implicit function as an optimization that does not require signed normals.

In this work we follow the third approach, directly reconstructing the indicator function from point samples with *unsigned* normals. Our work is motivated by the recent extension of Poisson Surface Reconstruction (PSR) [KBH06] to the case of manifolds with co-dimension larger than one [KLAK23]. For the concrete case of co-dimension 2, that work represents the space spanned by two normal frame vectors using the anti-symmetric matrix defined by their alternating product – an encoding of the normal space that is independent of the choice of framing vectors. In a similar fashion, we start from samples with *unsigned* normals and take the *outer product*. This results in a *symmetric* matrix that encodes the normal to the surface independent of the sign of the normal vector. Similar to PSR, this per-sample input is extended to a smoothly varying matrix field, which is then used to define an energy that has the indicator function as its minimizer.

As with other methods that do not assume oriented normals as input, the energy is not quadratic and reconstruction does not reduce to the solution of a linear system of equations. Concretely, "squaring" the PSR method leads to a quartic energy function that is not necessarily convex and, consequently, it is more difficult to obtain useful local minima. We develop an iterative and hierarchical approach for minimizing this energy. On each level, we use coordinate descent, i.e., the cubic partial derivative is set to zero. We demonstrate that this is significantly more effective than optimization using black-box solvers such as L-BFGS.

Given the efficiency of the solver and the robustness to noise inherited from PSR, we obtain reconstructions of comparable quality to those of existing state-of-the-art. The benefit, similar to PSR, lies in solving on a hierarchical grid, yielding solutions at high resolution more efficiently compared to (CPU-based) implementations of other methods.

Outline of the Paper We review related work on surface reconstruction in section 2 and focus on the PSR framework in section 3. We present the quartic energy resulting from "squaring" the PSR formulation in section 4, where we also discuss the details of our iterative, hierarchical solver. We compare to related work in section 5 and conclude in section 6.

2. Related Work

In the following we review methods most closely related to our approach, focusing on *implicit* surface reconstruction from unoriented samples. For other types of surface reconstruction that do not require normals we refer to the literature [BTS*14, Dey06]. Though all implicit reconstruction technique can be used to define oriented normals at the input samples (e.g. by evaluating the gradient of the implicit function), we distinguish between methods that sign normals, methods that solve for the signed normals, and methods that reconstruct without estimating the signed normals.

2.1. Orienting Surface Samples

Typically, approaches assigning oriented surface normals to a point-set proceed in two steps.

Unsigned Normals Given a point-set $\mathcal{P} \subset \mathbb{R}^3$, an arbitrarily signed normal, $\tilde{n} : \mathcal{P} \to S^2$, is first estimated at each sample. In the pioneering work of Hoppe et al. [HDD*92] this was done using the smallest eigenvector of the local covariance matrix. However, more robust estimation techniques have since been proposed [MN03, LCL*23, CP05].

Signing Function Next, a signing function $\sigma : \mathcal{P} \to \{-1, +1\}$ is found that makes the normals $n \equiv \sigma \cdot \tilde{n}$ globally consistent.

Binary Assignment A common approach for defining σ is to build a weighted graph $\mathcal{G} = (\mathcal{P}, \mathcal{E}, \omega)$ with vertices representing samples, edges $(p,q) \in \mathcal{E}$ connecting nearby samples, and edge weights $\omega(p,q) \in [-1,1]$ characterizing the correlation of neighboring normals $\tilde{n}(p)$ and $\tilde{n}(q)$. Given the graph, a consistent orientation is obtained by solving for the function maximizing the correlation:

$$E(\sigma) = \sum_{(p,q)\in\mathcal{E}} \omega(p,q) \cdot \sigma(p) \cdot \sigma(q).$$

This is an unconstrained binary quadratic programming problem (discussed in detail by Schertler et al. [SSG17]), highlighting the difficulty in reconstructing a surface from samples without orientation.

Different definitions of neighborhood and different heuristics to tackle the global optimization problem have been suggested. Hoppe et al. [HDD*92] define neighbors using nearest Euclidean distances and edge weights by the dot-product of the unsigned normals $\omega(p,q) = \langle \tilde{n}(p), \tilde{n}(q) \rangle$. The optimization problem is made tractable by replacing the graph with its minimum spanning tree. Then the optimal solution is obtained by assigning an arbitrary sign to the normal at a source vertex and propagating the sign to neighbors by flipping the sign when the edge weight is negative. Because the sign at a vertex is defined by the parity of sign flips along the path from the source, and because the tree is acyclic, the sign function is uniquely defined.

König and Gumhold [KG09] analyze and extend this and other [XWH*03] previously defined flipping energies. There are a variety of other constructions for both the graph and the edge weights [HLZ*09, SBY11, GG07, MHZ*21], Recently, heuristics for solving the optimization problem on graphs containing cycles have also been suggested [SSG17, JBG19].

Stokes' Theorem Also starting with unsigned normals and using the nearest-neighbor graph, Gotsman and Hormann [GH24] use an entirely different approach to sign the normals. They relate the problem of consistently signing normals to Stokes' Theorem. In particular, they show that a soft assignment of signs can be obtained

by starting with a set of vector fields and using those to define a system of linear equations discretizing Stokes' Theorem over the input samples. Solving the system and clamping to $\{-1,+1\}$ gives the desired signing function.

2.2. Direct Normal Optimization

Starting with arbitrary normals, iterative Poisson Surface Reconstruction [HWW*22] proceeds by iteratively invoking PSR to reconstruct a surface and using the computed surface to correct the normal estimates. The correction is motivated by the parity test for interior/exterior classification and the corrected normals are used in the next iteration.

A different approach has been to leverage the linear relationship between the indicator function $\chi(p)$ of a surface and the area-weighted normals (as characterized by the formula for the generalized winding number [BDS*18] and by the Gauss formula in potential theory [LSW19]):

$$\chi(q) \approx \sum_{p\in\mathcal{P}} -\frac{q-p}{4\pi|q-p|^3}\cdot n(p) \tag{1}$$

Formulating a quadratic objective on the quality of the estimated indicator function (e.g. sample interpolation or smoothness) and using the linear relationship between the normals and the indicator function, the normals giving the optimal indicator function are obtained by solving a linear system of equations [XDW*23,LWZ*24, LXSW23].

Recently, Lin et al. [LSL24] proposed a variant that replaces the linear solve with an approximate fixed-point iteration scheme. Their approach still uses the linear relationship between the winding number and the normals but provides a solution amenable to a GPU-implementation, resulting in faster reconstruction.

2.3. Direct Surface Reconstruction

Combinatorial Approaches Combinatorial approaches connect the samples into a triangulation and often do not require normal data, instead using the distances between points [Dey06]. Compared to implicit methods, the interpolatory nature is prone to artifacts in the presence of noise. Concrete examples include the Advancing Front [CD04] and Scale Space [DMSL11] approaches.

Signing the Unsigned Similar to propagation-based approaches, Mullen et al. [MdGD*10] solve for a sign field in order to sign an unsigned distance function. Noting the quartic complexity of the optimization problem, they propose a heuristic based on ray-shooting to estimate a sign away from the data and a propagation technique to obtain a globally consistent sign field over the 3D domain. They recover a signed distance function from the unsigned distance and sign information by minimizing a quadratic energy. This idea has been extended to data with more noise and outliers [GCA13].

Voronoi-based Variational Reconstruction Alliez et al. [ACSTD07] define an anisotropic Dirichlet energy measuring the alignment of the implicit function's gradient with a symmetric matrix field that encodes unsigned normal direction and reliability. The field is almost singular near the data and uniform away from it. The function maximizing gradient alignment subject to a Bilaplacian constraint is found as the solution to a generalized eigenvalue problem.

Variational Implicit Point Set Surfaces Huang et al. [HCJ19] formulate the problem of surface reconstruction as an optimization problem, asking that the points be close to the zero-level set, the gradients at the points have unit length, and that the implicit function minimize a smoothness energy. Interestingly, the global problem of minimizing the smoothness energy and approximately satisfying the positional constraints can be turned into an optimization problem for the gradients. While the energy to be minimized is quadratic, the unit-norm constraint makes this an optimization problem on a Stiefel manifold, which is difficult to solve. As a result, the overall optimization problem has cubic complexity.

Neural Approaches Predictive neural models can be optimized to estimate shape properties such as signed normals from raw points [GKOM18, LFS*23] and generalize to unseen data. Neural networks are also successfully employed as parametric representations of implicit functions, and architectures such as SIREN [SMB*20] enable the fitting of functions as well as (higher) derivatives. Both Divergence Guided Shape Implicits [BKG22] and Neural-Singular-Hessian [WZX*23] build upon the SIREN architecture and propose custom loss functions for reconstruction based on properties of signed distance functions – that the divergence of gradient is to zero almost everywhere ([BKG22]) and that the Hessian is rank-deficient near the surface ([WZX*23]).

3. Background

Our approach extends the earlier work of Poisson Surface Reconstruction, replacing vector fields with symmetric matrix fields. We review these before proceeding to the description of our method in the next section.

Throughout, we will let $\Omega \equiv [0,1]^3$ be the domain over which implicit functions are computed. (Samples are translated so that the center of mass is at the center of the cube and rescaled to be inside.) We use the notation $\langle\!\langle \cdot,\cdot \rangle\!\rangle$ (resp. $\|\cdot\|^2$) to denote the inner-product (resp. norm) on the space of scalar/vector/matrix-fields. For example, for matrix fields M and N we have:

$$\langle\!\langle M,N \rangle\!\rangle \equiv \int_\Omega \langle M(p),N(p)\rangle_F\, dp$$

(where we use the Frobenius inner-product as the inner-product on the space of matrices).

3.1. Poisson Surface Reconstruction

Poisson Surface Reconstruction (PSR) [KBH06, KH13] is a state-of-the-art method for reconstructing the (smoothed) indicator function of a solid $M \subset \Omega$, given oriented samples of its boundary, ∂M.

Volume to Surface PSR leverages the fact that, for any smoothing kernel, the divergence theorem provides an expression for the gradient of the smoothed indicator function as an integral over the solid's boundary. Concretely, letting $\chi : \Omega \to \mathbb{R}$ denote the indicator

function, $K : \mathbb{R}^3 \to \mathbb{R}$ be the smoothing kernel, and $\vec{N} : \partial M \to S^2$ the surface normals:

$$\nabla(K * \chi)(q) = \int_{\partial M} K_q(p) \cdot \vec{N}(p)\, dp \tag{2}$$

where K_q denotes the kernel centered at q, $K_q(p) \equiv K(p-q)$.

Numeric Integration Given a set of oriented samples, $\mathcal{P} = \{(p,\vec{n})\} \subset \mathbb{R}^3 \times S^2$, the approach approximates the right-hand-side by a discrete summation

$$\nabla(K * \chi)(q) \approx \sum_{(p,\vec{n})\in\mathcal{P}} a_p \cdot K_q(p) \cdot \vec{n} \equiv \vec{V}(q) \tag{3}$$

with a_p an estimate of the surface area associated with sample p.

Gradients to Scalars An approximation of the smoothed indicator function $\tilde{\chi}$ is calculated as the function whose gradient best matches the vector field $\vec{V}$, and evaluates to zero at the samples. It is computed by solving for the function $\tilde{\chi}$ minimizing:

$$E(\tilde{\chi}) = \|\nabla\tilde{\chi} - V\|^2 + \alpha \sum_{(p,n)\in\mathcal{P}} \tilde{\chi}^2(p). \tag{4}$$

with α the *screening weight*.

Discretization PSR discretizes the problem by selecting a finite function basis $\{\Phi_i : \Omega \to \mathbb{R}\}$ and expressing the indicator function with respect to this basis, $\tilde{\chi}(p) = \sum_i x_i \cdot \Phi_i(p)$. Setting $\boldsymbol{x} = \{x_i\}$, the energy in eq. (4) is minimized at $\boldsymbol{x}$ satisfying the linear equation $(S + \alpha M^{\mathcal{P}})\boldsymbol{x} = \boldsymbol{b}$ with S the stiffness matrix, $M^{\mathcal{P}}$ the mass matrix defined by the samples, and $\boldsymbol{b}$ the (dual of the) divergence of the estimated gradient field:

$$S_{ij} = \langle\!\langle \nabla\Phi_i, \nabla\Phi_j \rangle\!\rangle, \quad M^{\mathcal{P}}_{ij} = \sum_{(p,\vec{n})\in\mathcal{P}} \Phi_i(p)\cdot\Phi_j(p), \quad b_i = \langle\!\langle \nabla B_i, \vec{V} \rangle\!\rangle. \tag{5}$$

3.2. Symmetric Outer Products

Our goal is to use unsigned normals to define an energy whose minimizer is the indicator function. A natural representation of unsigned normals is by their symmetric outer product, $n \to n \odot n$, where the symmetric outer product of vectors $v, w \in \mathbb{R}^3$ is:

$$v \odot w \equiv (v \cdot w^\top + w \cdot v^\top)/2.$$

This representation is independent of the sign of n and has linear structure, allowing us to take weighted combinations when transforming the unsigned normals into a symmetric matrix field.

This representation of line fields via symmetric matrices, using the outer product to remove the sign ambiguity, is a well-established technique in other contexts of computer graphics, such as for non-photorealistic rendering [PZ07]. In the context of surface reconstruction, it has been used in the anisotropic Dirichlet energy of [ACSTD07].

4. Method

At a point on an orientable surface, the unit normal provides a representation of the tangent space via perpendicularity and encodes the local orientation of the surface by its sign. While the tangent space can be estimated from the local covariance, inside/outside classification is a global property and hard to compute.

We show that the PSR framework can be extended to handle unsigned normals by representing the target unsigned orientation with a symmetric matrix field. In particular, in this extension the global problem of finding a consistent sign becomes part of the optimization procedure.

4.1. Energy

Recalling that in PSR the target implicit function, $\tilde{\chi}$ is related to the vector field $\vec{V}$ computed from the oriented samples via:

$$\nabla\tilde{\chi} \approx \vec{V},$$

we "square" both sides to define a quartic energy:

$$E(\tilde{\chi}) = \|\nabla\tilde{\chi} \odot \nabla\tilde{\chi} - \vec{V} \odot \vec{V}\|^2 + \alpha \sum_p \tilde{\chi}^2(p) \tag{6}$$

Now we seek an implicit function $\tilde{\chi}$ with the property that the symmetric product of its gradients matches a target tensor field, while still evaluating to zero at the input samples.

As with the formulation proposed by [MdGD*10], our energy is quartic in and its optimization is non-trivial due to the presence of local minima. However, our approach differs in that our energy encodes direction as well as sign. Furthermore, we propose a novel technique for minimizing the energy.

4.2. Approximating the Target Tensor Field

To define the energy, we approximate the target tensor $\vec{V} \odot \vec{V}$, distributing the outer product of normals with a smoothing kernel as in PSR, setting:

$$T(q) \equiv \sum_{(p,\tilde{n})\in\mathcal{P}} a_p \cdot K_q(p) \cdot \tilde{n} \odot \tilde{n}. \tag{7}$$

A key property of this approximation is that it is invariant to the sign of the normals. In particular, we use the standard approach of estimating the normal line at a point p by computing the smallest eigenvector of the covariance matrix defined by the k nearest-neighbors of the point. This defines a unit normal at each sample up to sign, the choice of which does not affect the definition of T.

Discussion Returning to the analytic formulation, the tensor field T defined above can be viewed as a discretization of the integral:

$$T(q) = \int_{\partial M} K_q(p) \cdot \vec{N}(p) \odot \vec{N}(p)\, dp. \tag{8}$$

This differs from the outer product of the vector field:

$$\vec{V}(q) \odot \vec{V}(q) = \Big[\int_{\partial M} K_q(p) \cdot \vec{N}(p)\, dp\Big] \odot \Big[\int_{\partial M} K_q(p) \cdot \vec{N}(p)\, dp\Big]. \tag{9}$$

In particular, while the two are equivalent in the case that the surface is flat (i.e. the surface normals are constant) and the kernel is scaled so that the integral of K_q over the surface equals one, the analytic integrals will be different in general.

An example of these fields is shown in Figure 2. In our approximation, the ellipses are not singular near regions of high curvature. Furthermore, the principal axes of these ellipses do not match

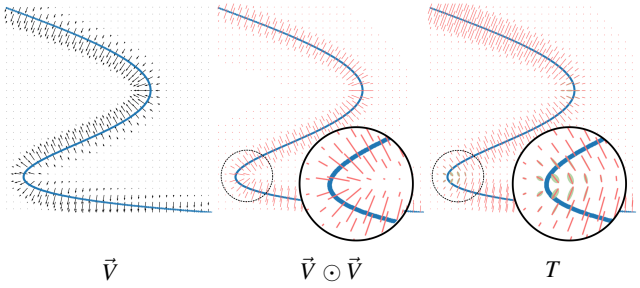

Figure 2: *Smoothed gradient field $\vec{V}$, the outer product field $\vec{V} \odot \vec{V}$ and our approximation T. As highlighted, errors in the approximation are more pronounced at sharper features, where the matrices of T are full-rank.*

the (non-singular) direction of the true outer product. This can be understood by considering the case of two adjacent, parallel, but oppositely oriented planes with q chosen to be a point equidistant from both. In this case, the outer product of the integral will be zero, because the oppositely aligned normals cancel, while the integral of outer products, computed by our approximation, will not.

We believe that part of this can be mitigated by estimating the extent to which the point-set is not co-planar around a sample and using that to down-weight the contribution of the outer product of the sample's normal to the tensor – a direction we intend to pursue in future work.

4.3. Discretization

As in PSR, we discretize by choosing a basis $\{\Phi_i : \Omega \to \mathbb{R}\}$, expressing the indicator function in this basis, $\chi(p) = \sum_i x_i \cdot \Phi_i(p)$. Setting $\boldsymbol{x} = \{x_i\}$, we get the energy (up to a constant):

$$E(\boldsymbol{x}) = \sum_{i,j,k,l} x_i \cdot x_j \cdot x_k \cdot x_l \cdot S_{ijkl} - 2\sum_{i,j} x_i \cdot x_j \cdot B_{ij} + \alpha \sum_{i,j} x_i \cdot x_j \cdot M^{\mathcal{P}}_{ij} \tag{10}$$

where $S \in \mathbb{R}^{N\times N\times N\times N}$ is a rank-4 tensor and $B \in \mathbb{R}^{N\times N}$ is a rank-2 tensor defined as:

$$S_{ijkl} = \langle\!\langle \nabla\Phi_i \odot \nabla\Phi_j, \nabla\Phi_k \odot \nabla\Phi_l \rangle\!\rangle \quad \text{and} \quad B_{ij} = \langle\!\langle \nabla\Phi_i \odot \nabla\Phi_j, T \rangle\!\rangle.$$

4.4. Implementation

Choosing a Basis We discretize the space of functions using first-order B-splines centered at the nodes of a regular grid with $R \times R \times R$ cells. Concretely, let $\phi^R_i : [0,1] \to \mathbb{R}$ be the 1D "hat" function centered at i/R:

$$\phi^R_i(s) = \begin{cases} s\cdot R - (i-1) & s\cdot R \in [i-1, i] \\ (i+1) - s\cdot R & s\cdot R \in [i, i+1] \\ 0 & \text{otherwise} \end{cases}.$$

Then for a multi-index $I = \{i_1, i_2, i_3\}$, with $i_j \in \{0, \dots, R\}$, indexing a grid node, the multivariate B-spline associated with node I is the product of the univariate B-splines:

$$\Phi^R_I(p) \equiv \phi^R_{i_1}(p_1) \cdot \phi^R_{i_2}(p_2) \cdot \phi^R_{i_3}(p_3).$$

Using this basis, we define the basis $\{\Pi^R_{IJ} \equiv \nabla\Phi^R_I \odot \nabla\Phi^R_J\}$ for symmetric matrix fields. We note that Π^R_{IJ} is a piecewise multi-quadratic function, that $\Pi^R_{IJ} = \Pi^R_{JI}$, and that Π^N_{IJ} is non-zero only if Φ^R_I and Φ^R_J have overlapping support – i.e. if $|I - J|_\infty \leq 1$.

In particular, it follows that the tensors M and B are sparse, with $O(R^3)$ non-zero entries. The expected number of non-zero entries in $M^{\mathcal{P}}$ is $O(R^2)$ since the samples live on a co-dimension one surface.

Computing the System Coefficients The entries of the rank-4 tensor M are obtained using finite-elements assembly – expressing the integral over Ω as the sum of integrals over the (at most 2^3) grid cells on which Φ^R_I, Φ^R_J, Φ^R_K, and Φ^R_L are simultaneously supported. Using the fact that the functions are strictly polynomial within each cell, these integrals are computed in closed-form.

For spacial efficiency, we leverage the fact that the entries of M are translation-invariant (away from the boundaries of the grid), and compactify the representation by using a stencil whose size is independent of grid resolution.

For simplicity, we approximate the matrix field $T : \Omega \to \mathbb{R}^{3\times 3}$ by a piecewise-constant field whose value within a cell is obtained by evaluating T at the cell center. Computing the coefficients of B is then performed using finite-elements assembly as above.

In practice, we obtain the contribution of sample $p \in \mathcal{P}$ to the matrix field T using covariance analysis on the $k = 20$ nearest neighbors to obtain a sign-invariant tensor $t = \tilde{n} \odot \tilde{n}$ and multi-linearly distributing the tensor t to the 2^3 corners of the cell within which p resides.

Relaxing the Solution Given an estimate for the solution $\boldsymbol{x} = \{x_I\}$ we relax the solution by performing multiple Gauss-Seidel relaxations. Within each Gauss-Sidel iteration we iterate over the individual coefficients, fixing all but the I-th coefficient, and compute the quartic polynomial in x_I. (As we do not need the constant term, the computation of the polynomial only requires considering coefficients x_J, x_K, x_L and entries $S_{IJKL}, M^{\mathcal{P}}_{IJ}, B_{IJ}$ where J, K, and L are within the one-ring of I.)

We differentiate the quartic, compute the zeros of the derivative, and set x_I to the root at which the quartic polynomial is minimized. By construction, this approach is guaranteed to monotonically reduce the energy.

Multigrid For better convergence, and to avoid getting trapped in local minima, we leverage a hierarchical solver. To this end, we use the fact the space spanned by B-splines defined over an $R \times R \times R$ grid is a subspace of the space spanned by B-splines defined over a $2R \times 2R \times 2R$ grid. In particular, if P is the prolongation matrix taking coarse scalar field coefficients to fine scalar field coefficients, then the fine constraints B^{2R} and coarse constraints B^R are related as $B^R = P^\top \cdot B^{2R} \cdot P$. (Recall that B^R and B^{2R} are rank-2 tensors.)

Similar to [KLAK23] this allows us to design a coarse-to-fine

solver in which we initialize the system by computing B at the finest resolution $R = 2^L$ and successively restrict the constraint to get the constraints at levels $\bar{L} \leq \ell \leq L$. We then solve the system in a coarse-to-fine manner, performing multiple passes of Gauss-Seidel relaxation at level ℓ, and then prolonging the estimated solution $\boldsymbol{x}^\ell$ to an initial guess $\boldsymbol{x}^{\ell+1}$ at the next level.

At the coarsest resolution, $\bar{L}$, we initialize the solution by setting a node's coefficient to the distance of the node from the center of the grid. This corresponds to the assumption that the reconstructed surface should be centered around the center of Ω, introduces a bias that the normals of the reconstructed surface should be outward-facing, and avoids the issue of the critical point at $\boldsymbol{x} = \mathbf{0}$.

Figure 1 shows reconstructions of the Stanford bunny from raw sensor data obtained at levels $\ell \in \{3,\dots,7\}$ visualizing both the surfaces (top) and the restriction of the implicit functions to a 2D plane passing through the volume (bottom).

Adaptive Discretization Though a direct implementation would use *all* the nodes of an $R \times R \times R$ grid, this would result in an algorithm with a space/time complexity of $O(R^3)$. By comparison, we expect the complexity of an input point cloud sampling a co-dimension one manifold to have complexity $O(R^2)$.

As our aim is to reconstruct the indicator function $\tilde{\chi}$, we leverage the observation from PSR that the piecewise-constant nature of the indicator function implies it can be accurately represented by an adaptive hierarchy of functions in which only nodes in the proximity of the input are used to span the space of functions. In particular:

- We represent the implicit function as the linear combination of first-order B-splines from across the different levels of the multi-resolution hierarchy (rather than just the B-splines at the finest resolution, for the regular discretization)
- We solve the system in a coarse-to-fine fashion, with the finer levels of the hierarchy "correcting" for the residual constraints not satisfied at the coarser resolutions.

As with PSR, the hierarchical solver only requires a constant number of iterations per level, so that the overall space/time complexity of our adaptively discretized solver is $O(R^2)$. This can be seen in table 1 and fig. 3 which compare the running time and peak memory utilization of the regular and adaptive discretizations as a function of the number of levels in the hierarchy. As expected, using a regular solver, complexity increases by a factor of roughly 8 with each level, whereas with the adaptive solver complexity increases by a factor of roughly 4.

And, as with PSR, we have found little difference in the quality of results. The primary difference being in regions where the extracted level-set is far from the input samples. In such regions a regular discretization tends to produce a smoother surface, due to the use of refined B-splines away from the samples.

Comparison to Kohlbrenner et al. As in the work of Kohlbrenner et al. [KLAK23], we use vector products to define orientation and solve a hierarchical system to obtain the geometry. However, our implementation differs in two important ways.

- Restricting our energy to a single coefficient gives a quartic polynomial. (In contrast, due to the anti-symmetry of the matrix representation, the restriction of the earlier approach is quadratic.)
- We solve using an adaptive discretization, significantly improving the performance of the solver. (Using an analogous discretization, we believe the earlier method could similarly be accelerated.)

4.5. Regularization

In our implementation, we would like the reconstructed surface to be contained entirely within the unit cube so we seek a solution χ that is constant along the boundary $\partial\Omega$. A direct approach would be to fix the value along the boundary to a constant. However, as we only know the solution up to scale we do not know what that constant should be. Instead, we modify the energy by adding the integrated squared norm of the *tangential* component of the gradient of χ along the boundary. That is, we update the energy in eq. (10):

$$E(\boldsymbol{x}) = \cdots + \beta \sum_{I,J} x_i \cdot x_j \cdot D_{IJ} \tag{11}$$

where β is the regularization weight and D is a rank-2 tensor with:

$$D_{IJ} = \int_{\partial\Omega} \left\langle \pi_p\left(\nabla \Phi_I^R(p)\right), \pi_p\left(\nabla \Phi_J^R(p)\right) \right\rangle dp$$

with π_p the projection onto the tangent space at p. We note that the tensor is sparse, with a non-zero entry at coefficient (I,J) only if the intersection of the supports of Φ_I and Φ_J intersects the boundary:

$$\text{supp}(\Phi_I) \cap \text{supp}(\Phi_J) \cap \partial\Omega \neq \emptyset.$$

Effects of the regularization are visualized in the 2D restrictions in fig. 1, with grid values along the boundary all having the same color.

5. Results

In this section we evaluate our approach. We start by considering run-time performance. Next we consider the solver itself, comparing it to the standard L-BFGS solver and discussing the problem of local minima. Finally, we present a number of reconstruction results, comparing to existing approaches.

In all experiments, we estimate unsigned normals using the covariance matrix computed from the $k = 20$ nearest neighbors. Unless otherwise noted, our hierarchical solver starts at level $\bar{L} = 4$ and solves up to level $L = 9$. We use 512 Gauss-Seidel iterations at the coarsest level, and 16 Gauss-Seidel iterations at all other levels. We use standard multi-coloring to parallelize the Gauss-Seidel relaxation. Nearest-neighbor computation is performed using the kD tree implementation in Trimesh2 [Rus20]. An implementation of our approach is available at github.com/mkazhdan/SymmetricPoissonReconstruction.

5.1. Performance

Using a hierarchical solver with a fixed number of relaxations per level, the cost of the solve is linear in the number of non-zero entries in the system matrix. As we use a first-order B-spline basis to represent scalar functions, the number of non-zeros is proportional

		$L=6$	$L=7$	$L=8$	$L=9$	$L=10$
(R)	Time (s)	3.88	9.93	43.16	342.59	-
	Memory (MB)	205	541	2820	20190	-
(A)	Time (s)	2.97	4.84	12.22	46.85	189.05
	Memory (MB)	221	501	1630	6689	25444

Table 1: *Runtime and peak memory usage for reconstructing the bunny from raw point data in 3D as a function of the finest level, L, using a regular (R) / adaptive (A) grid Results were obtained using an 12th Gen Intel(R) Core(TM) i9-12900K, 31 GB of RAM.*

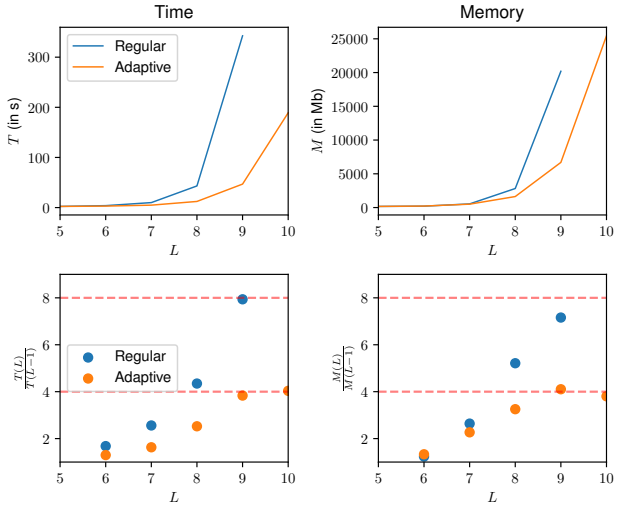

Figure 3: *Absolute values (top) and relative increase (bottom) in time (left) and memory (right) in function of the maximum depth L from ≈ 361k samples of the Bunny.*

to the number of basis functions. As discussed in section 4.4, using a resolution of R the reconstruction complexity (both temporal and spatial) is $O(R^3)$ for the regular discretization and $O(R^2)$ for the adaptive one. Table 1 and fig. 3 confirm the complexity, showing the empirical performance for reconstructions at different resolutions and highlighting the relative increase from the previous level.

5.2. Comparison to L-BFGS

To assess our solver, we compare the quality and performance of our approach to that of low-memory BFGS (L-BFGS) [LN89] as implemented in the NLopt package [Joh07].

Due to the slow running time of L-BFGS, we restricted our evaluation to the reconstruction of a 2D point cloud, solved at level $L = 8$ (i.e. on a grid of resolution 256×256). We also experimented with the preconditioned truncated Newton method with restarting [DS83] and the method of moving asymptotes [Sva02], but found them to be too slow. (For L-BFGS, we also experimented with fixing the number of gradients stored in memory, using a history of between 5 and 20 gradients, but found that the default values used by the package worked as well.)

	iters.=512	iters.=1024	iters.=2048	iters.=4096
L-BFGS	4.5×10^4	9.0×10^3	3.3×10^3	3.2×10^3
Ours	3.8×10^5	2.6×10^5	1.3×10^5	4.9×10^4
L-BFGS	49 (s)	103 (s)	194 (s)	464 (s)
Ours	15 (s)	26 (s)	49 (s)	117 (s)

Table 2: *Single-level fitting error (top) and solver time in second (bottom) as a function of the number of iterations, for reconstructing a 2D point-cloud*

Single-level Solvers We start by considering a single-level solver. Table 2 gives the fitting error (top) and solver time (bottom) for both methods. As the table shows, the L-BFGS algorithm does the best job of reducing the fitting error, significantly outperforming ours. However, the table also makes it clear that even for the simple 2D problem, neither approach works well as a single-level solver, as both fail to converge even after hundreds of seconds.

Hierarchical Solvers We also consider the effects of using a hierarchical solver for computing the indicator function. As before, we solve for the indicator function at level $L = 8$, but this time using a hierarchical solver starting at a level $\bar{L}$, with $1 \leq \bar{L} \leq L$. We experiment with both 512 and 4096 iterations at level $\bar{L}$, and continue using 16 iterations at all other levels.

Figure 4 shows the fitting error (left) and solver time (right), both plotted on a logarithmic scale as a function of $\bar{L}$. The order of the horizontal axes is reversed so that the solver becomes "more hierarchical" moving from left to right. Note that for $\bar{L} = L = 8$, we reproduce the results from table 2.

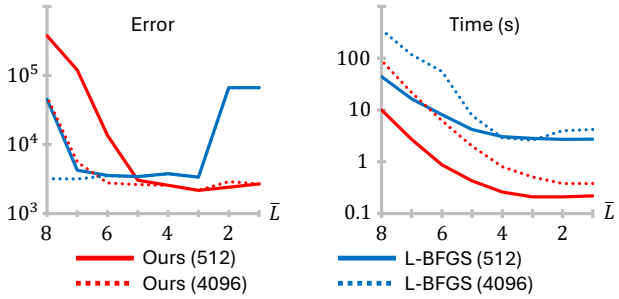

Figure 4: *Hierarchical solver fitting error (left) and solver time (right) as a function of the coarsest level of the hierarchy, $\bar{L}$.*

Examining the timing (right) we observe that the solvers become more efficient as they are more hierarchical. This is expected because the solve at the coarsest level uses significantly more iterations than the solves at the finer levels. Additionally, we find that increasing the number of iterations at the coarsest level makes the solver more computationally expensive.

Examining the fitting error (left) we note that while the performance of the L-BFGS solver is not improved with the hierarchy (in fact, fitting error *increases* uniformly with decreased $\bar{L}$, when using 4096 iterations), our solver's error decreases as the coarsest level of

the hierarchy does, until $\bar{L} \approx 3$. We believe that at too low a resolution (e.g. $\bar{L} < 3$) the system is too coarse to provide a meaningful initial guess.

In particular we note that using a hierarchical solver with $\bar{L} = 3$, we achieve a smaller fitting error than the best result of L-BFGS (2135 vs. 3191) at a fraction of the time (0.29 seconds vs. 464 seconds).

5.3. Local Minima

As noted above, in addition to providing a more efficient solver, a hierarchical approach also produces reconstructions with lower fitting error. We believe this is because the fitting energy is quartic, making it possible for naive optimization to get trapped at an undesired local minimum. Using a hierarchical solver, we "smooth" the energy landscape at lower resolutions, obtaining an approximate solution in the right valley. The solution is then improved at finer levels of the hierarchy.

As an example, fig. 5 shows the reconstruction of the cow's head obtained with a hierarchical (left) and a single-level (right) solver. Using a single-level solver, the zero level-set still passes near the input samples. However, the gradients of the implicit function are not consistently oriented, resulting in a Swiss-cheese reconstruction. In contrast, the surface reconstructed using a hierarchical solver has no such artifacts. This can also be seen by examining the values of the implicit function on a cutting plane passing through the 3D volume (top), which shows that the hierarchical solver produces a function more closely resembling an indicator function.

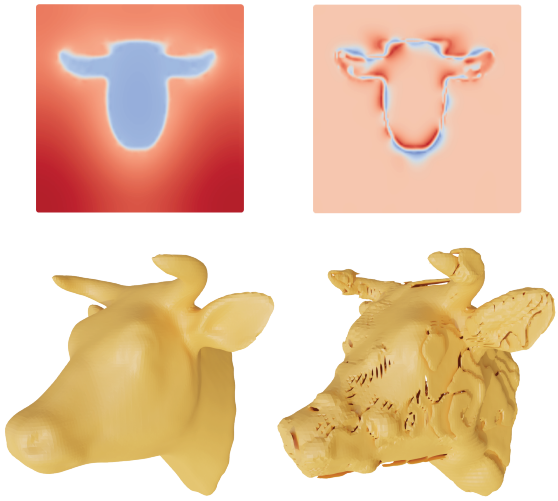

Figure 5: *Without a hierarchy, we obtain a locally optimal solution for the cow head (right column). Inconsistent gradient sign changes are clearly visible in the slice through the indicator function (upper right). The hierarchical approach finds a globally consistent solution (left column).*

5.4. Comparison

We compare our method (SymPR) with the results obtained by iterative Poisson Reconstruction (iPSR) [HWW*22], Winding Number Normal Consistency (WNNC) [LSL24], Parametric Gauss Reconstruction (PGR) [LXSW23], Neural Singular Hessian (NSH) [WZX*23], as well as the *Advancing Front* (AF) [CD04] and *Scale Space* (ScSp) [DMSL11] methods implemented int the *CGAL* library. For methods that are not part of *CGAL*, we use the implementation provided by the authors with standard parameters, unless noted otherwise. We found 5000 training iterations to be sufficient for NSH, producing good results and still running comparably quickly. We set the maximum facet size parameter of *Advancing Front* to 0.1 times the bounding box diagonal.

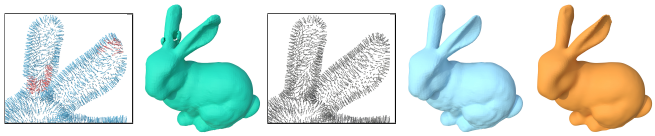

Figure 6: *Wrongly oriented normals from [GH24] can lead to artifacts in subsequent PSR output (left). Normals that point in the same same (resp. opposite) direction as the ground truth (GT) normals are colored blue (resp. red). Our SymPR reconstruction from unoriented points (right) is close to the PSR reconstruction from ground truth normals (center).*

We do not compare to the method of Huang et al. [HCJ19], as we found it to be limited by the size of the input point-cloud. We also do not compare to the method of Xu et al. [XDW*23], despite the high quality reconstructions presented in the paper (including on complicated inputs for which iPSR and other methods fail), because the reported run-times are orders of magnitude slower than for any of the compared methods.

We have experimented with the recent technique of Gotsman and Horman [GH24] based on an implementation generously provided by the authors. As it temporarily uses a dense representation of the system matrix, it is unable to process the point sets we use for comparisons. We were able to orient smaller point sets, which we then fed to the PSR algorithm to generate a surface. We found that orienting the points could be delicate when the sampling density was small relative to the reach, and it was difficult to find parameters producing consistently oriented normals. Figure 6 shows an example for the BUNNY sampled with 8.5K points, visualizing the reconstruction based on normals obtained by the method of Gotsman and Horman. Their approach has trouble generating correctly aligned normals near the base of the Bunny's right ear, resulting in spuriously reconstructed geometry. In contrast, our approach correctly reconstructs the geometry.

Both Voronoi-based Variational Reconstruction (VBVR) [ACSTD07] and Signing the Unsigned (STU) [MdGD*10] are closely related to our approach. Though reference implementations are not openly available, we were able to experiment with code generously provided by the authors. Both use tetrahedral meshes for discretization and subsequently depend on their quality. This makes direct comparison of reconstruction quality difficult. In our experience, both methods can produce high quality results, as in fig. 7. However care is required in choosing parameters. (Additionally, we have found that computation bottlenecks limit the applicability to

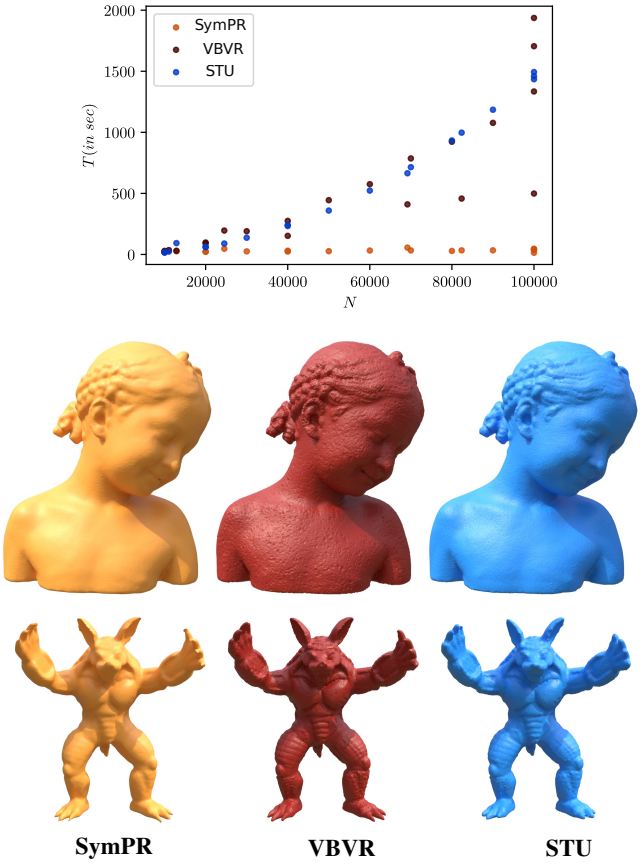

Figure 7: *Execution time in function of the number of points N (top) and reconstruction examples (bottom) for SymPR (ours), VBVR, STU (left to right columns).*

smaller meshes.) As such, we did not include these in further comparisons.

Runtime PGR works well and is fast for very small meshes. However, it scales poorly with input size both in time and memory. In our experience this limits the application to inputs with not more than around 40K points.

For the remaining methods, runtimes on a subset of the EPFL statues dataset [EPF14] are shown in fig. 8. For both NSH and our method, computation is dominated by the the number of degrees of freedom in the system and the number of iterations, both held constant across the different experiments. Thus, we see a near-constant running time for both methods. In contrast, the other approaches bottle-neck on per-point calculations, resulting in running times that grow with the size of the input. In particular, we see that our proposed approach is noticeably faster than existing methods. (Though we have not explored this, it would also be informative to consider the performance of these methods on appropriately down-sampled subsets of the point-clouds.) Both NSH and WNNC leverage GPU parallelization.

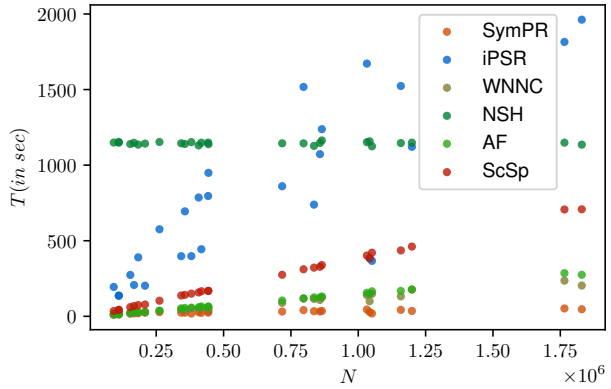

Figure 8: *Runtime for different methods on a subset of the EPFL statues dataset.*

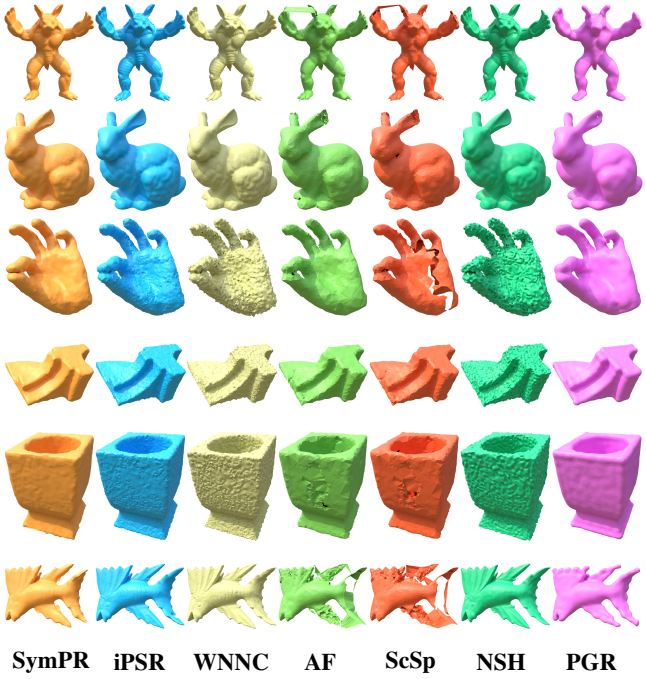

Figure 9: *Reconstruction results on point-clouds with fewer than 40K samples.*

Small Point-Clouds We show reconstruction results on smaller point-clouds (thereby allowing the evaluation of PGR) in fig. 9. We notice artifacts in the combinatorial approaches *Advancing Front* and *Scale Space*, but the other methods recover the general shape successfully. PGR exhibits more smoothing, with iPSR,WNNC, and NSH better reproducing the fine detail in the BUNNY, FANDISK, ARMADILLO, and WOODFISH models, but also generating more noisy reconstructions for the HAND, FANDISK, and VASE.

Larger Point-Clouds We compare results for larger meshes in fig. 10. While *Advancing Front* and *Scale Space* show impressive

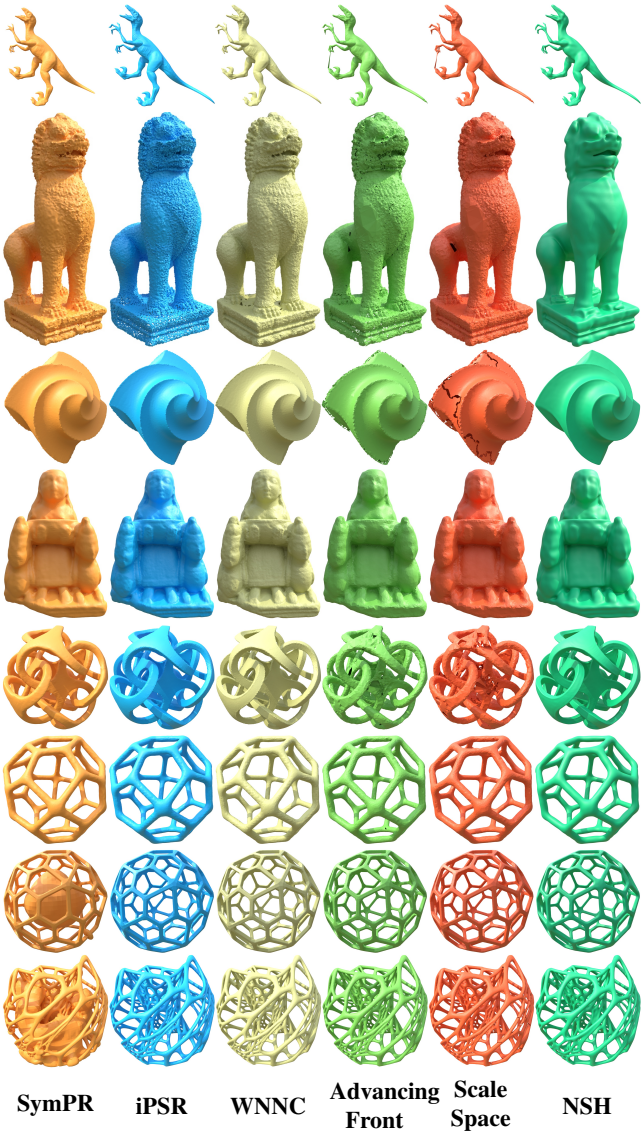

Figure 10: *Reconstruction results for larger point-clouds, including difficult high genus objects*

results on difficult, high genus data (bottom three rows), they also appear to have trouble at sharp corners (top five rows). This may be because it is difficult to propagate normal orientation across right angles (since the inner-product is close to zero) and harder still to propagate across sharp creases and thin features where a consistent orientation would have the normals oppositely oriented on the two sides.

iPSR, WNNC, and NSH perform well on all models, with iPSR and WNNC faithfully reproducing the detail in the models and NSH having trouble with fine features (e.g. the LION in the second row).

SymPR faithfully reconstructs most models, though it does have trouble with the thin, high-genus models in the two last rows. We

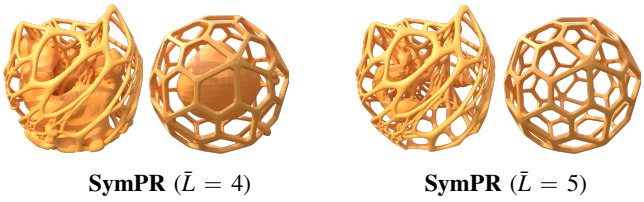

Figure 11: *For some models, the choice of coarsest level, $\bar{L}$, used for initialization affects recovery of correct orientation at higher levels.*

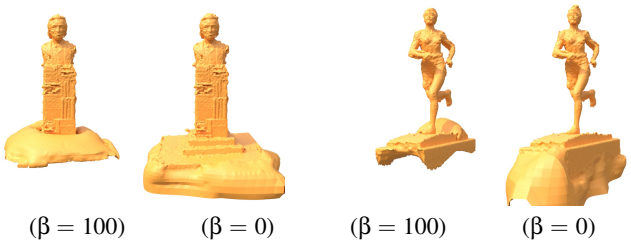

Figure 12: *The closed mesh assumption enforced by boundary conditions can cause artifacts on open meshes ($\beta \neq 0$). Disabling the regularization can avoid this ($\beta = 0$).*

believe that this may be due to the quartic optimization energy's having multiple local minima. As shown in fig. 11, changing the coarsest level from $\bar{L} = 4$ to $\bar{L} = 5$ somewhat mitigates the issue.

Real-World Data We also compare the approaches on point-clouds from EPFL's Statue Model Repository [EPF14]. The datasets are obtained by taking multiple photographs of each statue and using structure-from-motion to produce the 3D point-clouds. As such, the datasets suffer from noise, misalignment, and occlusion. Figure 15 shows results for nine of the statues. As in the previous experiment, we find that *Advancing Front* and *Scale Space* introduce geometric artifacts while NSH tends to provide more smoothed-out reconstructions. iPSR and WNNC tend to do a better job reproducing the fine-features, but also tend to interpolate noise.

Dirichlet Regularization We note that the introduction of Dirichlet boundary constraints in eq. (11) assumes that the point-cloud represents a water-tight mesh. This often does not hold for real-world data where physical restrictions constrain the scanning directions. (For example, scanning statues, it is not typically possible to obtain data from the bottom the base.) As shown in fig. 12, better results may be obtained when removing this regularizer.

Accuracy We uniformly sample $N \in \{10K, 100K, 500K, 2M\}$ points from the Bimba model and compare the reconstructions to the ground truth in fig. 13. Methods that discretize using the points directly (WNNC, Advancing Front, and Scale Space) exhibit the expected trade-off between reconstruction time and quality as a function of the number of samples. Methods that discretize using a fixed resolution grid (NSH and SymPR) tend to have the same reconstruction time and quality, independent of the number of sam-

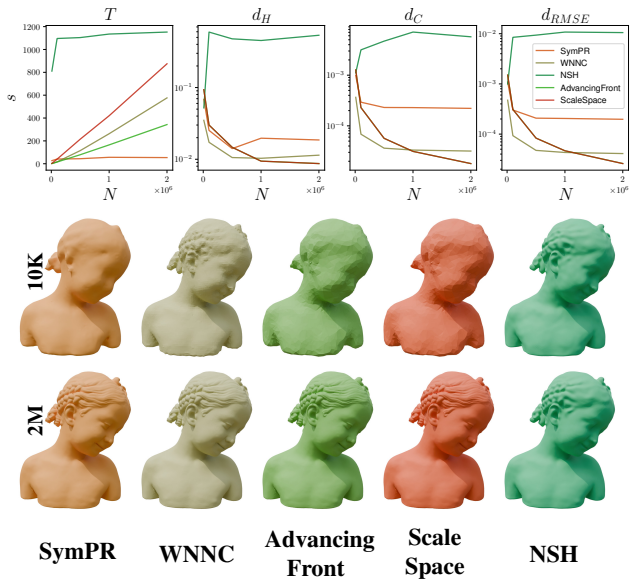

Figure 13: *Reconstructions from regular samples of the Bimba model. The plots show execution time T and Hausdorff (d_H), Chamfer (d_C) and root mean squared error (d_{RMSE}) to the ground truth mesh in function of the number of samples N (top). Distances are in units of bounding box diagonal. We show reconstructions for $N = 10K$ and $N = 2M$ (bottom).*

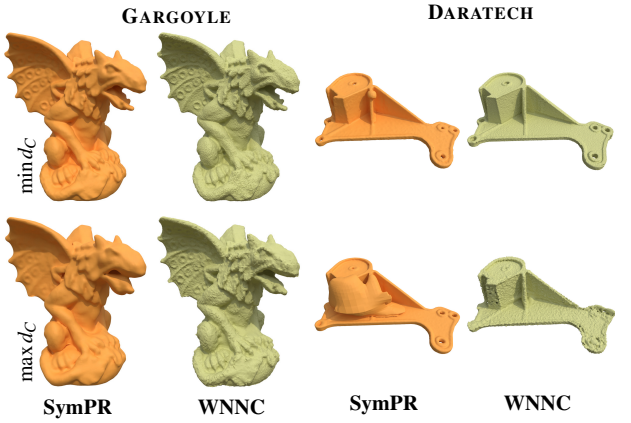

Figure 14: *SymPR and WNNC reconstructions with smallest (top row) and largest (bottom row) Chamfer distance d_C among the* 48 *point sets of the* GARGOYLE *and* DARATECH *model from the Berger et al. benchmark [BLN*13].*

ples. For these, the reconstruction quality is limited by the resolution of the grid which is effectively 256^3 for NSH and 512^3 for SymPR.

We also evaluate our method on the benchmark dataset of Berger et al. [BLN*13]. It consists of simulated scans of objects of different complexity as well as the ground truth meshes. The scans feature a range of common scanning artifacts. We show a boxplot of the Chamfer distance in fig. 16, and the numerical values for additional metrics in table 3. WNNC tends to have the lowest average reconstruction errors but its reconstructions exhibit surface artifacts in the presence of noise (see fig. 14). Our method achieves competitive results but suffers from wrongly oriented parts on the difficult DARATECH model. While all methods can lead to additional connected components and holes, this is particularly common for WNNC as indicated by the average deficiencies ΔCC and ΔG.

6. Conclusion and Future Directions

We presented a novel approach for surface reconstruction from unoriented points. Squaring the variational formulation of PSR, we showed that the indicator function can be expressed as the minimizer of a quartic energy whose coefficients can be approximated using only unsigned normal information. We developed a novel, hierarchical solver, leveraging the multi-resolution structure and relaxing the system by coordinate descent. Empirically, we have shown that our solver out-performs black-box non-linear solvers, and efficiently produces high quality reconstructions even for challenging, real-world data.

Future Adaptations

- *Improved Initial Guess* As the energy is quartic, we cannot guarantee that the solution is a global minimum. This is particularly challenging as models with thin structures may not have a good representation at coarser resolutions. We will explore if this can be mitigated by leveraging a full V-cycle solver, allowing the higher-resolutions to first smooth the residual, producing a constraint that is better represented at the coarser levels.
- *Tensor Approximation* As discussed in section 4.2 our approach of splatting the outer-product is only an approximation of the outer-product of the splats. We will explore ways to tighten this approximation by adaptively rescaling the splatting kernel, and using local covariance information to simulate the cancellation of normals that may occur at thin features and regions of high curvature.
- *Efficient Implementation* Though significantly faster and less memory intensive than existing CPU-based implementations, our solver is only marginally faster than the GPU-based implementation of WNNC. We will explore adapting our implementation to run on the GPU to further improve run-time performance.

Acknowledgements

We would like to thank Pierre Alliez for the code of STU and VBVR and the helpful discussion regarding the approaches. We would also like to thank Craig Gotsman and Kai Horman for providing us with the implementation of their method. Funded, in part, by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101055448, ERC Advanced Grand EMERGE). Open Access funding enabled and organized by Projekt DEAL.

References

[ACSTD07] ALLIEZ P., COHEN-STEINER D., TONG Y., DESBRUN M.: Voronoi-based variational reconstruction of unoriented point sets. In

Figure 15: *Reconstruction results on a selection of models from the EPFL statues dataset. The data is noisy, unevenly sampled, contains holes and most models are open at the bottom.*

Proceedings of the Fifth Eurographics Symposium on Geometry Processing (Goslar, DEU, 2007), SGP '07, Eurographics Association, p. 39–48. 3, 4, 8

[BDS*18] BARILL G., DICKSON N. G., SCHMIDT R. M., LEVIN D. I. W., JACOBSON A.: Fast winding numbers for soups and clouds. *ACM Trans. Graph. 37*, 4 (2018), 43. doi:10.1145/3197517.3201337. 3

[BKG22] BEN-SHABAT Y., KONEPUTUGODAGE C. H., GOULD S.: DiGS : Divergence guided shape implicit neural representation for unoriented point clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022* (2022), IEEE, pp. 19301–19310. doi:10.1109/CVPR52688.2022.01872. 3

[BLN*13] BERGER M., LEVINE J. A., NONATO L. G., TAUBIN G., SILVA C. T.: A benchmark for surface reconstruction. *ACM Trans. Graph. 32*, 2 (Apr. 2013). URL: https://doi.org/10.1145/2451236.2451246, doi:10.1145/2451236.2451246. 11, 13, 14

[BTS*14] BERGER M., TAGLIASACCHI A., SEVERSKY L. M., ALLIEZ P., LEVINE J. A., SHARF A., SILVA C. T.: State of the Art in Surface Reconstruction from Point Clouds. In *Eurographics 2014 - State of the Art Reports* (2014), Lefebvre S., Spagnuolo M., (Eds.), The Eurographics Association, pp. 161–185. doi:10.2312/egst.20141040. 2

[CD04] COHEN-STEINER D., DA F.: A greedy delaunay-based surface reconstruction algorithm. *Vis. Comput. 20*, 1 (2004), 4–16. doi:10.1007/S00371-003-0217-Z. 3, 8

[CP05] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. *Comput. Aided Geom. Des. 22* (2005), 121–146. doi:10.1016/J.CAGD.2004.09.004. 2

[Dey06] DEY T. K.: *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2006. 2, 3

[Dip21] Dipole reference implementation. github.com/galmetzer/dipole-normal-prop, 2021. Accessed: 2025-05-29. 15

[DMSL11] DIGNE J., MOREL J., SOUZANI C., LARTIGUE C.: Scale space meshing of raw data point sets. *Comput. Graph. Forum 30*, 6 (2011), 1630–1642. doi:10.1111/J.1467-8659.2011.01848.X. 3, 8

[DS83] DEMBO R. S., STEIHAUG T.: Truncated-Newton algorithms for large-scale unconstrained optimization. *Mathematical Programming 26* (1983), 190–212. doi:10.1007/bf02592055. 7

[EPF14] EPFL GEOMETRIC COMPUTING LABORATORY: EPFL statue model repository, 2014. URL: https://lgg.epfl.ch/statues.php. 9, 10, 13

[GCA13] GIRAUDOT S., COHEN-STEINER D., ALLIEZ P.: Noise-adaptive shape reconstruction from raw point sets. *Comput. Graph. Forum 32*, 5 (2013), 229–238. doi:10.1111/CGF.12189. 3

[GG07] GUENNEBAUD G., GROSS M. H.: Algebraic point set surfaces. *ACM Trans. Graph. 26*, 3 (2007), 23. doi:10.1145/1276377.1276406. 2

[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. 2, 8

[GKOM18] GUERRERO P., KLEIMAN Y., OVSJANIKOV M., MITRA N. J.: Pcpnet learning local shape properties from raw point clouds. *Comput. Graph. Forum 37*, 2 (2018), 75–85. doi:10.1111/CGF.13343. 3

[HCJ19] HUANG Z., CARR N., JU T.: Variational implicit point set surfaces. *ACM Trans. Graph. 38*, 4 (July 2019). doi:10.1145/3306346.3322994. 3, 8

[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* (New York, NY, USA, 1992), SIGGRAPH '92, ACM, pp. 71–78. doi:10.1145/133994.134011. 2

[HLZ*09] HUANG H., LI D., ZHANG H., ASCHER U. M., COHEN-OR D.: Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph. 28*, 5 (2009), 176. doi:10.1145/1618452.1618522. 2

[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 (2022), 128:1–128:13. doi:10.1145/3528223.3530096. 3, 8, 15

[iPS22] ipsr reference implementation. github.com/houfei0801/ipsr, 2022. Accessed: 2025-05-29. 15

[JBG19] JAKOB J., BUCHENAU C., GUTHE M.: Parallel globally consistent normal orientation of raw unorganized point clouds. *Comput. Graph. Forum 38*, 5 (2019), 163–173. doi:10.1111/CGF.13797. 2

[Joh07] JOHNSON S. G.: The NLopt nonlinear-optimization package. https://github.com/stevengj/nlopt, 2007. 7

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70. 2, 3

[KG09] KÖNIG S., GUMHOLD S.: Consistent propagation of normal orientations in point clouds. In *14th International Workshop on Vision, Modeling, and Visualization, VMV 2009, November 16-18, 2009, Braunschweig, Germany* (2009), Magnor M. A., Rosenhahn B., Theisel H., (Eds.), DNB, pp. 83–92. 2

[KH13] KAZHDAN M., HOPPE H.: Screened poisson surface reconstruction. *ACM Trans. Graph. 32*, 3 (jul 2013). doi:10.1145/2487228.2487237. 3

[KLAK23] KOHLBRENNER M., LEE S., ALEXA M., KAZHDAN M.: Poisson manifold reconstruction — beyond co-dimension one. *Computer Graphics Forum 42*, 5 (2023). 2, 5, 6

[LCL*23] LACHAUD J.-O., COEURJOLLY D., LABART C., ROMON P., THIBERT B.: Lightweight curvature estimation on point clouds with randomized corrected curvature measures. *Computer Graphics Forum 42*, 5 (2023), e14910. doi:https://doi.org/10.1111/cgf.14910. 2

[LFS*23] LI Q., FENG H., SHI K., FANG Y., LIU Y., HAN Z.: Neural gradient learning and optimization for oriented point normal estimation. In *SIGGRAPH Asia 2023 Conference Papers, SA 2023, Sydney, NSW, Australia, December 12-15, 2023* (2023), Kim J., Lin M. C., Bickel B., (Eds.), ACM, pp. 122:1–122:9. doi:10.1145/3610548.3618253. 3

[LN89] LIU D. C., NOCEDAL J.: On the limited memory BFGS method for large scale optimization. *Mathematical Programming 45* (1989), 503–528. doi:10.1007/bf01589116. 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. 3, 8

[LSW19] LU W., SHI Z., WANG B.: Surface reconstruction based on the modified gauss formula. *ACM Trans. Graph. 38*, 1 (2019), 2:1–2:18. doi:10.1145/3233984. 3

[LWZ*24] LIU W., WANG X., ZHAO H., XUE X., WU Z., LU X., HE Y.: Consistent point orientation for manifold surfaces via boundary integration. In *ACM SIGGRAPH 2024 Conference Papers* (New York, NY, USA, 2024), SIGGRAPH '24, Association for Computing Machinery. URL: https://doi.org/10.1145/3641519.3657475, doi:10.1145/3641519.3657475. 3

[LXSW23] 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 (2023), 14:1–14:19. doi:10.1145/3554730. 3, 8, 15

[MdGD*10] MULLEN P., DE GOES F., DESBRUN M., COHEN-STEINER D., ALLIEZ P.: Signing the unsigned: Robust surface reconstruction from raw pointsets. *Comput. Graph. Forum 29*, 5 (2010), 1733–1741. doi:10.1111/J.1467-8659.2010.01782.X. 3, 4, 8

[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 (2021), 165:1–165:14. doi:10.1145/3450626.3459835. 2, 15

[MN03] MITRA N. J., NGUYEN A. T.: Estimating surface normals in noisy point cloud data. In *Proceedings of the 19th ACM Symposium on Computational Geometry, San Diego, CA, USA, June 8-10, 2003* (2003), Fortune S., (Ed.), ACM, pp. 322–328. doi:10.1145/777792.777840. 2

[PGR22] Pgr reference implementation. github.com/jsnln/ParametricGaussRecon, 2022. Accessed: 2025-05-29. 15

[PZ07] PALACIOS J., ZHANG E.: Rotational symmetry field design on surfaces. *ACM Trans. Graph. 26*, 3 (jul 2007), 55–es. doi:10.1145/1276377.1276446. 4

[Rus20] RUSINKIEWICZ S.: trimesh2. https://gfx.cs.princeton.edu/proj/trimesh2/, 2020. 6

[SBY11] SEVERSKY L. M., BERGER M. S., YIN L.: Harmonic point cloud orientation. *Comput. Graph. 35*, 3 (2011), 492–499. doi:10.1016/J.CAG.2011.03.012. 2

[SMB*20] SITZMANN V., MARTEL J. N. P., BERGMAN A. W., LINDELL D. B., WETZSTEIN G.: Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (2020), Larochelle H., Ranzato M., Hadsell R., Balcan M., Lin H., (Eds.). 3

[SSG17] SCHERTLER N., SAVCHYNSKYY B., GUMHOLD S.: Towards globally optimal normal orientations for large point clouds. *Comput. Graph. Forum 36*, 1 (2017), 197–208. doi:10.1111/CGF.12795. 2

[Sta96] Stanford 3d scanning repository. https://graphics.stanford.edu/data/3Dscanrep/, 1996. 13

[Sva02] SVANBERG K.: A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization 12* (2002), 555–573. doi:10.1137/s1052623499362822. 7

[WZX*23] WANG Z., ZHANG Y., XU R., ZHANG F., WANG P., CHEN S., XIN S., WANG W., TU C.: Neural-singular-hessian: Implicit neural representation of unoriented point clouds by enforcing singular hessian. *ACM Trans. Graph. 42*, 6 (2023), 274:1–274:14. doi:10.1145/3618311. 3, 8

[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. doi:10.1145/3592129. 3, 8

[XWH*03] XIE H., WANG J., HUA J., QIN H., KAUFMAN A.: Piecewise C^1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression. In *IEEE Visualization, 2003. VIS 2003.* (2003), IEEE, pp. 91–98. 2

[ZJ16] ZHOU Q., JACOBSON A.: Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797* (2016). 13

Appendix: Data

We use the dataset from [BLN*13], as well as a selection of publicly available test models including from the Stanford 3D Scanning Repository [Sta96] (ARMADILLO and BUNNY in figs. 1, 6, 7 and 9), aim@shape (COW, RAPTOR, and BIMBA in figs. 5, 7, 10 and 13), EPFL [EPF14] (figs. 12 and 15), and Thingi10k [ZJ16] (METATRON [id54725], PENTAGONALICOSITETRAHEDRON [id50931], CELLULARTHING [id61258], and PENTAGONALHEXECONTAHEDRON [id509320] in fig. 10).

For the comparisons in figs. 9 and 10, the point samples are either taken from the repositories of the reference implementations of

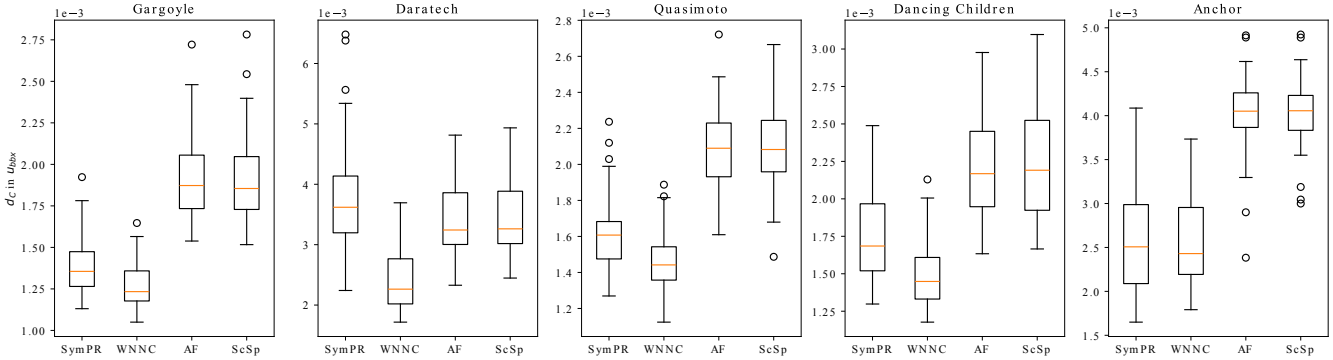

Figure 16: *Mean Chamfer distance d_C from and to the ground truth mesh. We show boxplots of the values on the* 48 *point sets of each of the* 5 *meshes in the [BLN*13] benchmark. All values are given in units of bounding box diagonals U_{bbx} of the respective ground truth mesh.*

		d_C			d_{RMSE}			ΔCC			ΔG		
Model	Method	min	μ	max	min	μ	max	min	μ	max	min	μ	max
ANCHOR	AF	0.26	0.44	0.54	0.53	1.08	1.25	-4	-0.40	0	-5	0.44	4
	SymPR	0.18	0.29	0.45	0.35	0.63	1.03	-30	-3.29	0	-28	-4.67	0
	ScSp	0.33	0.44	0.54	0.76	1.08	1.25	-4	-0.27	0	-4	0.83	4
	WNNC	0.20	0.28	0.41	0.49	0.69	0.96	-3131	-872.83	-63	-363	-63.79	-7
DARATECH	AF	0.24	0.36	0.51	0.39	0.54	0.68	-6	-1.54	0	-91	-26.21	2
	SymPR	0.24	0.40	0.68	0.37	0.77	1.35	-52	-11.56	-1	-114	-36.67	-11
	ScSp	0.26	0.36	0.52	0.41	0.54	0.69	-4	-1.15	0	-74	-24.62	4
	WNNC	0.18	0.26	0.39	0.23	0.44	0.78	-1547	-480.50	-12	-117	-41.92	-8
DANCING	AF	0.17	0.23	0.30	0.27	0.40	0.56	-2	-0.46	0	-13	-3.90	2
CHILDREN	SymPR	0.13	0.18	0.25	0.18	0.26	0.42	-2	-0.33	0	-11	-1.15	0
	ScSp	0.17	0.23	0.32	0.27	0.40	0.58	-2	-0.27	0	-10	-3.81	2
	WNNC	0.12	0.15	0.22	0.16	0.22	0.41	-439	-122.42	-7	-34	-5.10	0
GARGOYLE	AF	0.18	0.22	0.31	0.33	0.42	0.60	-1	-0.08	0	-4	-0.62	0
	SymPR	0.13	0.16	0.22	0.17	0.22	0.34	-3	-0.58	0	-10	-1.17	0
	ScSp	0.17	0.22	0.32	0.32	0.42	0.62	-1	-0.12	0	-3	-0.48	0
	WNNC	0.12	0.15	0.19	0.16	0.21	0.32	-484	-237.58	-10	-57	-8.23	0
QUASIMOTO	AF	0.15	0.19	0.25	0.25	0.36	0.46	-2	-0.19	0	-1	-0.19	0
	SymPR	0.12	0.15	0.20	0.15	0.20	0.32	-2	-0.21	0	-11	-0.71	0
	ScSp	0.14	0.19	0.24	0.21	0.36	0.46	-2	-0.29	0	-3	-0.29	0
	WNNC	0.10	0.13	0.17	0.14	0.18	0.25	-172	-52.06	0	-17	-2.25	0

Table 3: *Reconstruction accuracy for different methods on the [BLN*13] dataset. We calculate the Chamfer distance d_C and RMSE d_{RMSE} as the mean value from and to the ground truth and measure the defect in the number of connected components ΔCC and genus ΔG with respect to the ground truth. For all metrics, we report the mean (μ) value and extreme values (*min, max*) over the* 48 *different samplings of the* 5 *models in the dataset.*

	Point Set	N	Origin
Fig. 9	ARMADILLO	40000	PGR
	BUNNY	20000	PGR
	HAND	10000	Dipole
	FANDISK	11031	Dipole
	VASE	24545	Dipole
	WOODFISH	12887	iPSR
Fig. 10	Raptor	100000	*Uniform*
	LION	749999	Dipole
	FLOWER	100000	Dipole
	GALERA	100000	Dipole
	METATRON	69170	iPSR
	PENTAGONAL ICOSITETRAHEDRON	82371	iPSR
	PENTAGONAL HEXECONTAHEDRON	1000000	*Uniform*
	CELLULARTHING	1000000	*Uniform*

Table 4: *Size N and source of the point sets in figs. 9 and 10.*

PGR [HWW*22, PGR22], Dipole Propagation [MHZ*21, Dip21] and iPSR [LXSW23, iPS22], or are obtained through uniform sampling of the the surface meshes. Details are summarized in table 4.

Appendix: Pseudocode

We provide pseudocode for computing the implicit function from a set of unoriented samples. We recall that our goal is to solve for the function $\tilde{\chi}$ minimizing the energy:

$$E(\tilde{\chi}) = \|\nabla\tilde{\chi}\odot\nabla\tilde{\chi} - T\|^2 + \alpha\sum_p \tilde{\chi}^2(p)$$

where $T:\Omega\to\mathbb{R}^{3\times3}$ is the symmetric matrix field defined by the samples' unsigned normals.

We discretize the system using first-order B-splines (piecewise tri-linear "hat" functions) centered at the corners of an $N\times N\times N$ grid. Letting $\mathcal{K}=\{0,\dots,N\}\times\{0,\dots,N\}\times\{0,\dots,N\}$ index the corners of the grid, and letting $\{\Phi_i\}_{i\in\mathcal{K}}$ be the set of B-splines centered at the corners of the grid, we represent the solution as $\tilde{\chi}=\sum_{i\in\mathcal{K}} x_i\cdot\Phi_i$ and seek the coefficients $\mathbf{x}\in\mathbb{R}^{|\mathcal{K}|}$ minimizing:

$$\begin{aligned} E(\boldsymbol{x}) &= \left\|\nabla\left(\sum_{i\in\mathcal{K}} x_i\cdot\Phi_i\right)\odot\nabla\left(\sum_{i\in\mathcal{K}} x_i\cdot\Phi_i\right) - T\right\|^2 \\ &\qquad + \alpha\sum_p\left(\sum_{i\in\mathcal{K}} x_i\cdot\Phi_i(p)\right)^2 \\ &= \sum_{i,j,k,l\in\mathcal{K}} S_{ijkl}\cdot x_i\cdot x_j\cdot x_k\cdot x_l + \sum_{i,j\in\mathcal{K}}(-2B_{ij}+\alpha M_{ij})\cdot x_i\cdot x_j + c \end{aligned} \tag{12}$$

where $S\in\mathbb{R}^{|\mathcal{K}|\times|\mathcal{K}|\times|\mathcal{K}|\times|\mathcal{K}|}$ is a 4-tensor giving the integrals of the dot-products of the outer-products of gradients of pairs of basis functions, $B\in\mathbb{R}^{|\mathcal{K}|\times|\mathcal{K}|}$ is a 2-tensor giving the integrals of the dot-products of the target matrix field with the outer-products of gradients of pairs of basis functions, $M\in\mathbb{R}^{|\mathcal{K}|\times|\mathcal{K}|}$ is a 2-tensor giving the sums over the samples of the products of the values of pairs of basis functions, and c is a constant (whose value does not affect the optimization):

$$S_{ijkl} = \langle\!\langle\nabla\Phi_i\odot\nabla\Phi_j, \nabla\Phi_k\odot\nabla\Phi_l\rangle\!\rangle,$$
$$B_{ij} = \langle\!\langle\nabla\Phi_i\odot\nabla\Phi_j, T\rangle\!\rangle, \qquad M_{ij} = \sum_p \Phi_i(p)\cdot\Phi_j(p).$$

We begin by describing the construction of the system. Then, we describe the single-level implementation of iterative relaxation. Finally, we describe the hierarchical solver.

Constructing the System

We assume that we are given a set of point-samples $\mathcal{P}=\{(p,C)\}$ with a sample's position in the unit cube, $p\in\Omega$, and symmetric matrix $C\in\mathbb{R}^{3\times3}$ obtained as the outer product of the sample's unsigned unit normal. When the input is a raw point cloud, unsigned normals are estimated by finding the $K=20$ nearest-neighbors, computing the covariance matrix, and using the (unsigned) direction of least covariance.

Constructing T (alg. 1) We define the target symmetric matrix field to be a piecewise tri-linear function given by per-corner matrix coefficients $\boldsymbol{t}\in(\mathbb{R}^{3\times3})^{|\mathcal{K}|}$:

$$T(q) \equiv \sum_{i\in\mathcal{K}} t_i\cdot\Phi_i(q).$$

We set the coefficients by iterating over the samples (line 1), computing the position of the sample in grid-coordinates (line 2), identifying the corner indices within a 2-ring neighborhood (line 3), and splatting the sample's matrix into the corners using a cubic B-spline kernel, $K:\mathbb{R}^3\to\mathbb{R}$, (line 4).

Algorithm 1: MatrixFieldCoefficients

Require: $\mathcal{P}\subset\Omega\times\mathbb{R}^{3\times3}$
1: **for** $(p,C)\in\mathcal{P}$:
2: $\quad\tilde{p}=p\cdot N$
3: $\quad$**for** $i\in\mathcal{K}$ s.t. $\|i-\tilde{p}\|_\infty\le2$:
4: $\quad\quad t_i \mathrel{+}= K(i-\tilde{p})\cdot C$
5: **return** $\boldsymbol{t}$

Constructing B (alg. 2) Given the target matrix field, T, we compute its dualized representation as a matrix $B\in\mathbb{R}^{|\mathcal{K}|\times|\mathcal{K}|}$. Our approach uses finite-element assembly, where we iterate over the cells of the grid and compute the contribution of each cell to every pair of functions supported on that cell. (Given a cell index $c\in\{0,\dots,N-1\}\times\{0,\dots,N-1\}\times\{0,\dots,N-1\}$, we denote by $\mathrm{Cell}(c)$ the grid cell indexed by c.)

We first iterate over all cells (line 1) and compute the cell center (line 2). Then, for every pair of functions supported on the cell (lines 3 and 4), we compute the integral of the dot-product of the outer-product of the functions' gradients with the matrix field inside the cell and add the integrated dot-product to the associated matrix coefficient (line 5). The scaling by $1/N$ in line 5 accounts for the fact that in scaling from grid-coordinates to unit-cube-coordinates, the volume of a grid cell scales by $1/N^3$ while the two gradients' magnitudes scale by N.

Algorithm 2: MatrixFieldDual

Require: $T : \Omega \to \mathbb{R}^{3\times 3}$
1: **for** $c \in \{0,\dots,N-1\}\times\{0,\dots,N-1\}\times\{0,\dots,N-1\}$:
2: $q = c + (0.5, 0.5, 0.5)$
3: **for** $i \in \mathcal{K}$ s.t. $\|i-q\|_\infty \leq 1$:
4: **for** $j \in \mathcal{K}$ s.t. $\|j-q\|_\infty \leq 1$:
5: B_{ij} += $\frac{1}{N}\int_{\text{Cell}(c)} \langle \nabla\Phi_i \odot \nabla\Phi_j, T\rangle$
6: **return** B

We note that the matrix $B \in \mathbb{R}^{|\mathcal{K}|\times|\mathcal{K}|}$ is sparse as B_{ij} is non-zero only if the functions Φ_i and Φ_j have overlapping support.

Computing M (alg. 3) For value interpolation, we define a screening matrix $M \in \mathbb{R}^{|\mathcal{K}|\times|\mathcal{K}|}$ giving the sum of the products of the values of the basis functions, summed over the input samples.

We iterate over all samples (line 1), compute the grid-coordinates of the sample (line 2), identify all pairs of functions supported on the sample (lines 3 and 4), and add the product of the evaluation of the functions at the sample to the associated matrix entry (line 5).

Algorithm 3: ScreeningMatrix

Require: $\mathcal{P} \subset \Omega \times \mathbb{R}^{3\times 3}$
1: **for** $(p, C) \in \mathcal{P}$:
2: $\tilde{p} = p \cdot N$
3: **for** $i \in \mathcal{K}$ s.t. $\|i-\tilde{p}\|_\infty \leq 1$:
4: **for** $j \in \mathcal{K}$ s.t. $\|j-\tilde{p}\|_\infty \leq 1$:
5: M_{ij} += $\Phi_i(\tilde{p}) \cdot \Phi_j(\tilde{p})$
6: **return** M

As with the matrix field dual, the screening matrix has non-zero entries only for pairs of functions with overlapping support.

Computing S The computation of the entries of S is straightforward as it is input-independent and we omit the details. However, we note that 4-tensor is sparse (with S_{ijkl} non-zero only for 4-tuples $\{i,j,k,l\}$ such that the intersection of the supports of Φ_i, Φ_j, Φ_k, and Φ_l are non-trivial). Furthermore, away from the unit-cube boundary, the entries of S are shift-invariant (so that the tensor can be compactly represented using a fixed-sized stencil).

Single-Level Relaxation (alg. 4)

Given an estimate of the solution coefficients $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{K}|}$, and the matrices $B, M \in \mathbb{R}^{|\mathcal{K}|\times|\mathcal{K}|}$ we optimize eq. (12) by repeatedly iterating over the coefficients of $\boldsymbol{x}$, fixing all but the current coefficient, computing the associated quartic polynomial, and solving for the value minimizing the quartic.

For each relaxation pass (line 1), we iterate over the corners (line 2), compute the quartic polynomial defined by keeping all but the i-th coefficient fixed (line 3), differentiate the quartic (line 4), compute the roots of the derivative (line 5), and set the i-th coefficient to the root minimizing the energy (line 6).

Algorithm 4: SingleLevelRelaxation

Require: iters $\in \mathbb{Z}^{\geq 0}$, $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{K}|}$, $B, M \in \mathbb{R}^{|\mathcal{K}|\times|\mathcal{K}|}$
1: **for** $\iota \in \{1,\dots,\text{iters}\}$:
2: **for** $i \in \mathcal{K}$:
3: $P_4(s) = \text{QuarticEnergy}(i, \boldsymbol{x}, B, M)$
4: $P_3(s) = \frac{d}{ds} P_4(s)$
5: $\Upsilon = \text{Roots}(P_3)$
6: $x_i = \text{argmin}_{r\in\Upsilon} P_4(r)$

Quartic Polynomial Computation (alg. 5) Fixing all but the i-th coefficient of $\boldsymbol{x}$, we obtain a quartic polynomial describing the energy. To compute (all but the constant term of) the polynomial, we define functions:

- $\text{Overlap}(j_1,\dots,j_n)$ taking a set of corner indices and returning a boolean indicating if the intersection of the supports of the associated functions is non-empty:

$$\text{Overlap}(j_1,\dots,j_n) \equiv \left(\bigcap_{j\in\{j_1,\dots,j_n\}} \text{Supp}(\Phi_j)\right) \neq \emptyset.$$

- $\mu(i, \{j_1,\dots,j_n\})$ giving the number of times corner index i appears in the set of corner indices $\{j_1,\dots,j_n\}$:

$$\mu(i,\{j_1,\dots,j_n\}) \equiv \sum_{k=1}^{n} \delta_{i j_k}.$$

- $\boldsymbol{x}(i,j)$ returning the j-th coefficient of $\boldsymbol{x}$ when corner indices i and j are different, and 1 when they are equal:

$$\boldsymbol{x}(i,j) \equiv \begin{cases} x_j & \text{if } i \neq j \\ 1 & \text{otherwise} \end{cases}$$

First we iterate over all 4-tuples of corner indices, at least one of which is i and whose associated functions have overlapping supports (line 1), determine the degree of the coefficient the 4-tuple contributes to by counting the number of indices equal to i (line 2), and accumulate the contribution to the associated polynomial coefficient (line 3). Next we iterate over all pairs of indices, at least one of which is i and whose associated functions have overlapping support (line 4), determine the degree of the coefficient the pair contributes to (line 5), and accumulate the contribution to the associated polynomial coefficient (line 6).

Algorithm 5: QuarticEnergy

Require: $i \in \mathcal{K}$, $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{K}|}$, $B, M \in \mathbb{R}^{|\mathcal{K}|\times|\mathcal{K}|}$
1: **for** $j,k,l,m \in \mathcal{K}$ s.t. $i \in \{j,k,l,m\}$ and $\text{Overlap}(j,k,l,m)$:
2: $d = \mu(i, \{j,k,l,m\})$
3: c_d += $S_{jklm} \cdot \boldsymbol{x}(i,j) \cdot \boldsymbol{x}(i,k) \cdot \boldsymbol{x}(i,l) \cdot \boldsymbol{x}(i,m)$
4: **for** $j,k \in \mathcal{K}$ s.t. $i \in \{j,k\}$ and $\text{Overlap}(j,k)$:
5: $d = \mu(i, \{j,k\})$
6: c_d += $(-2B_{jk} + \alpha M_{jk}) \cdot \boldsymbol{x}(i,j) \cdot \boldsymbol{x}(i,k)$
7: **return** $c_4 \cdot s^4 + c_3 \cdot s^3 + c_2 \cdot s^2 + c_1 \cdot s$

Hierarchical Relaxation (alg. 6)

To solve the system hierarchically, we first perform a fine-to-coarse pass in which we restrict the constraints and then perform a coarse-to-fine pass in which we relax and prolong the solution coefficients.

We use superscript ℓ to denote variables associated with hierarchy level $\bar{L} \leq \ell \leq L$, we set $P^\ell \in \mathbb{R}^{|\mathcal{K}^{\ell+1}|\times|\mathcal{K}^\ell|}$ to be the standard prolongation matrix for trivariate first-order B-splines:

$$P^\ell_{ij} \equiv \begin{cases} \left(\frac{1}{2}\right)^{\|i-2\cdot j\|^2} & \text{if } \|i-2\cdot j\|_\infty \leq 1 \\ 0 & \text{otherwise} \end{cases},$$

and we denote by $R^\ell \equiv (P^\ell)^\top$ its transpose.

In the restriction pass, we iterate over the hierarchy levels from finest to coarsest (line 1) and restrict both the matrix field dual and the screening matrix (lines 2 and 3). Next, we initialize the solution at the coarsest level (line 4) and perform a large number of relaxations (line 5). Finally, we proceed to the prolongation phase, iterating over the hierarchy levels from coarsest to finest (line 6), prolonging the solution from the previous coarser level (line 7) and further relaxing the solution at the current level (line 8).

Algorithm 6: HierarchicalSolver

Require: $B^L, M^L \in \mathbb{R}^{|\mathcal{K}^L|\times|\mathcal{K}^L|}$
1: **for** $\ell \in \{L-1,\dots\bar{L}\}$:
2: $\quad B^\ell = R^\ell \cdot B^{\ell+1} \cdot P^\ell$
3: $\quad M^\ell = R^\ell \cdot M^{\ell+1} \cdot P^\ell$
4: $\boldsymbol{x}^{\bar{L}} = \text{InitSolution}(\bar{L})$
5: SingleLevelRelaxation($\overline{\text{iters}}, x^{\bar{L}}, B^{\bar{L}}, M^{\bar{L}}$)
6: **for** $\ell \in \{\bar{L}+1,\dots,L\}$:
7: $\quad \boldsymbol{x}^\ell = P^{\ell-1} \cdot \boldsymbol{x}^{\ell-1}$
8: $\quad$ SingleLevelRelaxation(iters, x^ℓ, B^ℓ, M^ℓ)
9: **return** $\boldsymbol{x}^L$

Coarse Solution Initialization (alg. 7) To set the initial guess for the solution at the coarsest level, we compute the grid center (line 1), iterate over all the corners (line 2), and set the associated coefficient to the squared distance of the corner from the center (line 3).

Algorithm 7: InitSolution

Require: $\ell \in \mathbb{Z}^{\geq 0}$
1: $c = 2^\ell \cdot (0.5, 0.5, 0.5)$
2: **for** $i \in \mathcal{K}$:
3: $\quad x_i = \|i-c\|^2$
4: **return** $\boldsymbol{x}$