Variational Surface Reconstruction Using Natural Neighbors

JIANJUN XIA and TAO JU, Washington University in St. Louis, USA

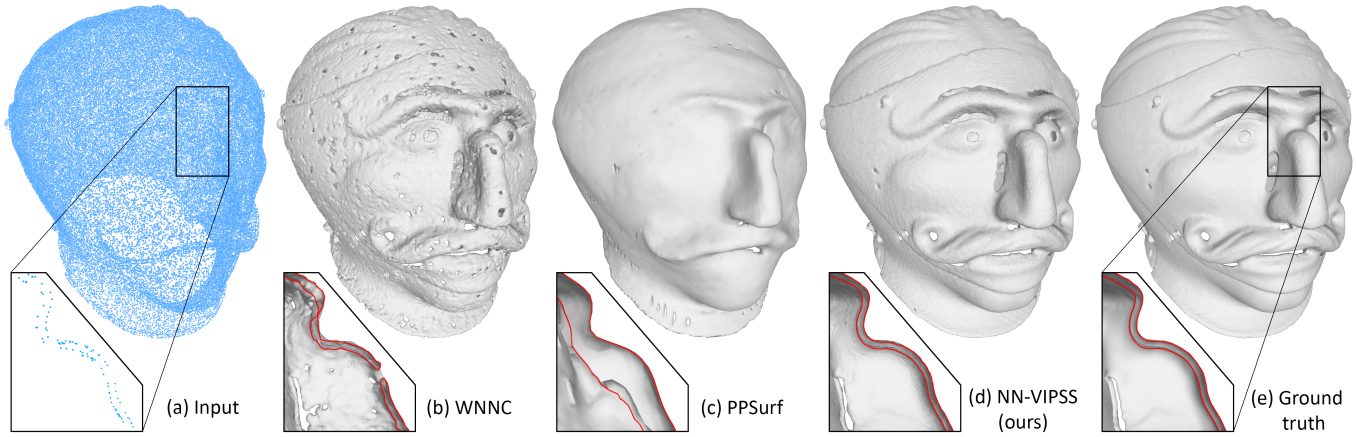


Fig. 1. Given a point cloud *without* normals (a) that sparsely samples a thin Helmet (from [Laric 2012], 100K samples), the state-of-the-art normal estimation method [Lin et al. 2024] results in a "holy" helmet (b), while the latest learning-based reconstruction method [Erler et al. 2024] smears the facial features (c). Our method (NN-VIPSS) closely approximates the ground truth (e). Close-up views show a cross-section of the surface.

Surface reconstruction from points is a fundamental problem in computer graphics. While numerous methods have been proposed, it remains challenging to reconstruct from sparse and non-uniform point distributions, particularly when normals are absent. We present a robust and scalable method for reconstructing an implicit surface from points without normals. By exploring the locality of natural neighborhoods, we propose local reformulations of a previous global method, known for its ability to surface sparse points but high computational cost, thereby significantly improving its scalability while retaining its robustness. Experiments show that our method achieves comparable speed to existing reconstruction methods on large inputs while producing fewer artifacts in under-sampled regions.

CCS Concepts: • **Computing methodologies** → **Point-based models**; **Volumetric models**.

Additional Key Words and Phrases: Surface reconstruction, implicit surfaces, Hermite interpolation, radial basis functions, natural neighbor interpolation

ACM Reference Format:
Jianjun Xia and Tao Ju. 2025. Variational Surface Reconstruction Using Natural Neighbors. *ACM Trans. Graph.* 44, 4 (August 2025), 19 pages. https://doi.org/10.1145/3731191

1 Introduction

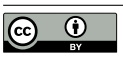
Reconstructing 3D surfaces from scattered points is a fundamental problem in computer graphics and has been extensively studied in the last few decades [Berger et al. 2017]. A common approach is to formulate the reconstruction problem as finding an implicit function whose zero level set represents the surface. The implicit approach enjoys several benefits, such as flexibility in dealing with arbitrary surface topology, ensuring a smooth and manifold output, and convenience for volumetric operations such as offsets and booleans.

Traditional implicit reconstruction methods, such as the popular Screened Poisson Reconstruction (SPR) [Kazhdan and Hoppe 2013], require normal vectors in addition to point locations. However, such information may not always be available or accurate, in which case it is necessary to estimate normals prior to reconstruction. While many methods exist for normal estimation, the problem remains challenging when the density of points is too low relative to the size of shape features (e.g., Figure 1 (a)). Incorrect normals, in turn, lead to surface artifacts (e.g., Figure 1 (b)). While data-driven methods for implicit reconstruction have shown great promise, they struggle with capturing fine features from sparse samples (e.g., Figure 1 (c)). We review these methods in more details in Section 2.

We present a new implicit reconstruction method that does not require normals and can more robustly deal with sparse and non-uniform point distributions. Our method builds on the *Variational Implicit Point Set Surface* (VIPSS) method of Huang et al. [2019], which produces a smooth signed-distance-like function via global optimization. Specifically, it solves for the Hermite data (a scalar and an unit vector) at each input point such that a smooth function interpolating the data at all points (the *Duchon's interpolant* [Duchon 1977]) minimizes a second-order distortion measure (the *Duchon's energy* [Duchon 1977]). Thanks to the global nature of both the interpolant and energy, VIPSS excels at surfacing very sparse points. Unfortunately, the method has a high complexity (cubic in the input size) and is impractical for inputs beyond a few thousand points.

In this paper, we propose a variant of VIPSS that can scale to hundreds of thousands of points. Our idea is to reformulate both the

Authors' Contact Information: Jianjun Xia, jianjun.x@wustl.edu; Tao Ju, taoju@wustl.edu, Washington University in St. Louis, St. Louis, MO, USA.



ACM 1557-7368/2025/8-ART
https://doi.org/10.1145/3731191

interpolant and energy so that they can be locally computed. Specifically, we approximate the global Duchon's interpolant by blending local interpolants, each defined over the *natural neighbors* of a point – edge neighbors in the Delaunay Triangulation. By adopting *Natural Neighbor Coordinates* (NNC) [Sibson 1980] as blending weights, we show that our blended interpolant, called *Natural-Neighbor-Duchon* (NND) interpolant, retains the key strengths of Duchon's interpolant, including interpolation, smoothness, and robustness against sparse and non-uniform point distributions. The blended form gives rise to a localized energy definition in lieu of Duchon's energy, which in turn leads to a new variational problem that can be locally constructed and solved.

Our method, called *Natural-Neighbor-VIPSS* (NN-VIPSS), retains several features of VIPSS. It involves a single parameter that needs to be tuned (λ), which balances fitting accuracy with surface smoothness. In particular, the implicit surface exactly interpolates all points at $\lambda = 0$. Except for the surfacing step, our method does not involve any spatial discretization (such as octrees). Furthermore, our method reproduces linear geometry and commutes with similarity transformations. Our experiments showed that NN-VIPSS performs as robustly as VIPSS on small-sized, sparse or non-uniform point sets, but with significantly improved speed and scalable to much larger inputs (we tested up to one million points). Compared to existing reconstruction methods, NN-VIPSS achieves comparable speed but shows greater robustness in under-sampled regions (e.g., Figure 1 (d)).

The rest of the paper is organized as follows. After reviewing related works on surface reconstruction (Section 2), we recall concepts and methods that our method builds on (Section 3). The technical discussion starts with a motivation and overview of our method (Section 4), followed by details on our proposed interpolant (Section 5) and variational method (Section 6), and ends with a discussion on asymptotic complexity (Section 7). Finally, we present the experimental results (Section 8) and discuss limitations (Section 9).

2 Related Works

2.1 Scattered data interpolation

Closely related to surface reconstruction, scattered data interpolation is the problem of constructing a function (interpolant) that interpolates given data (values, derivatives, etc.) at scattered points in space. There is a rich literature on the subject; see for example [Anjyo et al. 2014; Wendland 2004]. We briefly review two techniques that are most related to our work.

2.1.1 Natural Neighbor Interpolation. One way to interpolate scattered data is by taking weighted sum of data associated with input points near the query location. A popular choice of such weights it the Natural Neighbor Coordinates (NNC) introduced by Sibson [Sibson 1980], which are defined using the Voronoi Diagram of the input and query points (see details in Section 3.2). The NNC-weighted sum of scalar data has several desirable properties as an interpolant, such as exact interpolation, reproducing constant and linear functions, being smooth (except at the input points), and local computations (due to the locality of natural neighbors). To improve the smoothness of NNC at input points, various higher-order extensions of NNC have been proposed, such as the C^1 coordinates of Sibson [Sibson 1981] and Farin [Farin 1990] and the C^2 coordinates of Hiyoshi [Hiyoshi and Sugihara 2004]. These coordinates allow interpolation of not only values, but also derivatives at input points. We refer readers to [Bobach 2009] for an excellent survey on the topic of natural-neighbor-based interpolation.

Instead of taking weighted sum of data, we blend local functions using NNC as blending weights. By choosing suitable local functions, our interpolant achieves smooth interpolation of Hermite data without using higher-order coordinates. Besides, we observed that our blended Hermite interpolant produces less undulations than interpolation using higher-order variants of NNC (see Figure 5).

2.1.2 Radial Basis Functions (RBF). An RBF interpolant consists of weighted sum of radial kernels centered at the points, where the weights (called *coefficients*) are found by solving a linear system defined by the given scalar data. RBF can be generalized to interpolate Hermite data, known as Hermite RBF (or HRBF), to which Duchon's interpolant belong (see details in Section 3.3). We refer readers to books [Buhmann 2003; Wendland 2004] for in-depth discussions on theoretical and computational aspects of RBF.

With globally supported kernels, (H)RBF can be computationally expensive to solve and evaluate when the point number is large. While compactly supported kernels (e.g., Wendland's functions [Wendland 1995]) can reduce complexity, they are not suited for interpolating sparse samples due to the limited support (e.g., Figure 5 (f)). The Fast Multipole Method (FMM) accelerates evaluations of globally supported kernels using far-field expansions, and it can achieve impressive speed-up in asymptotic complexity [Greengard and Rokhlin 1987]. However, the practical overhead of FMM can be significant for large inputs. For example, [Carr et al. 2001] reports over three hours for solving RBF at 80K points, and more recently [Zhong et al. 2020] reports more than 6 minutes to solve HRBF at 60K points. In contrast, our interpolant takes just a few minutes to solve for a million points (see Figure 16 and Table 5).

Another way to speed up RBF is to use *partition-of-unity* [Franke and Nielson 1980]. This approach uses a set of overlapping subdomains and defines an RBF interpolant over points inside each subdomain. These local interpolants are then blended using smooth weights that vanish outside each subdomain and have unit sum. Existing methods typically use disks or boxes as subdomains [Franke 1982; Wendland 2004], constructed such that each subdomain contains a bounded number of points. However, similar to the use of kernels with compact support, the use of fixed-size subdomains reduces the ability to interpolate data across large gaps.

We follow the same partition-of-unity approach to approximate Duchon's interpolant but replace the fixed-size subdomains with the natural neighborhood around each point. This choice is motivated in part by the piecewise structure of Duchon's interpolant in 1D and in part by the adaptivity of natural neighborhoods to point distributions. Using NNC as blending weights, our blended interpolant was found to closely approximate Duchon's interpolant, even for sparse and non-uniform data (Figures 5 and 7), while maintaining locality in computation.

2.2 Implicit surface reconstruction

We briefly review methods that, like ours, reconstruct a surface as the zero level set of an implicit function. Besides the implicit approach, the surface can be reconstructed using explicit approaches, such as Delaunay triangulation [Amenta et al. 2001; Bernardini et al. 1999; Dey and Goswami 2003] and mesh deformation [Hanocka et al. 2020]. We refer readers to surveys [Berger et al. 2017; Huang et al. 2024] for comprehensive discussions on surface reconstruction.

2.2.1 Points with normals. Given normal vectors associated with points, many methods aim at reconstructing a signed-distance-like function whose gradient at each point aligns with the given normal. The partition-of-unity approach was used to blend linear functions [Boissonnat and Cazals 2002; Hoppe et al. 1992] and polynomials [Ohtake et al. 2003a] defined in local neighborhoods. Moving Least Squares (MLS) solves for a local polynomial *per* spatial location that fits nearby samples and normals [Dey and Sun 2005; Guennebaud and Gross 2007; Kolluri 2008; Öztireli et al. 2009; Shen et al. 2004]. Both RBF and HRBF have been used, with the former applied to offset points in the normal directions with signed values [Carr et al. 2001; Dinh et al. 2002; Morse et al. 2001; Ohtake et al. 2003b; Samozino et al. 2006; Turk and O'Brien 2002; Walder et al. 2007] and the latter interpolating normals vectors and zero values [Brazil et al. 2010; Ijiri et al. 2013; Liu et al. 2016].

Another class of methods reconstruct an "indicator function", which is 1 (resp. 0) in the interior (resp. exterior) of the shape. The Poisson Reconstruction method [Kazhdan et al. 2006] and its variants [Kazhdan and Hoppe 2013; Manson et al. 2008; Pan and Skala 2012; Taubin 2012] solve the Poisson equation guided by a smooth vector field obtained from the normals. Methods like [Barill et al. 2018; Lu et al. 2018] reconstruct the electric potential field defined by *dipoles*, also known as *generalized winding numbers*. While being efficient and producing smooth surfaces, these methods are not suited for downstream tasks that require distances and gradients away from the surfaces (e.g., computing offsets or volume rendering). In addition, we found that these methods may not handle sparse inputs or missing points as robustly as global interpolation methods, such as HRBF (see Figure 8).

2.2.2 Points without normals. One way to deal with the lack of normals is to first compute an unsigned distance field and then *sign* the field afterwards [Giraudot et al. 2013; Hornung and Kobbelt 2006; Mullen et al. 2010; Poranne et al. 2010]. However, inferring an accurate unsigned distance function is a challenging problem itself, and a dense sampling is often required. Additionally, the two-step approach increases the complexity of implementation as well as the number of parameters.

Other methods, like ours, compute the implicit function in a single step by solving a variational problem [Alliez et al. 2007; Lu et al. 2005; Schölkopf et al. 2004; Walder et al. 2005; Zhao et al. 2001]. The optimization objective includes fitting accuracy and various regularization terms to encourage smooth surfaces while avoiding the trivial (constant zero) solution. Some of them require domain discretization [Alliez et al. 2007; Lu et al. 2005; Zhao et al. 2001] or numerical integration [Walder et al. 2005], and all involve multiple parameters to tune. In contrast, the variational method of [Huang et al. 2019] involves a single parameter that controls the amount of approximation (zero for exact interpolation), requires no discretization or integration, and was shown to be more robust in surfacing sparse points than previous variational methods. Its main drawback, which we address in this paper, is its prohibitive computational cost (see details in Section 3.3).

Implicit functions, such as occupancy function or signed distance functions (SDFs), can also be obtained directly from point clouds using data-driven methods [Chen and Zhang 2019; Mescheder et al. 2019]. In particular, recent methods such as [Erler et al. 2024, 2020] improve the generalizability of previous approaches by combining learned priors at both global and local levels. However, handling sparse samples and capturing fine features remain challenging for learning-based methods, as we show in Figures 1 (d) and 15.

2.2.3 Normal estimation. Given a point cloud, normal vectors can be estimated in two phases, a local phase that computes an un-oriented line direction at each point based on its local neighborhood, and a global phase that obtains a consistent orientation among all points. The un-oriented directions can be obtained by locally fitting planes [Hoppe et al. 1992] or polynomials [Cazals and Pouget 2003; Guennebaud and Gross 2007], or by analyzing the shape of Voronoi cells [Alliez et al. 2007; Merigot et al. 2011]. While robustness can be improved for noisy samples [Mitra et al. 2004] and around sharp features [Boulch and Marlet 2012; Li et al. 2010] by choosing suitable local neighborhoods, obtaining correct directions on sparse samples remains difficult. Approaches for the global phase include computing the Minimum Spanning Tree [Hoppe et al. 1992], iterative propagation [Metzer et al. 2021], and global optimization [Ma et al. 2024; Schertler et al. 2017; Xiao et al. 2023]. As errors in un-oriented directions generally cannot be corrected by flipping the orientations, handling sparse samples is inherently challenging for the two-phase approach.

Other methods estimate oriented normal vectors in a single phase via global optimization [Hou et al. 2022; Lin et al. 2024, 2022; Wang et al. 2011; Xu et al. 2023]. Many of these methods simultaneously solve for the normals and an implicit function, and their objectives are based on characteristics of an indicator function, such as a fixed value (0.5) at each point [Lin et al. 2024, 2022], bimodal distribution of values (around 0 and 1) elsewhere [Xu et al. 2023], and normal-aligned gradients at input points [Hou et al. 2022; Lin et al. 2024]. These variational methods show greatly improved robustness over two-phase approaches. However, we found in our experiments that these methods still struggle in sparsely sampled regions (e.g., Figure 1 (b) and more in Section 8).

3 Preliminaries

To make the paper self-contained, we briefly review the fundamental concepts and detail two prior methods that our work builds upon, namely Sibson's Natural Neighbor Coordinates [Sibson 1980] (Section 3.2) and the variational reconstruction method, VIPSS [Huang et al. 2019] (Section 3.3).

3.1 Voronoi Diagram, Delaunay Triangulations, and natural neighbors

Given a finite set of points $X = \{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$, the *Voronoi cell* of each x_i consists of all locations that are no further from x_i than from any other $x_j \in X$ ($j \neq i$). Each Voronoi cell is a convex polyhedron, and the union of all Voronoi cells and their boundary facets is the *Voronoi Diagram* (VD) of X. The dual of VD of X is the *Delaunay Triangulation* (DT) of X, which consists of d-dimensional simplices (e.g., edges, triangles and tetrahedra for $d = 1, 2, 3$) that form a disjoint partitioning of the convex hull of X. Each k-dimensional element of DT corresponds to a $(d-k)$-dimensional element of VD.

The *natural neighbors* of $x_i \in X$ is the set of points $x_j \in X$ whose Voronoi cells share common facets with the Voronoi cell of x_i. Equivalently, each such x_j is connected to x_i by a DT edge. Intuitively, x_j is a natural neighbor of x_i if and only if there exists a d-sphere with x_i, x_j on the sphere and no other point of X inside. Figure 2 shows the VD (a) and DT (b) of a set of points in 2D, highlighting a point (green) and its natural neighbors (red).

3.2 Natural Neighbor Coordinates

Natural Neighbor Coordinates (NNC) were introduced by Sibson [Sibson 1980] to smoothly and locally interpolate scattered (scalar) data. Given scalar values s_i associated with each point $x_i \in X$, a scalar function can be defined as a weighted sum $f(x) = \sum_{i=1}^{n} w_i(x) s_i$. For a point x inside the convex hull of X, Sibson defines the weight function $w_i(x)$ as the volume fraction of the Voronoi cell of x in the VD of the union set $X \cup \{x\}$ that comes from the Voronoi cell of x_i in the VD of X. That is,

$$w_i(x) = \frac{vol(V_X(x_i) \cap V_{X\cup\{x\}}(x))}{vol(V_{X\cup\{x\}}(x))} \tag{1}$$

where $V_X(x_i)$ denotes the Voronoi cell of x_i in the VD of X. In the illustration in Figure 2 (c), the intersection region in the numerator of Equation 1 is shaded.. NNC have several properties that make them ideal for scattered data interpolation, including partition of unity (i.e., $\sum_{i=1}^{n} w_i(x) = 1$), Lagrange property (i.e., $w_i(x_i) = 1$ and $w_j(x_i) = 0$ for any $j \neq i$), linear reproduction (i.e., $\sum_{i=1}^{n} w_i(x) x_i = x$), C^{d-1} continuity except at the points X (where they are C^0), and local support.

The Voronoi cell $V_{X\cup\{x\}}(x)$ has an infinite volume when x lies outside the convex hull of X, where NNC become undefined. To extend NNC beyond the convex hull, a simple strategy is to add *ghost points* to expand the convex hull [Bobach 2009]. These ghost points, denoted by Y and located on a convex shape that encloses X, are not associated with any Hermite data and hence have zero coordinates. To maintain a partition of unity, NNC can be modified so that the denominator in Equation 1 only considers the contributions from Voronoi cells of the input points $x_i \in X$:

$$w_i(x) = \frac{vol(V_{X'}(x_i) \cap V_{X'\cup\{x\}}(x))}{\sum_i vol(V_{X'}(x_i) \cap V_{X'\cup\{x\}}(x))}, \tag{2}$$

where $X' = X \cup Y$. It can be verified that the modification retains all properties of NNC in the larger convex hull of Y, except linear reproduction, which is lost when x's natural neighbors include ghost points. In this paper, we call the modified weights in Equation 2 the *extended NNC*.

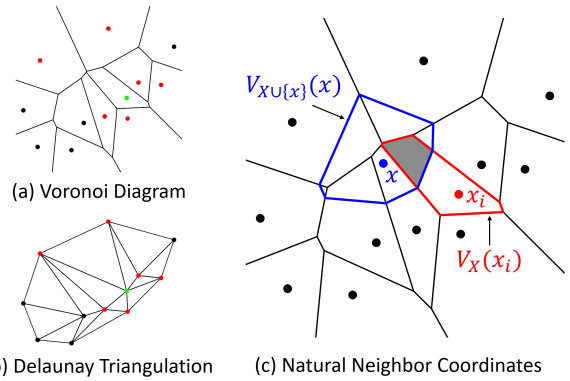


Fig. 2. The Voronoi Diagram (a) and Delaunary Triangulation (b) of 2D points, highlighting a point (green) and its natural neighbors (red), and an illustration for Natural Neighbor Coordinates (c).

3.3 VIPSS

The idea behind [Huang et al. 2019] is to find a smooth, signed-distance-like implicit function whose zero level set approximates the input points X. By representing the function as a Hermite interpolant, the problem can be formulated as seeking the Hermite data consisting of scalars $S = \{s_1, \ldots, s_n\}$ and unit-length vectors $G = \{g_1, \ldots, g_n\}$ that minimizes the squared scalars (fitting term) and non-smoothness of the interpolant (regularization term)[1]:

$$\begin{aligned} &\text{Minimizes:} && S^T S + \lambda\, E(f_{S,G}) \\ &\text{Subject to:} && {g_i}^T g_i = 1, \quad \forall i = 1, \ldots, n \end{aligned} \tag{3}$$

Here, $f_{S,G}$ is a function that interpolates $\{S, G\}$ (i.e., values and gradients of $f_{S,G}$ are $\{s_i, g_i\}$ at each x_i), E is some energy that measures the non-smoothness of a function, and λ is the balancing weight. Once the solution Hermite data $\{S^*, G^*\}$ is found, its interpolant f_{S^*,G^*} is the reconstructed implicit function.

The two key ingredients in this formulation are the Hermite interpolant $f_{S,G}$ and the energy E. For the former, Huang et al. [2019] adopts the *Duchon's interpolant* [Duchon 1977], which excels at smoothly interpolating sparse samples. The interpolant is represented as HRBF with a globally supported kernel,

$$f_{S,G}(\mathbf{x}) = \sum_{i=1}^{n} a_i \phi(x, x_i) + \sum_{i=1}^{n} b_i^T D^{0,1} \phi(x, x_i) + p^T X + q \tag{4}$$

where $\phi(x, y) = \|x - y\|^3$ is the triharmonic kernel, D is the derivative operator ($D^{i,j}\phi(x, y)$ takes the i-th and j-th partial derivatives of ϕ w.r.t. x and y), and coefficients $a_i \in \mathbb{R}, b_i \in \mathbb{R}^d, p \in \mathbb{R}^d, q \in \mathbb{R}$ are determined by Hermite data $\{S, G\}$ via a system of linear equations that enforces interpolation ($f_{S,G}(x_i) = s_i, Df_{S,G}(x_i) = g_i$ for all i) and additional constraints necessary for the existence of a

[1] All vectors in this paper are assumed to be column vectors.

unique solution ($\sum_i a_i = 0$ and $\sum_i a_i x_i + \sum_i b_i = 0$):

$$M\begin{pmatrix} A \\ B \\ p \\ q \end{pmatrix} = \begin{pmatrix} S \\ G \\ 0 \\ 0 \end{pmatrix} \tag{5}$$

Here, $A = \{a_1, \ldots, a_n\}$, B is the flattened array of $\{b_1, \ldots, b_n\}$, and G is the flattened array of $\{g_1, \ldots, g_n\}$. The matrix M, a symmetric matrix of length $(d+1)(n+1)$, is known as the *interpolation matrix*.

For the energy E, Huang et al. adopts the *Duchon's energy* [Duchon 1977], a 2nd-order integral measure that generalizes the 1-dimensional thin plate energy. Given Hermite data $\{S, G\}$, Duchon's energy is minimized by the Duchon's interpolant defined above. Furthermore, the minimal energy has a simple closed form:

$$E(f_{S,G}) = \begin{pmatrix} S^T & G^T \end{pmatrix} J \begin{pmatrix} S \\ G \end{pmatrix}, \tag{6}$$

where J is the top-left block of length $(d+1)n$ in M^{-1}. This simplifies the objective in Equation 3 to a quadratic function of $\{S, G\}$,

$$S^T S + \lambda \begin{pmatrix} S^T & G^T \end{pmatrix} J \begin{pmatrix} S \\ G \end{pmatrix} \tag{7}$$

The global nature of both Duchon's interpolant and Duchon's energy allows VIPSS to robustly surface sparse and non-uniformly distributed points. However, it also leads to a high computational cost. In particular, constructing the matrix J in Equation 7 involves inverting the interpolation matrix M, a dense matrix of size $O(n^2)$, which takes $O(n^3)$ time. The optimization solver takes $O(n^2)$ time in each iteration to compute the objective and gradient. Quadratic complexity also applies to the construction of the Duchon's interpolant from the solved Hermite data, which involves solving Equation 5 by multiplying the rhs with the already computed M^{-1}. The high complexity limits the practical use of VIPSS to just a few thousand points, beyond which it quickly becomes too time-consuming.

4 Motivation and overview

To scale up variational reconstruction, our idea is to replace the global interpolant and energy in the variational formulation of VIPSS (Equation 3) with ones that can be locally computed, without compromising its ability to handle sparse and non-uniform point distributions.

Our local formulations are motivated by observing the piecewise structure in Duchon's interpolant in one dimension ($d = 1$), which leads to significantly lower computational complexity. In this case, Duchon's interpolant is the *piecewise cubic Hermite Spline*, where each piece is a cubic function interpolating the values and derivatives between two adjacent points – precisely Duchon's interpolant over those two points (see Figure 3 (a)). Constructing the (global) interpolant over all points therefore amounts to constructing the $n-1$ (local) two-point interpolants, which takes linear time instead of quadratic for the global interpolant.

Furthermore, Duchon's energy of the global interpolant can also be replaced by the *sum* of energy of local, two-point interpolants. This is because Duchon's interpolant in 1D is only non-linear within the interval spanned by the input points, and the intervals between successive points form a disjoint partition of the interval spanned by all points. The summative form of energy takes linear time to construct or evaluate, in contrast to the cubic or quadratic complexity to perform the same tasks using the original Duchon's energy.

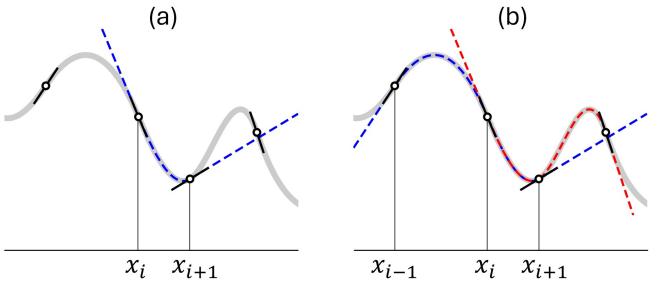


Fig. 3. Duchon's interpolant in 1D (gray curve) of values and derivatives (black circles and lines) overlaid with Duchon's interpolants of a pair (a) and two triples (b) of consecutive data points.

Unfortunately, Duchon's interpolant in higher dimensions no longer possesses a simple piecewise structure. To this end, we introduce a new Hermite interpolant that closely approximates Duchon's interpolant but can be locally computed (Section 5). The new interpolant has a semi-piecewise structure that gives rises to a summative energy definition, which enables local computations for variational reconstruction (Section 6). We end the technical discussion by a complexity analysis of our method (Section 7).

5 Hermite interpolation

Given points $X = \{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$ associated with scalars $S = \{s_1, \ldots, s_n\}$ and vectors $G = \{g_1, \ldots, g_n\}$, we seek a smooth Hermite interpolant whose value and gradient at each point x_i coincide with s_i and g_i. The energy of this interpolant will be used as a regularizer in our variational problem (see next section), and the interpolant of the optimized Hermite data will be the reconstruction output.

Ideally, the Hermite interpolant should robustly handle sparse and non-uniform samples, like Duchon's interpolant, but can be locally constructed. Our definition is motivated by the piecewise structure of Duchon's interpolant in 1D. There, the global Duchon's interpolant over all points is made up of local Duchon's interpolants over pairs of consecutive points. Moving to higher dimensions, we define local Duchon's interpolants over *subsets* of the input points (Section 5.1). These local interpolants are then *blended* in a partition-of-unity approach to be our final interpolant (Section 5.2).

5.1 Defining local interpolants

The building blocks of our interpolant are Duchon's interpolants over subsets $X_i \subseteq X$. Our choice of the subsets is guided by two considerations. On one hand, to achieve locality, the size of each subset should be as small as possible. On the other hand, to enable blending, the space spanned by each subset (i.e., its "support") should be large enough to cover the domain with sufficient overlaps between neighboring subsets.

As a motivating example, let's consider the one-dimensional case again. As discussed previously, Duchon's interpolant defined over

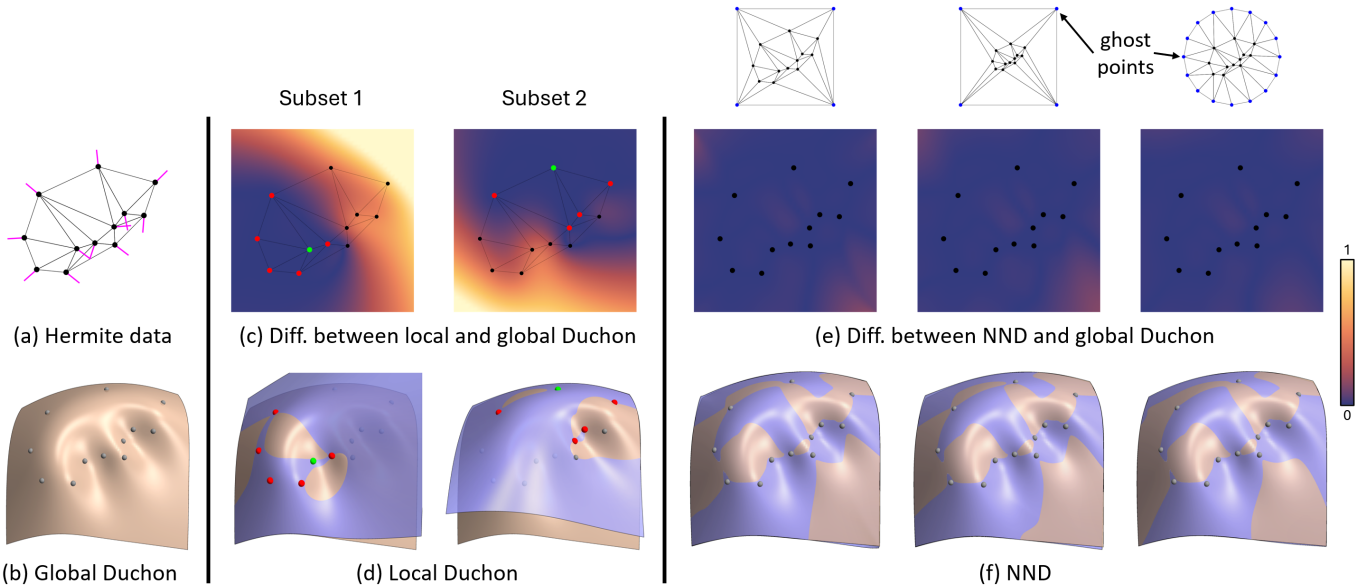


Fig. 4. Left: input 2D Hermite data (a) and the global Duchon's interpolant as a height surface (b). Middle: local Duchon's interpolants over two subsets as (light blue) height surfaces (d) and their differences to the global Duchon's interpolant (e). Right: NND using three ghost point configurations (top).

two consecutive points $\{x_i, x_{i+1}\}$ exactly matches Duchon's interpolant over all points in the interval $[x_i, x_{i+1}]$. However, these intervals are disjoint from each other. To enable overlaps, we can define each subset X_i to include a point x_i and *its left and right neighbors*, x_{i-1} (if $i > 1$) and x_{i+1} (if $i < n$). Duchon's interpolant over each such X_i coincides with the global interpolant in the extended interval $[x_{i-1}, x_{i+1}]$ (i.e., the support of X_i), and the supports of X_i and X_{i+1} overlap in $[x_i, x_{i+1}]$ (see Figure 3 (b)).

A natural generalization of the subsets above to $d > 1$ dimensions is to have each X_i consist of x_i and its *natural neighbors* (reviewed in Section 3.1). Note that, for $d = 1$, the natural neighbors are precisely the left and right neighbors (if they exist). For $d > 1$, the natural-neighbor-based subsets offer an overlapping coverage of the domain, regardless of the point distributions. Specifically, if we consider the Delaunay Triangulation (DT) simplicies (e.g., edges, triangles, or tetrahedra for $d = 1, 2, 3$) containing x_i as the support of X_i, the union of supports of all subsets covers the convex hull of X, and the supports of subsets centered at two points connected by a DT edge overlap in DT simplicies containing that edge. Compared to fixed-size neighborhood definitions, such as points within a fixed distance or k-nearest neighbors, the size (both in dimension and cardinality) of a natural neighborhood varies with the local distribution of points to ensure a well-covered domain. The adaptability makes natural neighbors ideal for defining subsets used in blending, particularly on non-uniform samples.

Empirically, we observed that Duchon's interpolant over a subset defined using natural neighbors locally resembles Duchon's interpolant over all points. This is demonstrated in 2D in Figure 4. The input in (a) consists of planar points associated with zero scalars (i.e., $s_i \equiv 0$) and unit-length vectors, indicated by the magenta lines. The local Duchon's interpolants over two subsets, each consisting of an input point (green) and its natural neighbors (red), are visualized as height surfaces (light blue) in (d) and by their differences to the global Duchon's interpolant over all points in (c). Note that the local and global interpolants closely overlap near the respective subsets. The resemblance makes these local interpolants suited for blending.

5.2 Blending interpolants

We follow the partition-of-unity strategy [Franke 1982] to construct our Hermite interpolant by blending the Duchon's interpolants over all natural-neighbor-based subsets,

$$\tilde{f}_{S,G}(x) = \sum_{i=1}^{n} w_i(x) f_{S_i,G_i}(x), \tag{8}$$

where $\{S_i, G_i\}$ are the Hermite data associated with subset X_i and $w_i(x)$ is the blending weight of x w.r.t. X_i.

A good candidate of the blending weights is the Natural Neighbor Coordinates (NNC) [Sibson 1980]. As reviewed in Section 3.2, NNC enjoys an array of properties suited for scattered data interpolation, such as partition-of-unity, Lagrange property, smoothness (away from X), locality, and linear reproduction. While NNC is only defined within the convex hull of X, the restriction can be relaxed by introducing ghost points outside X and using the extended NNC (Equation 2). Accordingly, we modify the subsets X_i to include those points in X (not ghost points) that are natural neighbors of x_i in the union set of X and the ghost points.

The key limitation of both NNC and extended NNC is that they are not smooth at the input points X. This is problematic for reconstructing a smooth surface that interpolates X. Also, unlike NNC, the extended NNC does not warrant linear reproduction. Fortunately, these limitations disappear when we combine extended NNC with Duchon's interpolants over natural neighborhoods. As we show

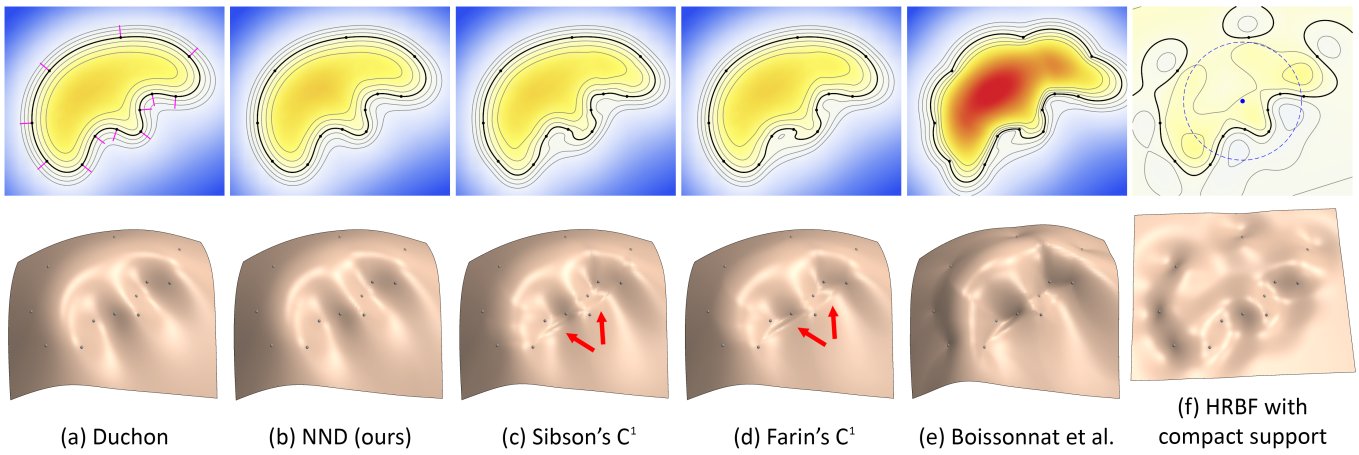


Fig. 5. Comparing 2D Hermite interpolation using Duchon's interpolant (a), our NND (b), Sibson's C^1 coordinates [Sibson 1981] (c), Farin's C^1 coordinates [Farin 1990] (d), method of [Boissonnat and Cazals 2002] (e), and HRBF with a compactly supported kernel (f). Each function is visualized by colors with contours (top, thick curve is zero contour) and the height surface (bottom).

below, the blended interpolant $\tilde{f}_{S,G}(x)$ is C^1 continuous everywhere and reproduces linear functions (see proof in Appendix A):

Proposition 5.1. *The following properties hold for $\tilde{f}_{S,G}(x)$ in Equation 8, given $w_i(x)$ defined by Equation 2, ghost points Y, and x inside the convex hull of Y:*

(1) ***Interpolation**: $\tilde{f}_{S,G}(x_i) = s_i, D\tilde{f}_{S,G}(x_i) = g_i$ for all $x_i \in X$.*
(2) ***Smoothness**: $D\tilde{f}_{S,G}(x)$ exists for any x.*
(3) ***Linear reproduction**: If these is a linear function f such that $s_i = f(x_i)$ and $g_i = Df(x_i)$ for every $i = 1, \ldots, n$, then $\tilde{f}_{S,G} = f$.*

We call the blended interpolant $\tilde{f}_{S,G}$ (Equation 8), with our natural-neighbor-based choices of subsets and blending weights, the *Natural-Neighbor Duchon's interpolant* (or NND). Although lacking theoretical analysis, we found in our experiments that NND closely approximates Duchon's interpolant. Furthermore, the choice of the ghost points seems to have little impact on the interpolant. This is demonstrated in Figure 4, where we explored three choices of ghost point configurations, shown at the top of (e), with varying numbers of ghost points (blue) and their distances to the input points (black). Observe from the difference maps (e) and height surfaces (f) that NND remains a close approximation of Duchon's interpolant in each configuration.

Compared with existing, locally-computable Hermite interpolants, NND appears to produce smoother-looking interpolations that more closely resemble Duchon's interpolant. As shown in Figure 5, interpolations using higher-order extensions of NNC, such as the C^1 coordinates of Sibson [1981] and Farin [1990], exhibit notable undulations as indicated in (c,d). Similarly to our approach, Boissonnat and Cazals [2002] use NNC as weights to blend local Duchon's interpolants over subsets. However, each of their subset X_i consists of a single point x_i, and their method inherits the non-smoothness of NNC at each input point, as seen in (e). Lastly, methods using fixed-size neighborhoods, such as HRBF with compactly supported kernels, may have difficulty connecting distant samples, as shown in (f).

Here, we use a compactly supported kernel $\phi(x, y) = \psi(\|x - y\|/r)$, where ψ is the Wendland's function ($\psi(d) = (1-d)^4(1+4d)$ if $d \le 1$ and $\psi(d) = 0$ otherwise) and the kernel size r is the radius of the blue dotted circle drawn at the top of (f).

6 Variational reconstruction

The local Hermite interpolant, NND, possesses a *semi-piecewise* structure as it blends together (overlapping) pieces, each being Duchon's interpolant f_{S_i,G_i} over a subset X_i. The structure suggests a simple definition of its energy as the sum of Duchon's energies of all pieces,

$$\tilde{E}(\tilde{f}_{S,G}) = \sum_{i=1}^{n} E(f_{S_i,G_i}), \tag{9}$$

Using the closed-form of Duchon's energy (Equation 6), we obtain a local, natural-neighborhood-based reformulation of the variational problem in VIPSS (Equation 3) as

$$\begin{aligned} \text{Minimizes:} \quad & S^T S + \lambda \textstyle\sum_{i=1}^{n} \begin{pmatrix} S_i^T & G_i^T \end{pmatrix} J_i \begin{pmatrix} S_i \\ G_i \end{pmatrix} \\ \text{Subject to:} \quad & {g_i}^T g_i = 1, \quad \forall i = 1, \ldots, n. \end{aligned} \tag{10}$$

Here, J_i is the top-left block of length $(d+1)|X_i|$ in the inverse of the interpolation matrix of the subset X_i. We call the zero level set of $\tilde{f}_{S^*,G^*}$, where $\{S^*, G^*\}$ is the solution to Equation 10, the *Natural-Neighbor VIPSS* (or NN-VIPSS).

We next show that NN-VIPSS inherits the same properties of VIPSS that are desirable for surface reconstruction, including exact interpolation, linear reproduction, and commutativity with similarity transformations. We will then discuss our strategy for solving the minimization problem in Equation 10, including a local method for initialization.

6.1 Properties

We first show that NN-VIPSS exactly interpolates all input points when $\lambda = 0$. In this case, the second term in the objective of Equation 10 vanishes, and $S = 0$ becomes the trivial minimizer. To optimize the vectors G at $\lambda = 0$, observe that the objective becomes $\lambda \sum_{i=1}^{n} G_i^T J_i' G_i$ when $S = 0$, where J_i' is the lower-right block of J_i of length $d|X_i|$. Hence the minimizer G of Equation 10 as $\lambda \to 0$ is the solution to the following reduced (and parameter-free) problem:

$$\begin{aligned} &\text{Minimizes:} && \textstyle\sum_{i=1}^{n} G_i^T J_i' G_i \\ &\text{Subject to:} && g_i{}^T g_i = 1, \quad \forall i = 1, \ldots, n \end{aligned} \tag{11}$$

Second, NN-VIPSS reproduces linear geometry. That is, if all points lie on a $(d-1)$-dimensional hyperplane L in $\mathbb{R}^d$, NN-VIPSS coincides with L. Since Duchon's interpolant reproduces linear functions, the local interpolants $f_{S_i^*, G_i^*}$ that interpolate the Hermite data $\{S^*, G^*\}$, where $S^* = 0$ and G^* assumes the unit normal of L, are the same linear function f whose zero level set is L. As linear functions have zero Duchon's energy, $\{S^*, G^*\}$ is the solution to Equation 10 with zero objective, and NND interpolant $\tilde{f}_{S^*, G^*}$ recovers f due to its linear reproduction property.

Lastly, NN-VIPSS is invariant to isometric transformations (translations and rotations). This is due to the isometry-invariance of both Duchon's energy and the extended NNC. Furthermore, like VIPSS, NN-VIPSS has the following commutativity with uniform scaling: the NN-VIPSS of points uniformly scaled by some factor σ is the same as the σ-scaled NN-VIPSS of the original points, after multiplying λ by σ^3. This can be stated formally as (see proof in Appendix B),

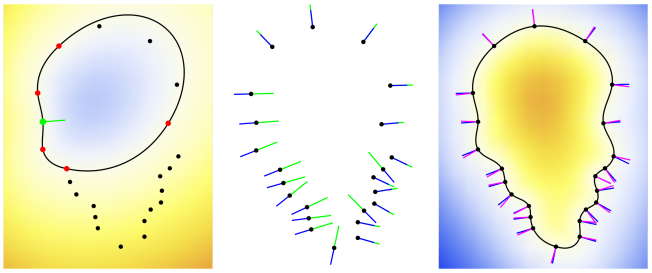
Proposition 6.1. *Let $X, \hat{X}$ be two point sets such that $\hat{X} = \sigma X$ for some $\sigma > 0$. If Hermite data $\{S^*, G^*\}$ associated with X is a solution of Equation 10 w.r.t. some $\lambda \geq 0$, then Hermite data $\{\hat{S}^* = \sigma S^*, \hat{G}^* = G^*\}$ associated with $\hat{X}$ is a solution w.r.t. $\hat{\lambda} = \sigma^3 \lambda$. Furthermore, interpolant $\tilde{f}_{\hat{S}^*, \hat{G}^*}$ over $\hat{X}$ is related to $\tilde{f}_{S^*, G^*}$ over X by $\tilde{f}_{\hat{S}^*, \hat{G}^*} = \sigma \tilde{f}_{S^*, G^*}(x/\sigma)$.*

6.2 Optimization

The constrained quadratic optimization problem of Equation 10 can be computationally challenging to solve. Huang et al. [2019] optimizes VIPSS by representing each vector g_i as two spherical angles in some chosen coordinate system, thus eliminating the unit vector constraints. We found that their approach can be sensitive to the choice of the coordinate system and often results in high-energy local minima (see Figure 10). Instead, we replace the hard constraints by adding a soft penalty term measuring the deviation of g_i from a unit vector:

$$S^T S + \lambda \sum_{i=1}^{n} \begin{pmatrix} S_i^T & G_i^T \end{pmatrix} J_i \begin{pmatrix} S_i \\ G_i \end{pmatrix} + \lambda\alpha \sum_{i=1}^{n} (g_i^T g_i - 1)^2 \tag{12}$$

The choice of weight α trades off smoothness of the energy landscape (smoother for smaller α) with proximity of solution to NN-VIPSS (closer for bigger α). It can be verified that the modified objective retains the properties discussed in Section 6.1. In particular, interpolation is ensured at $\lambda = 0$, where the objective becomes $\sum_{i=1}^{n} G_i^T J_i' G_i + \alpha \sum_{i=1}^{n} (g_i^T g_i - 1)^2$, and commutativity with scaling holds after multiplying α by $1/\sigma$ where σ is the scaling factor. We minimize this quartic objective using the L-BFGS method [Liu and Nocedal 1989] (implemented in NLopt [Johnson 2007]) and normalize the solved g_i afterward.



(a) Initial Hermite (b) MST orientation (c) Optimization

Fig. 6. Optimization process ($\lambda = 0$). (a): Initial normal (green) at each point (green) computed using its natural neighbors (red). (b): Normals (blue) oriented using MST. (c): Optimized normals (magenta).

To initialize the optimization, Huang et al. relaxes the unit-vector constraint on each g_i to be one on all vectors (i.e., $G^T G = 1$). This transforms the constrained minimization of the objective in Equation 7 to finding the eigenvector of a matrix derived from J, which is a dense matrix of length $O(n)$. As a result, the initialization step has a high complexity of $O(n^3)$. Motivated by the summative form of our new objective (Equation 10), we adopt a more efficient, local approach for initialization.

First, we compute a scalar and vector at each point x_i by performing variational reconstruction *locally* on the subset X_i (Figure 6 (a)). While we could employ VIPSS, it would involve a non-linear optimization at each input point. Since we are only interested in the Hermite variables at the point x_i, we can formulate a reduced minimization problem where the unit-vector constraint is only applied to the vector at x_i and no other points in X_i. The reduced problem can be solved directly by computing the eigenvector of a $d \times d$ matrix (see details in Appendix C).

Next, given the resulting Hermite pair $\{s_i', g_i'\}$ at each x_i, we compute a consistent orientation of all g_i' using the Minimum Spanning Tree (MST) method of [Hoppe et al. 1992] on the Delaunay graph of X (Figure 6 (b)). Specifically, we assign a weight to each Delaunay edge between x_i, x_j as $|g_i' \cdot g_j'|$ and compute an MST of the weighted graph. We then root the MST at any point, fix an arbitrary normal orientation at the root, and traverse the MST from root to leaves while flipping the normal at a node if it forms an abtuse angle with the normal at its parent node (s_i' is negated if g_i' is flipped).

7 Complexity analysis

Our method's computational cost is largely determined by the sizes of the natural neighborhoods, which in turn depend on the point distribution. We will first analyze complexity as a function of the number of natural neighbors of each point x_i, denoted by n_i. We will then discuss the impact of point distributions on n_i and, in turn, our method's scalability.

In the following, we assume the dimensionality d is a fixed constant, and we denote the sums $\sum_i^n n_i$, $\sum_i^n n_i^2$, $\sum_i^n n_i^3$ respectively by N_1, N_2, N_3. Computing NN-VIPSS consists of three stages:

(1) Computing Delaunay Triangulation (DT) of X: The DT and its dual Voronoi Diagram (VD) are needed to obtain natural neighbors and compute the blending weights. Note that DT and VD in 3D have a worst-case complexity of $O(n^{\lceil d/2 \rceil})$ [McMullen 1970]. The most popular algorithms use incremental point insertion [Bowyer 1981; Edelsbrunner and Shah 1994; Watson 1981]. The cost of each insertion step is generally proportional to the number of natural neighbors of the inserted point. Adding the time for point location, the (expected) complexity of this step is $\boldsymbol{O(N_1 + n \log n)}$.

(2) Optimizing Hermite data (Section 6): It takes $O(N_3)$ time to compute the matrices J_i in the objective of Equation 10 by inverting the corresponding interpolation matrices of length $O(n_i)$. Initializing the Hermite data as described in Section 6.2 takes $O(N_3)$ time to solve a local variational problem over each X_i (using the eigenvector method in Appendix C) and an additional $O(N_1 \log n)$ time to compute MST over DT edges. In total, setting up the optimization takes $\boldsymbol{O(N_3 + N_1 \log n)}$ time. Finally, each solver iteration requires $\boldsymbol{O(N_2)}$ time to evaluate residues and gradients.

(3) Interpolating Hermite data (Section 5): Given the solved Hermite data, the NND interpolant $\tilde{f}_{S,G}$ in Equation 8 is constructed by obtaining the coefficients of the local interpolants f_{S_i,G_i} via Equation 5. Using the inverse of interpolation matrices computed in the previous stage, this takes $\boldsymbol{O(N_2)}$ time. Evaluating a query point x involves locating x in the DT of X, which takes $O(\log n)$ time, and computing extended NNC $w_i(x)$ and evaluating f_{S_i,G_i} at each natural neighbor x_i of x, both taking $O(n_i)$ time. This brings the total time for one NND evaluation to $\boldsymbol{O(\sum_{i \in r_x} n_i + \log n)}$, where r_x denotes the (indices of) natural neighbors of x.

The natural neighborhood size n_i, and hence the complexity of these stages, can vary significantly with how the points are distributed. We will consider two extreme scenarios (see summary in Table 1). In the worst case, each x_i can have $O(n)$ natural neighbors. In $d = 3$ dimensions, the only known cases that this happens is when the points are sampled from 1D curves [Amenta et al. 2007]. In this case, the setup of NN-VIPSS optimization dominates the complexity with $O(N_3) = O(n^4)$ time, making our method even less scalable than VIPSS. Recall that VIPSS takes $O(n^3)$ time to set up the optimization, $O(n^2)$ time for each solver iteration, and $O(n^2)$ time to construct the Duchon's interpolant.

At the other end of the spectrum, it is well-known that the number of DT edges, which is half the sum of natural neighbors N_1, is $O(n)$ for "nicely" distributed points in a sphere [Dwyer 1989] or on a $(d-1)$-dimensional polyhedral surface [Amenta et al. 2007]. If we further assume that the distribution is uniform and hence each n_i is $O(1)$, our method only needs $O(n \log n)$ time to compute DT and set up the optimization, $O(n)$ time for each solver iteration and $O(n)$ for building the interpolant, thus significantly improving the scalability of VIPSS. In fact, our construction of the interpolant in this best-case scenario is even faster than FMM, which needs $O(n \log n)$ time to solve for the (H)RBFs coefficients [Greengard and Rokhlin 1987]. On the other hand, while FMM takes constant time to evaluate, evaluating NND needs $O(|r_x| + \log n)$ time. The number of natural neighbors of the query point x, $|r_x|$, depends on the location of x, with a worse-case of $O(n)$ and a best-case of $O(1)$ if x is drawn from the same nice distribution as X.

Table 1. Time complexity comparison in each algorithm step between VIPSS, Fast Multipole Method (only for solving and evaluating HRBFs), and NN-VIPSS (with best-case and worst-case scenarios of point distributions). DT computation in NN-VIPSS is included in Optimization Setup. Best complexity for each step is in **bold**.

		VIPSS	FMM	NN-VIPSS	
				Best-case	Worst-case
Optimization	Setup	$O(n^3)$	–	$\boldsymbol{O(n \log n)}$	$O(n^4)$
	Solve (per iter)	$O(n^2)$	–	$\boldsymbol{O(n)}$	$O(n^3)$
Interpolation	Solve	$O(n^2)$	$O(n \log n)$	$\boldsymbol{O(n)}$	$O(n^3)$
	Evaluate	$O(n)$	$\boldsymbol{O(1)}$	$O(\log n)$	$O(n^2)$

In our experiments, we found that most of the large inputs in practice (e.g., digital scans) are closer to the best-case scenario above. In particular, the number of natural neighbors n_i is typically around a few dozens and rarely goes above a hundred. As a result, our method was able to process inputs with hundreds of thousands of points within a few minutes, significantly exceeding VIPSS's limit of a few thousand points.

8 Results

8.1 Implementation details

Our method is implemented in C++ and builds on several libraries. We use Tetgen [Hang 2015] to construct the DT, from which we obtain the natural neighborhoods. We adopt the method described in [Boissonnat and Cazals 2002] for computing the Natural Neighbor Coordinates (NNC) of a query point. We use Armadillo [Sanderson and Curtin 2016] with OpenBLAS [Xianyi et al. 2012] for most of our linear algebra routines and Eigen [Guennebaud et al. 2010] with OpenMP for efficient sparse matrix operations. Optimization is performed using the L-BFGS [Liu and Nocedal 1989] implementation in NLopt [Johnson 2007] with default parameters and 10,000 maximum iterations. We use OpenMP to parallelize most of our local, per-point operations, such as constructing the matrices J_i (Equation 10), initializing the Hermite data (Appendix C), constructing local interpolants f_{S_i,G_i}, and evaluating those interpolants and blending weights at each natural neighbor of the query point.

For Hermite interpolation, we use 42 ghost points located uniformly on a sphere (via icosahedral subdivision) circumscribing the cube $[-1.5, 1.5]^3$ (we scale all inputs to be within $[-1, 1]^3$). Similar to 2D (see Figure 4 (e,f)), we found that the number and location of ghost points have little impact on our NND interpolant in 3D. The level set of NND is discretized by first computing a tetrahedral grid inside $[-1.5, 1.5]^3$ using the adaptive refinement method of [Ju et al. 2024] (with threshold $\epsilon = 0.003$ or 0.1% of the grid dimension, unless otherwise stated), followed by Marching Tetrahedra. Compared to discretization methods based on uniform grids, such as Marching Cubes, we found that adaptive grids can better capture surface details without an excessively large grid.

Tests were run on a workstation running Ubuntu 24.04 with AMD 7950x CPU (4.5Hz and 32 threads), 64GB RAM, and NVidia RTX3090 GPU with 24GB RAM. Our method runs entirely on CPU, and GPU

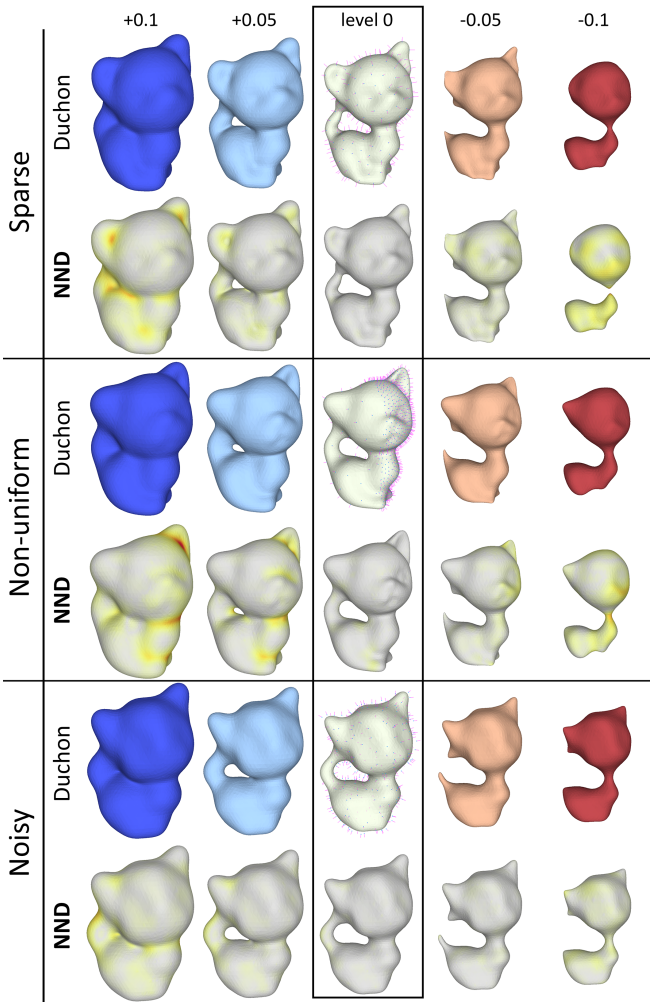


Fig. 7. Comparing the level set surfaces of NND and Duchon's interpolant on sparse (top, 250 points), non-uniform (middle), and noisy (bottom) samples of the Kitten. Input Hermite data are shown in the middle column (points as blue dots and normals as magenta lines). NND level sets are colored by distance to the corresponding level sets of Duchon's interpolant (red for distances greater than 0.04, or 2% of bounding box).

is only used for testing other methods that require it [Lin et al. 2024, 2022; Metzer et al. 2021].

8.2 Hermite interpolation

We start by evaluating our Hermite interpolant, NND (Section 5). As in 2D, we observed that NND, despite its local formulation, closely approximates the global Duchon's interpolant. Figure 7 compares both interpolants on three different samplings of the Kitten model. The input Hermite data has zero values (i.e., points are exactly interpolated) for the sparse and non-uniform samplings (top and middle rows) and positive values (i.e., points are approximated) for the noisy sampling (bottom rows). Observe that the level sets of the two interpolants agree well in each sampling.

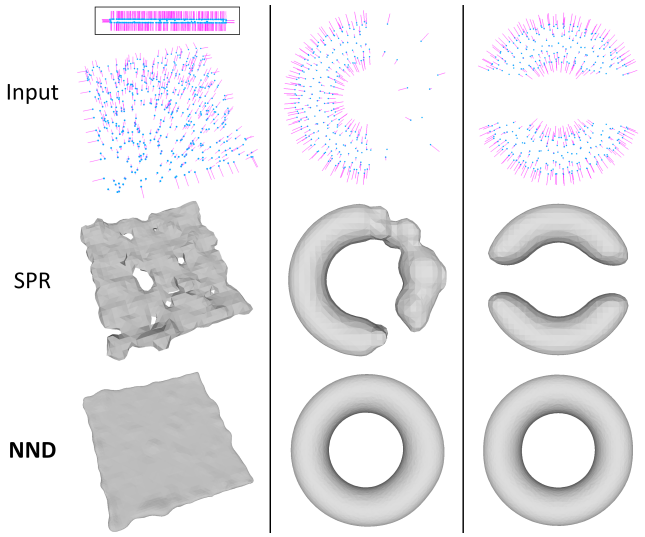


Fig. 8. Comparing SPR [Kazhdan and Hoppe 2013] (at octree level 10) and our NND interpolant on oriented points that are sparse (left), non-uniform (middle), or contain large missing regions (right). The input consists of points and normals sampled from a thin plate (left, see side view at the top) and torus (middle and right). NND assumes zero value at each point.

Like Duchon's interpolant, NND has several advantages over existing implicit reconstruction methods for points *with* normals. First, existing methods like SPR [Kazhdan and Hoppe 2013] and those based on generalized winding numbers [Barill et al. 2018; Lu et al. 2018] produce an indicator function, which is discontinuous at the zero level set and has vanishing gradient everywhere else. In contrast, NND is C^1 everywhere inside the convex hull of the ghost points, and the function gradually increases or decreases away from the zero level set, as seen in Figure 5 (b). This behavior mimics that of a signed distance field and makes NND suited for downstream tasks such as computing approximate offsets (e.g., the non-zero level sets in Figure 7), adaptive domain refinement [Ju et al. 2024], and volumetric rendering [Hart 1996]. Second, while methods like SPR excel at producing smooth surfaces given densely sampled points with normals, they are not as robust as NND in dealing with sparse, non-uniform, or missing samples, as demonstrated in Figure 8.

8.3 Variational reconstruction

We next evaluate our variational reconstruction method, NN-VIPSS (Section 6), on a variety of 3D inputs, with a focus on sparse and non-uniform point distributions.

8.3.1 Parameters. Like VIPSS, NN-VIPSS employs a parameter λ to balance the competing goals of data fitting and surface smoothness. Setting $\lambda = 0$ leads to exact interpolation of all points (Section 6.1), which is suited for noise-free inputs. Increasing λ leads to surfaces with greater deviation from the input and generally smoother reconstructions, which is more suited for inputs with noise. The effect of λ is demonstrated in Figure 9.

We demonstrate in Figure 10 the benefit of using the soft penalty term in Equation 12, weighted by parameter α, over enforcing the

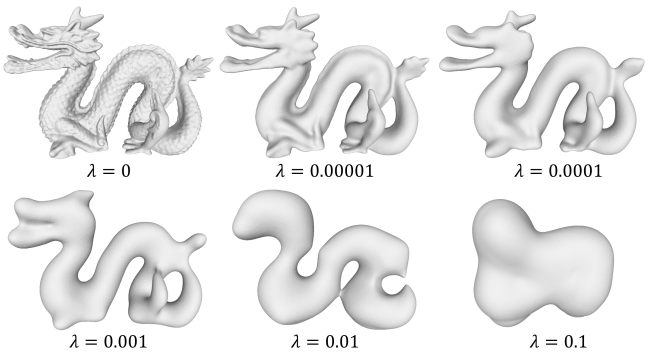


Fig. 9. NN-VIPSS at increasing λ on a 20K sampling of Dragon from [Laric 2012].

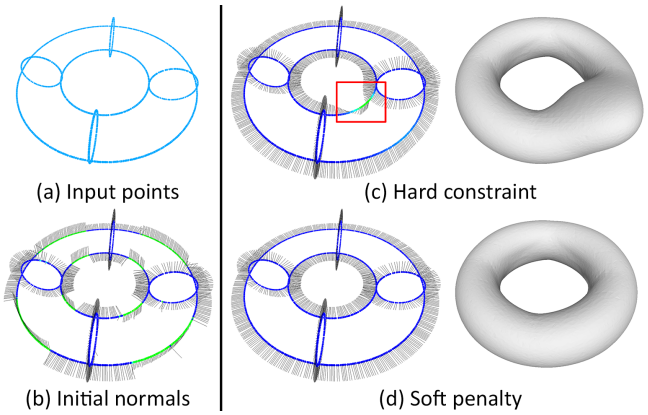


Fig. 10. Comparing optimized normals and surfaces using a hard constraint (Equation 10) (c) versus a soft penalty term (Equation 12) (d) given the same initial normals (b) ($\lambda = 0$). Input points (a) are sampled on cross-sections of a torus. Points in (b,c,d) are jet-colored by angular differences between computed and ground truth normals (blue/green/red means parallel/orthogonal/anti-parallel directions).

gradients to have unit lengths as a hard constraint in Equation 10. For this extremely sparse input, initialization produces many incorrect normals (see (b)), which are difficult to rectify using the constrained approach (see the red box in (c)). In contrast, the soft penalty affords more flexibility, which allows the optimization to find a better solution (see (d)). In our test suite, we did not observe notable differences for α in the range between 10 and 100, and we set $\alpha = 50$ in all tests.

8.3.2 Comparison setup. Besides VIPSS, we will compare with recent methods for normal estimation, including two-step methods such as Dipole [Metzer et al. 2021] and IsoConstraint [Xiao et al. 2023], and variational approaches such as PGR [Lin et al. 2022] (GPU only), iPSR [Hou et al. 2022], GCNO [Xu et al. 2023], and WNNC [Lin et al. 2024] (both GPU and CPU versions). We also consider the latest data-driven method for implicit reconstruction without normals, PPSurf [Erler et al. 2024]. The normal estimation methods involve multiple parameters. We found that their effects on the results are not always intuitive, and so we leave most of them at default settings. WNNC has several preset configurations (called *levels*) for controlling the smoothing of the winding number field. These levels are tailored for different amounts of noise, and we found that no single level works well for all sparse or non-uniform inputs. We therefore hand-tune the level for each example and report the one that performs best.

We compare the normals computed by these methods with ours (i.e., the vectors in the optimized Hermite data) as well as the resulting surfaces. Most of these methods resort to SPR for surface reconstruction from the estimated normals. As SPR may fail on sparse points even when the normals are correct (see Figure 8), for fairness and consistency, we use our NND interpolant for all methods on noise-free examples where exact interpolation is desired. We have verified that NND reconstructions are visually comparable, if not better, than SPR on all such examples. We adopt SPR as the reconstruction method for other methods on noisy examples, where the desirable surfaces approximate instead of interpolate the points. In this case, NND is not applicable for other methods, since it needs Hermite data with non-zero values that are only provided by our method and VIPSS. We use default parameters in SPR and an octree depth of 10.

8.3.3 Qualitative comparisons. We start with visual comparisons. Figures 11 and 12 show the results on small-scale inputs - up to a few thousand points - where all methods, including VIPSS, can be applied. Observe that NN-VIPSS, like VIPSS, produces plausible normals and surfaces for extremely sparse (Figure 11) and non-uniform (Figure 12) distributions, whereas other methods produce greater, and often significant errors in normals that lead to notable surface artifacts. In particular, the two-step normal orientation methods (IsoConstraint and Dipole) rely on accurate un-oriented line directions, which are difficult to obtain from sparse points (e.g., the Torus examples). The same applies to iPSR, as it is based on SPR, which cannot handle extreme sparsity and large missing regions (see Figure 8). The other methods (WNNC, GCNO, PGR) are all based on the generalized winding numbers [Barill et al. 2018], which work well for uniformly sparse samples (e.g., Torus 1) but struggle with non-uniform samples (e.g., Torus 2 and curve samples) and around thin shape features (e.g., Bathtub and Chair). Note that WNNC needs different level parameters (0 and 5) to get the best results.

We next consider larger inputs consisting of tens or hundreds of thousands of points. Methods that are limited to a few thousand points, such as VIPSS and GNCO, or low tens of thousands of points, such as PGR, are excluded from this evaluation. The examples in Figures 1 and 13 are created by non-uniform Monte-Carlo sampling of 3D shapes containing fine features, such as thin sheets (e.g., Helmets and Brain) and narrow tubes (e.g., Mobius), where point density is low relative to the feature size; see close-ups on Brain and Mobius. Such sparse sampling scenarios, similar to those in Figure 11 (e.g., Torus 1 and Bathtub), are challenging for existing normal estimation methods, which tend to create many topological errors such as holes and disconnections. In contrast, our method robustly captures these fine features and can more faithfully recover the surface topology. For example, our Mobius reconstruction has the closest genus (1311) to the ground truth (1316).

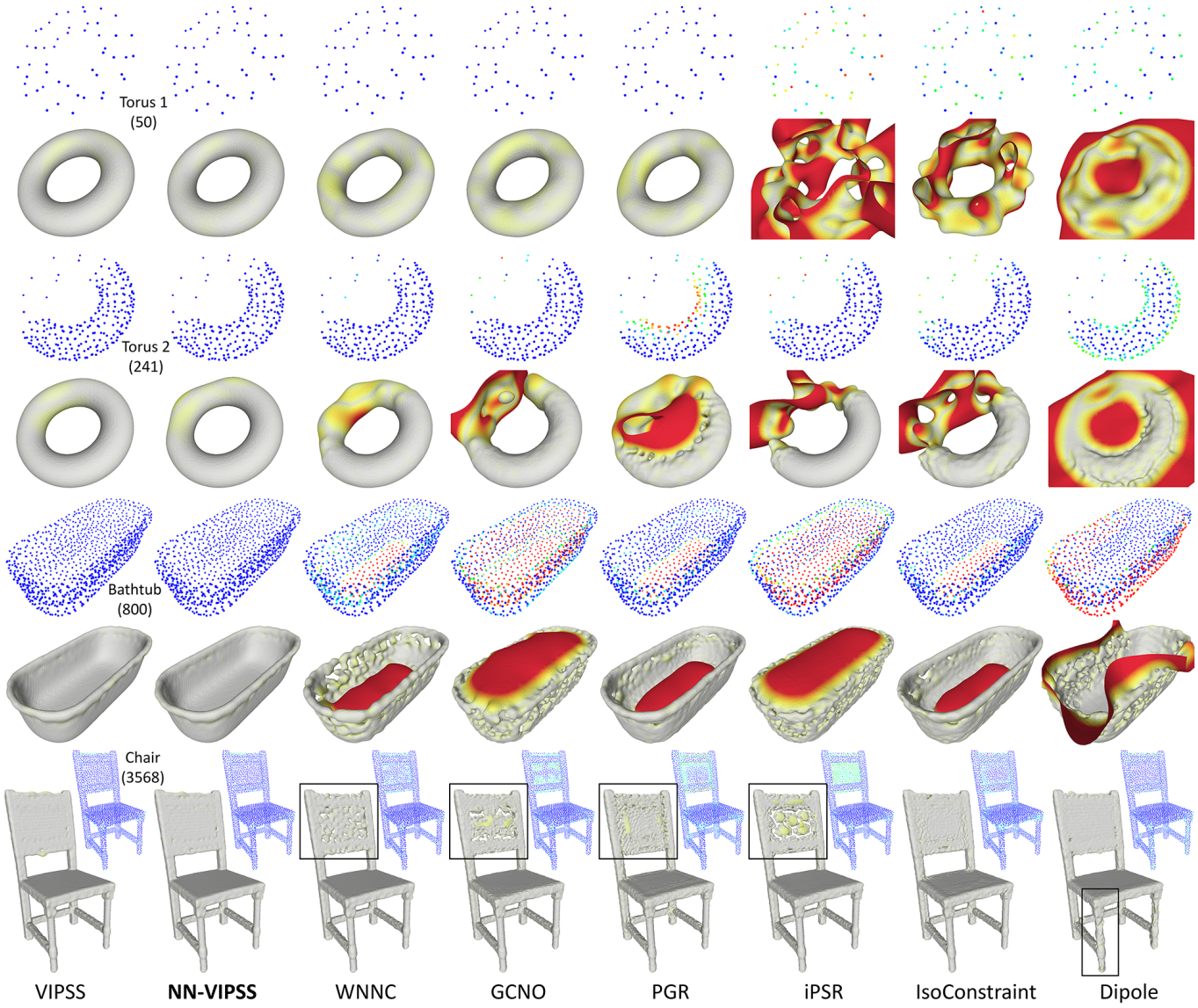


Fig. 11. Comparing the normals and surfaces produced by various methods on sparse samples of 3D shapes (Tori and Bathtub taken from [Huang et al. 2019] and Chair from [Huang et al. 2024]). Points are colored by normal differences with ground truth, and surfaces are colored by per-vertex distances to the original shape (red for distances greater than 0.15). NN-VIPSS uses $\lambda = 0$. WNNC level is 0 for Torus 1 and Chair, and 5 for Torus 2 and Bathtub. All surfaces are produced by NND.

Moving onto noisy data, we consider two simulated scans from the benchmark [Berger et al. 2017] (Figure 14). These scans contain multiple artifacts, such as noise, outliers, and large missing areas. Compared to WNNC, which is one of the best performers in our previous experiments, our method produces better reconstructions near sparsely sampled regions (e.g., cigar in Lord) and missing parts (e.g., legs in Lord and shaft hole in Anchor). We also compare with SPR using the normals provided in the input scans. Even with the additional normal input, SPR still produces large artifacts near missing parts, echoing our previous observations in synthetic examples (Figure 8).

Finally, we compare our method with the data-driven method, PPSurf [Erler et al. 2024], in Figures 1 and 15. While the learned priors allow PPSurf to smoothly surface noisy and dense samples, it tends to over-smooth geometric features (e.g. sealed openings on Helmet) and struggles with sparse inputs (e.g., Hand and Bathtub).

8.3.4 Quantitative comparisons. We report in Tables 2 and 3 the errors in the normals and surfaces computed by various methods

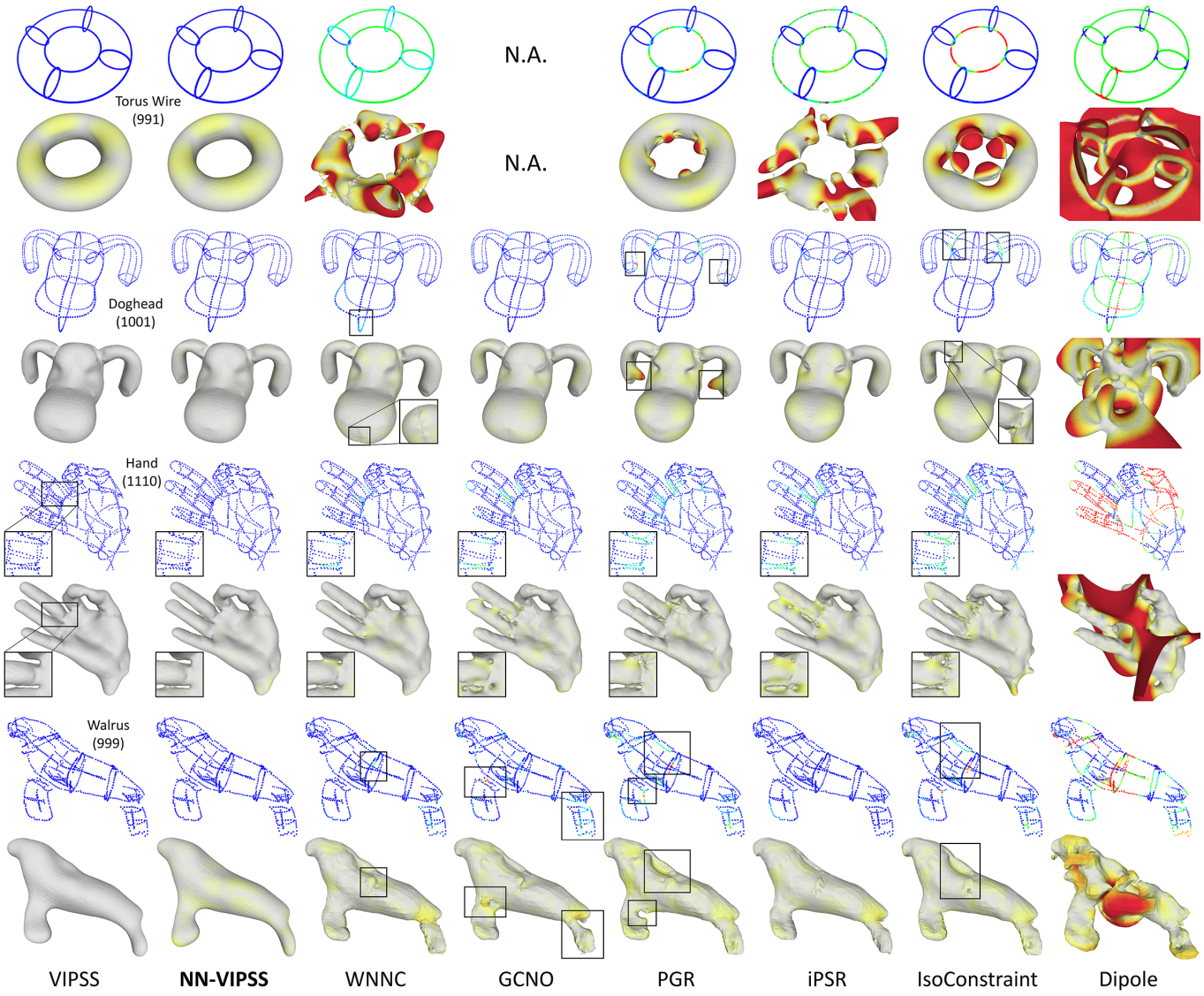


Fig. 12. Comparing the normals and surfaces produced by various methods on curve samples (taken from [Huang et al. 2019]). Points are colored by normal differences with VIPSS, and surfaces are colored by distances to the VIPSS surface (red for distances greater than 0.15). NN-VIPSS uses $\lambda = 0$ except Walrus where $\lambda = 0.0005$, and WNNC level is 5. GCNO did not terminate on Torus Wire after running for 30 minutes. All surfaces are produced by NND interpolation except Walrus, where VIPSS and NN-VIPSS use NND and other methods use SPR [Kazhdan and Hoppe 2013].

for all examples in Figures 11, 12, 13, and 14. The normal errors are calculated as the mean of $(1 - g_i \cdot g_i^*)/2$, where g_i, g_i^* are the computed and ground truth normals at an input point x_i, over all points. The surface errors are reported by the Chamfer Distance (CD) between 100K uniform samples of the computed and ground truth surfaces. We use VIPSS normals and surfaces as the ground truth for the curve samples in Figure 12. Normal errors are omitted for the examples in Figure 14 as no ground truth normals are provided in the benchmark [Berger et al. 2017]. In all examples, NN-VIPSS achieves the lowest errors in both normals and surfaces among all methods other than VIPSS.

We further evaluate our method on the latest surface reconstruction benchmark [Huang et al. 2024]. This dataset includes curated models from multiple sources, including Thingi10k [Zhou and Jacobson 2016], 3DNet [Wohlkinger et al. 2012], ABCD [Koch et al. 2019], and Three D Scans [Laric 2012]. After removing non-manifold and self-intersecting surfaces, we sample the remaining 1419 models each by 100K points using Monte Carlo sampling. We compare our NN-VIPSS with methods scalable to large inputs, including WNNC, iPSR, IsoConstraint, and Dipole. For a fair comparison, we use SPR to reconstruct surfaces from the normals computed by all methods, including our NN-VIPSS. The mean errors over all models, for

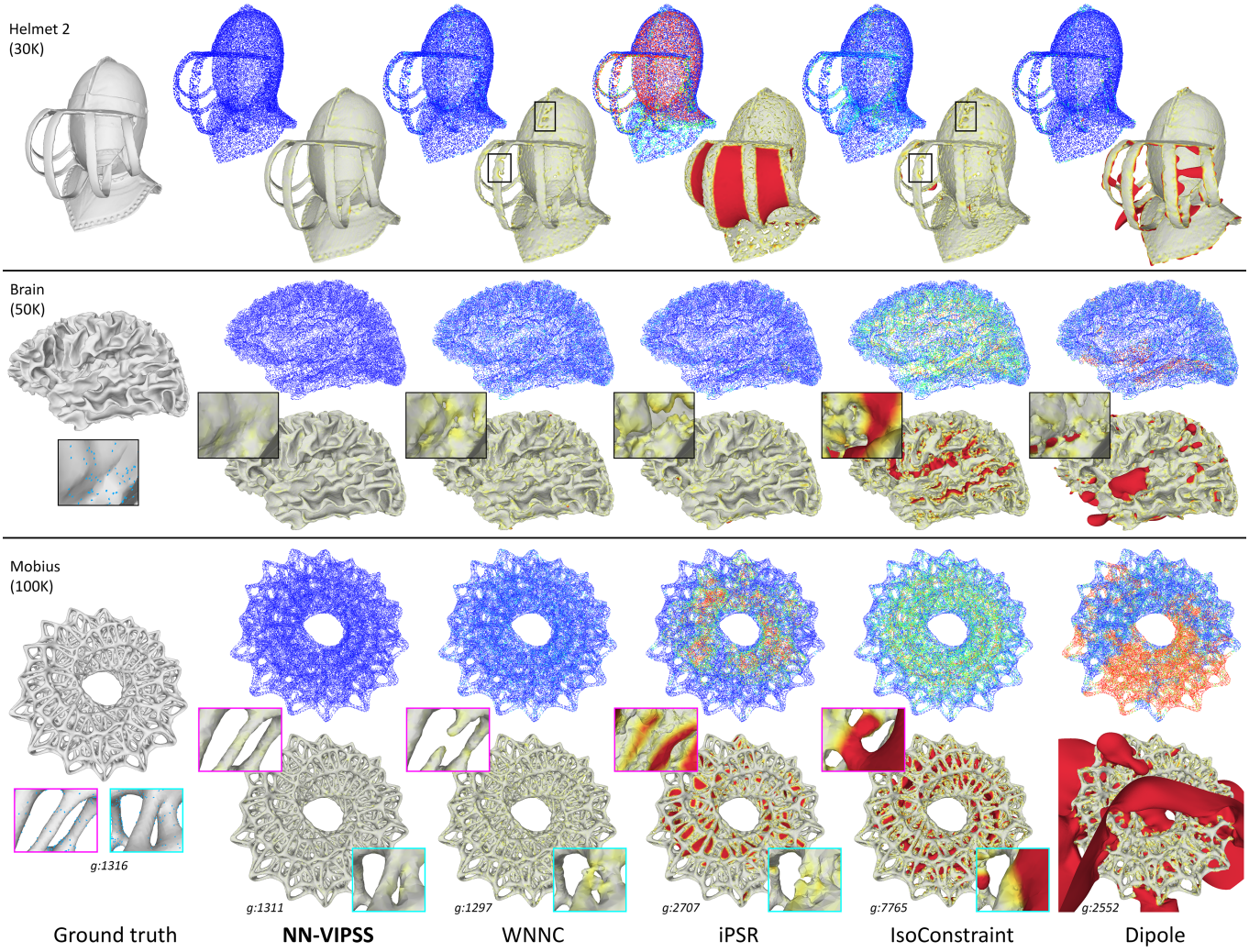


Fig. 13. Comparing the normals and surfaces produced by various methods on points sampled from 3D shapes (Helmet 2 from [Laric 2012] and Mobius from [Lin et al. 2024]). Points are colored by normal differences with ground truth and surfaces are colored by distances to ground truth (red for distances greater than 0.02). NN-VIPSS uses $\lambda = 0$. WNNC level is 0 for Brain and Mobius and 1 for Helmet 2. All surfaces are produced by NND interpolation. Genus is noted (g) for Mobius ground truth and reconstructions.

Table 2. Normal error (NE) and Chamfer Distance (CD) of all methods for examples in Figures 11 and 12. Lowest and second-lowest errors for each example are respectively in ***bold-italic*** and **bold**.

Method	VIPSS		NN-VIPSS		WNNC		GCNO		PGR		iPSR		IsoConstraint		Dipole	
	NE	CD	NE	CD	NE	CD	NE	CD	NE	CD	NE	CD	NE	CD	NE	CD
Torus 1	***0.0002***	***0.0053***	**0.0009**	**0.0062**	0.0076	0.0108	0.0057	0.0100	0.0064	0.0105	0.4595	0.1154	0.2699	0.0641	0.2024	0.1312
Torus 2	***0.0001***	***0.0069***	**0.0003**	**0.0103**	0.0068	0.0225	0.0214	0.0652	0.2164	0.0571	0.0363	0.0690	0.0637	0.0798	0.2222	0.1133
Bathtub	***0.0038***	**0.0132**	**0.0047**	***0.0124***	0.1615	0.0181	0.4595	0.0318	0.1422	0.0160	0.4771	0.0301	0.1289	0.0146	0.3575	0.0347
Chair	***0.0175***	***0.0043***	**0.0177**	***0.0043***	0.0298	0.0045	0.0500	0.0048	0.0827	0.0052	0.0689	0.0053	0.0561	0.0047	0.0418	0.0045
Torus Wire	***0.0001***	***0.0107***	***0.0001***	***0.0107***	0.3647	0.0633	-	-	0.0978	0.0124	0.2530	0.0708	0.1594	0.0343	0.4536	0.1594
Doghead	***0***	***0***	**0.0016**	**0.0048**	0.0226	0.0067	0.0165	0.0081	0.0650	0.0120	0.0175	0.0092	0.0529	0.0118	0.2543	0.0980
Hand	***0***	***0***	**0.0010**	**0.0041**	0.0194	0.0058	0.0374	0.0066	0.0530	0.0074	0.0399	0.0074	0.0615	0.0097	0.4711	0.0879
Walrus	***0***	***0***	**0.0078**	**0.0090**	0.0243	0.0096	0.0633	0.0099	0.0808	0.0128	0.0172	0.0098	0.0749	0.0098	0.2992	0.0420

both normals and surfaces, are reported in Table 4. Once again, our method achieves the lowest errors among all methods.

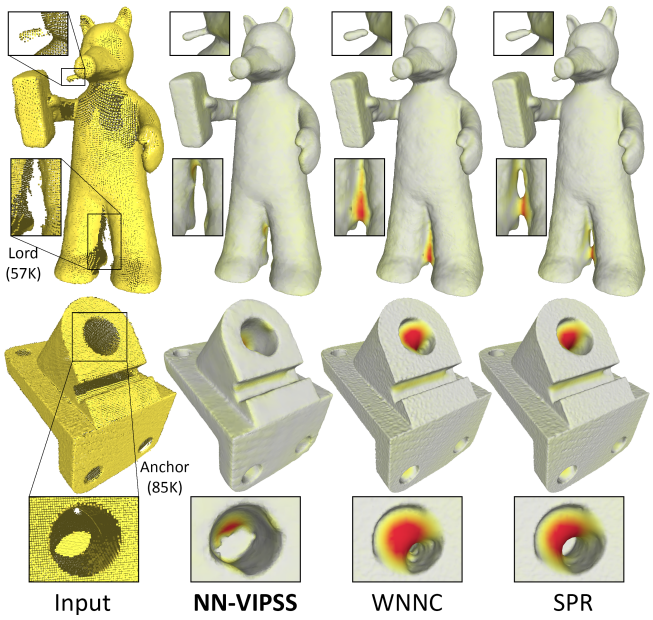


Fig. 14. Comparing our method ($\lambda = 0.0002$ for Lord and 0.001 for Anchor) with WNNC (using level 1 and surfaced by SPR) and SPR (using normals provided in the input) on incomplete and noisy scans from [Berger et al. 2017]. Surfaces are colored by distances to ground truth dense scans (red for distances greater than 0.04 for Lord and 0.1 for Anchor). Points (left) are shaded using the provided normals (not used for reconstruction by NN-VIPSS or WNNC).

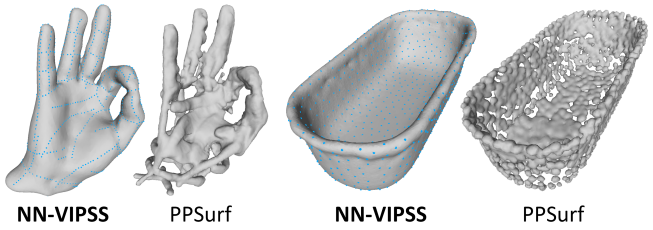


Fig. 15. Comparing our method ($\lambda = 0$) with PPSurf [Erler et al. 2024] on sparse samples (blue dots over NN-VIPSS surfaces), Hand (Figure 12) and Bathtub (Figure 11).

8.4 Performance

The running times of our and other methods on larger examples in the paper are reported in Table 5. We separately report the optimization and interpolation stages of our method, while the latter includes surfacing using the adaptive method of [Ju et al. 2024]. We only report the timing of other methods for computing the normals. While not the most efficient, our method achieves comparable speed as others. Note that our optimization takes significantly longer time for inputs with noise (e.g., Lord and Anchor), where non-zero lambda values are used. This is one of the limitations of our method, which will be discussed further in Section 9.

Table 3. Normal error (NE) and Chamfer Distance (CD) of all methods for examples in Figures 13 and 14. Lowest errors for each example are in **bold**.

Method	NN-VIPSS		WNNC		iPSR		IsoConstraint		Dipole	
	NE	CD	NE	CD	NE	CD	NE	CD	NE	CD
Helmet 2	**0.0019**	**0.0056**	0.0633	0.0059	0.4259	0.0157	0.0682	0.0062	0.0260	0.0163
Brain	**0.0020**	**0.0058**	0.0484	0.0060	0.0334	0.0060	0.2645	0.0108	0.1088	0.0084
Mobius	**0.0045**	**0.0070**	0.0491	0.0071	0.2659	0.0088	0.2965	0.0112	0.4527	0.0470
Lord	-	**0.0043**	-	0.0044	-	0.0045	-	0.0045	-	0.0044
Anchor	-	**0.0075**	-	0.0099	-	0.0104	-	0.0104	-	0.0100

Table 4. Mean Normal errors (NE) and Chamfer Distances (CD) on a large-scale benchmark [Huang et al. 2024]. Lowest errors are in **bold**.

Method	NN-VIPSS	WNNC	iPSR	IsoConstraint	Dipole
NE	**0.00598**	0.00776	0.01122	0.06401	0.04015
CD	**0.00221**	0.00222	0.00227	0.00254	0.00468

Table 5. Running time (in seconds) of various methods on the larger inputs shown in this paper, as well as three taken from [Laric 2012] (Bressant and Dragon use the original samples, and Murex uses a Monte Carlo sampling). Figure numbers are in parentheses. NN-VIPSS time is separated into optimization (including all solver iterations) and interpolation (including surfacing). Timings of other methods do not include surfacing.

	# points	**NN-VIPSS** opt.+interp.	WNNC gpu (cpu)	iPSR	Iso	Dipole
Helmet 2 (13)	30,000	7.0+16.2	1.8 (2.7)	124.8	131.2	21.0
Brain (13)	50,000	9.4+14.0	2.9 (5.3)	158.7	131.5	45.7
Lord (14)	57,264	314.4+14.3	2.2 (3.7)	38.8	85.5	52.8
Anchor (14)	85,127	456.7+14.9	3.3 (5.5)	103.5	145.7	118.6
Helmet (1)	100,000	20.8+49.5	4.9 (8.5)	233.4	168.5	138.1
Mobius (13)	100,000	22.7+32.0	6.5 (10.9)	1187.6	213.3	139.4
Bressant	532,913	197.8+160.8	16.9 (38.4)	401.3	244.2	137.9
Murex	1,000,000	199.5+254.3	87.1 (204.2)	594.6	489.3	221.3
Dragon	1,180,060	186.2+292.0	72.9 (143.8)	576.4	479.1	274.6

To validate the theoretical complexity analysis in Section 7, we will consider two examples that illustrate the best-case and worst-case scenarios (both using $\lambda = 0$). For the best-case scenario (Figure 16), we sample uniformly on smooth surfaces, including up to 4000 points on Kitten (a) and up to 1M points on Dragon (b,c,d). Observe in (a) that VIPSS exhibits a fast, polynomial increase in running time, making it impractical to use beyond a few thousand points. In contrast, NN-VIPSS takes negligible time for small sample sizes. For larger inputs, as shown in (b), each step of NN-VIPSS scales near linearly with the number of samples, matching the analysis in Table 1. The sudden drop of Optimization Solve time after 700K samples is because the points are dense enough such that the initial Hermite data is sufficiently close to the minimizer. Our method also exhibits a similar scaling behavior as other methods, as shown in (c). Finally, we analyze the natural neighborhood sizes at different sample sizes in (d). Even though the number of natural neighbors grows linearly for some points as density increases, as seen from the maximum curve, the majority of the points have constant, small neighborhood sizes, as evident in the low average and standard deviation. This explains the scalability of our method in this scenario.

We demonstrate the worst-case scenario (Figure 17) by uniformly sampling a curve network (Torus Wire in Figure 12) with increasing density. Observe in (b) that the natural neighborhood sizes for all points grow linearly with the sample size. As a result, our method

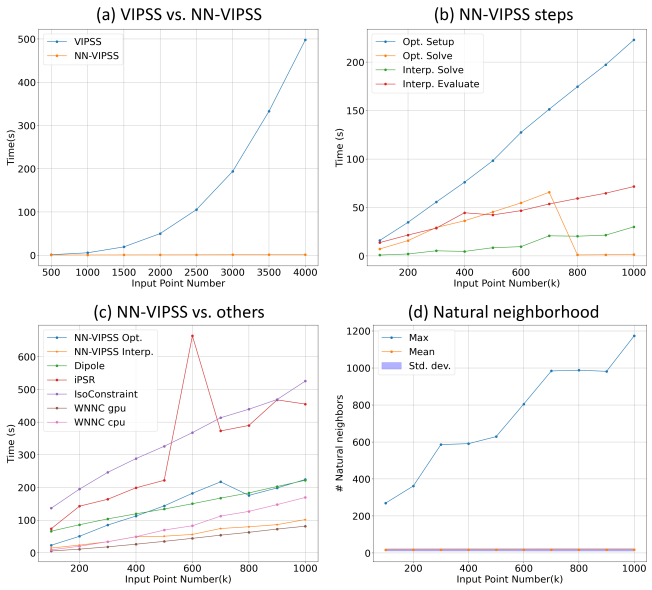


Fig. 16. Performance on uniform surface samples. (a): Comparing running time of VIPSS and NN-VIPSS on samples of the Kitten (Figure 7) up to 4000 points. (b): Timing of each step of NN-VIPSS in Table 1 on samples of the Dragon [Laric 2012] up to 1M points. (c): Comparing timing of all methods on the Dragon samples. (d): The mean/max/standard deviation of the number of natural neighbors per point for each Dragon sample size.

exhibits polynomial growth in time, as shown in (a), and quickly becomes too time-consuming even at 10K points. Note that such dense sampling along smooth curves is usually not necessary for the purpose of surface reconstruction. For example, our method can reconstruct a plausible surface from complex curve networks like those in Figure 12 using just a thousand samples.

9 Conclusion and limitations

This paper introduces a new surface reconstruction method for points without normals. By exploring the locality of natural neighborhoods, we reformulated a global, variational method [Huang et al. 2019] to significantly improve its scalability while retaining its robustness on sparse and non-uniform point distributions. Our method involves a single parameter, needs no discretization (except for surface extraction), and achieves comparable runtime to state-of-the-art methods while producing fewer surface artifacts in regions with insufficient samples.

An important limitation of our method is that its scalability is tied to the locality of natural neighborhoods, which in turn depends on the point distribution. While favorable bounds are only known under strict sampling conditions [Amenta et al. 2007], we found that the size of natural neighborhoods in practical inputs for surface reconstruction is typically small (Figure 16 (d)). This allows our methods to handle large inputs. However, our method does not scale well when the natural neighborhoods are near linear in size for a significant portion of the input points, for example, when the points are densely sampled along curves (Figure 17). Another situation of poor scalability would be a significant amount of outliers, since

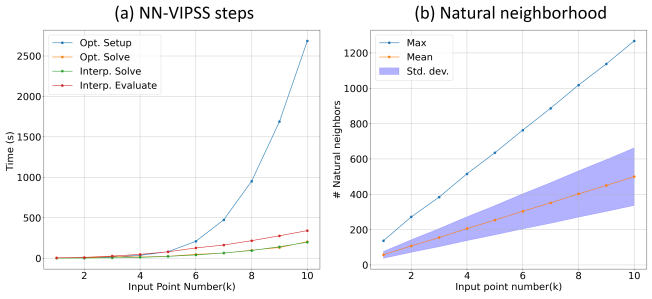


Fig. 17. Running time of NN-VIPSS on samples of Torus Wire (Figure 12) with increasing density (a) and statistics of natural neighborhood sizes (b). The linear growth in both the mean and standard deviation of the number of natural neighbors presents the worst-case scenario for our method in terms of scalability.

points away from the surface often have large natural neighborhoods. A potential remedy for this drawback is to *add* points into the input set where Hermite data will be optimized and interpolated. If located properly, the additional points may reduce the sizes of natural neighborhoods (in the extreme case, randomly distributed points have few natural neighbors as well [Dwyer 1989]).

Another limitation of our method is the slower convergence in optimization for larger λ and more data noise. As λ increases, the minimizing solution becomes smoother (see Figure 9), and the optimization problem becomes more global. As a result, the initial Hermite data computed using our local approach over each natural neighborhood may deviate further from the minimizer. As shown in Figure 18 (a), increasing λ requires more iterations to achieve the desired accuracy, and in turn longer optimization time, and such increases are more significant as the noise level rises. We take a closer look at one setting (0.2% noise level and $\lambda = 0.001$) by plotting the residues at each optimization iteration in (b) and showing four intermediate reconstructions in (c). While the residue drops quickly at the beginning of optimization, it plateaus afterwards with only marginal decreases, while the reconstructions exhibit gradual and localized changes (see the highlighted regions in (c)). Improving the convergence of our method on noisy inputs, which require larger λ, would be another important venue for future research.

Acknowledgments

This work is supported by NSF grant HCC-2401224 and a gift from Adobe Research. We would like to thank researchers who share their data and code, particularly Siyou Lin and Dong Xiao for their help in comparisons with [Lin et al. 2024; Xiao et al. 2023].

References

P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. 2007. Voronoi-based Variational Reconstruction of Unoriented Point Sets. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Barcelona, Spain) *(SGP '07)*. 39–48.

Nina Amenta, Dominique Attali, and Olivier Devillers. 2007. Complexity of delaunay triangulation for points on lower-dimensional˜ polyhedra. In *ACM-SIAM Symposium on Discrete Algorithms*. 1106–1113.

Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. 2001. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*. ACM, 249–266.

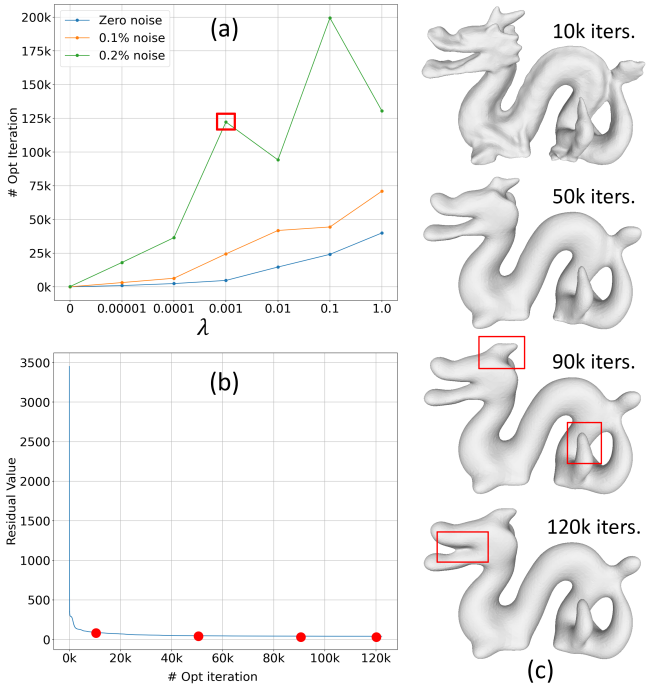


Fig. 18. (a) Optimization iterations till convergence (without capping at 10,000 iterations) on a 20K sampling of Dragon as λ increases and at different noise levels. Note that our method takes longer to converge with greater λ and more noise. (b) Residue over optimization iterations at 0.2% noise with $\lambda = 0.001$. This setting corresponds to the highlighted dot in (a). (c) Intermediate reconstructions at the four checkpoints highlighted in (b).

Ken Anjyo, John P Lewis, and Frédéric Pighin. 2014. Scattered data interpolation for computer graphics. In *ACM SIGGRAPH 2014 Courses*. 1–69.

Gavin Barill, Neil G Dickson, Ryan Schmidt, David IW Levin, and Alec Jacobson. 2018. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 43.

Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. 2017. A survey of surface reconstruction from point clouds. *Computer Graphics Forum* 36, 1 (2017), 301–329.

Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. 1999. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics* 5, 4 (1999), 349–359.

Tom Axel Bobach. 2009. *Natural neighbor interpolation-critical assessment and new contributions*. Ph. D. Dissertation. Technische Universität Kaiserslautern.

Jean-Daniel Boissonnat and Frédéric Cazals. 2002. Smooth surface reconstruction via natural neighbour interpolation of distance functions. *Computational Geometry* 22, 1-3 (2002), 185–203.

Alexandre Boulch and Renaud Marlet. 2012. Fast and Robust Normal Estimation for Point Clouds with Sharp Features. *Computer Graphics Forum* (2012).

Adrian Bowyer. 1981. Computing dirichlet tessellations. *The computer journal* 24, 2 (1981), 162–166.

E Brazil, Ives Macedo, M Costa Sousa, Luiz Henrique de Figueiredo, and Luiz Velho. 2010. Sketching variational hermite-rbf implicits. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*. Eurographics Association, 1–8.

Martin D Buhmann. 2003. *Radial basis functions: theory and implementations*. Vol. 12. Cambridge university press.

Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. 2001. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 67–76.

F. Cazals and M. Pouget. 2003. Estimating Differential Quantities Using Polynomial Fitting of Osculating Jets. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Aachen, Germany) *(SGP '03)*. 177–187.

Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5939–5948.

Tamal K Dey and Samrat Goswami. 2003. Tight cocone: a water-tight surface reconstructor. *Journal of Computing and Information Science in Engineering* 3, 4 (2003), 302–307.

Tamal K Dey and Jian Sun. 2005. An Adaptive MLS Surface for Reconstruction with Guarantees.. In *Symposium on Geometry processing*. 43–52.

Huong Quynh Dinh, Greg Turk, and Greg Slabaugh. 2002. Reconstructing surfaces by volumetric regularization using radial basis functions. *IEEE transactions on pattern analysis and machine intelligence* 24, 10 (2002), 1358–1371.

Jean Duchon. 1977. Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In *Constructive theory of functions of several variables*. Springer, 85–100.

Rex A Dwyer. 1989. Higher-dimensional Voronoi diagrams in linear expected time. In *Proceedings of the fifth annual symposium on Computational geometry*. 326–333.

Herbert Edelsbrunner and Nimish R Shah. 1994. Triangulating topological spaces. In *Proceedings of the tenth annual symposium on Computational geometry*. 285–292.

Philipp Erler, Lizeth Fuentes-Perez, Pedro Hermosilla, Paul Guerrero, Renato Pajarola, and Michael Wimmer. 2024. PPSurf: Combining Patches and Point Convolutions for Detailed Surface Reconstruction. In *Computer Graphics Forum*, Vol. 43. Wiley Online Library, e15000.

Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J Mitra, and Michael Wimmer. 2020. Points2surf learning implicit surfaces from point clouds. In *European Conference on Computer Vision*. Springer, 108–124.

Gerald Farin. 1990. Surfaces over Dirichlet tessellations. *Computer aided geometric design* 7, 1-4 (1990), 281–292.

Richard Franke. 1982. Smooth interpolation of scattered data by local thin plate splines. *Computers & mathematics with applications* 8, 4 (1982), 273–281.

Richard Franke and Greg Nielson. 1980. Smooth interpolation of large sets of scattered data. *International journal for numerical methods in engineering* 15, 11 (1980), 1691–1704.

Simon Giraudot, David Cohen-Steiner, and Pierre Alliez. 2013. Noise-adaptive shape reconstruction from raw point sets. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*. Eurographics Association, 229–238.

Leslie Greengard and Vladimir Rokhlin. 1987. A fast algorithm for particle simulations. *Journal of computational physics* 73, 2 (1987), 325–348.

Gaël Guennebaud and Markus Gross. 2007. Algebraic point set surfaces. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 23.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Si Hang. 2015. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw* 41, 2 (2015), 11.

Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. 2020. Point2Mesh: a self-prior for deformable meshes. *ACM Trans. Graph.* 39, 4, Article 126 (Aug. 2020), 12 pages.

John C Hart. 1996. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (1996), 527–545.

Hisamoto Hiyoshi and Kokichi Sugihara. 2004. Improving the global continuity of the natural neighbor interpolation. In *International conference on computational science and its applications*. Springer, 71–80.

Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. 1992. *Surface reconstruction from unorganized points*. Vol. 26. ACM.

Alexander Hornung and Leif Kobbelt. 2006. Robust reconstruction of watertight 3 d models from non-uniformly sampled point clouds without normal information. In *Symposium on geometry processing*. Citeseer, 41–50.

Fei Hou, Chiyu Wang, Wencheng Wang, Hong Qin, Chen Qian, and Ying He. 2022. Iterative poisson surface reconstruction (ipsr) for unoriented points. *arXiv preprint arXiv:2209.09510* (2022).

Zhiyang Huang, Nathan Carr, and Tao Ju. 2019. Variational implicit point set surfaces. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–13.

Zhangjin Huang, Yuxin Wen, Zihao Wang, Jinjuan Ren, and Kui Jia. 2024. Surface reconstruction from point clouds: A survey and a benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

Takashi Ijiri, Shin Yoshizawa, Yu Sato, Masaaki Ito, and Hideo Yokota. 2013. Bilateral Hermite Radial Basis Functions for Contour-based Volume Segmentation. *Computer Graphics Forum* 32, 2 (2013), 123–132. Proc. of EUROGRAPHICS'13.

Steven G. Johnson. 2007. The NLopt nonlinear-optimization package. https://github.com/stevengj/nlopt.

Yiwen Ju, Xingyi Du, Qingnan Zhou, Nathan Carr, and Tao Ju. 2024. Adaptive grid generation for discretizing implicit complexes. *ACM Trans. Graph.* 43, 4, Article 82 (July 2024), 17 pages. https://doi.org/10.1145/3658215

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson Surface Reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Cagliari, Sardinia, Italy) *(SGP '06)*. 61–70.

Michael Kazhdan and Hugues Hoppe. 2013. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)* 32, 3 (2013), 29.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A Big CAD Model Dataset For Geometric Deep Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ravikrishna Kolluri. 2008. Provably good moving least squares. *ACM Transactions on Algorithms (TALG)* 4, 2 (2008), 18.

Oliver Laric. 2012. Three D Scans. https://threedscans.com/.

Bao Li, Ruwen Schnabel, Reinhard Klein, Zhiquan Cheng, Gang Dang, and Jin Shiyao. 2010. Robust normal estimation for point clouds with sharp features. *Computers & Graphics* 34, 2 (2010), 94–106.

Siyou Lin, Zuoqiang Shi, and Yebin Liu. 2024. Fast and Globally Consistent Normal Orientation based on the Winding Number Normal Consistency. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–19.

Siyou Lin, Dong Xiao, Zuoqiang Shi, and Bin Wang. 2022. Surface reconstruction from point clouds without normals by parametrizing the gauss formula. *ACM Transactions on Graphics* 42, 2 (2022), 1–19.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming* 45 (1989), 503–528. https://doi.org/10.1007/bf01589116

Shengjun Liu, Charlie C.L. Wang, Guido Brunnett, and Jun Wang. 2016. A Closed-form Formulation of HRBF-based Surface Reconstruction by Approximate Solution. *Comput. Aided Des.* 78, C (Sept. 2016), 147–157.

DanFeng Lu, HongKai Zhao, Ming Jiang, ShuLin Zhou, and Tie Zhou. 2005. A surface reconstruction method for highly noisy point clouds. In *International Workshop on Variational, Geometric, and Level Set Methods in Computer Vision*. Springer, 283–294.

Wenjia Lu, Zuoqiang Shi, Jian Sun, and Bin Wang. 2018. Surface Reconstruction Based on the Modified Gauss Formula. *ACM Trans. Graph.* 38, 1, Article 2 (Dec. 2018), 18 pages.

Yueji Ma, Yanzun Meng, Dong Xiao, Zuoqiang Shi, and Bin Wang. 2024. Flipping-based iterative surface reconstruction for unoriented points. *Computer Aided Geometric Design* 111 (2024), 102315.

Josiah Manson, Guergana Petrova, and Scott Schaefer. 2008. Streaming surface reconstruction using wavelets. *Computer Graphics Forum* 27, 5 (2008), 1411–1420.

Peter McMullen. 1970. The maximum numbers of faces of a convex polytope. *Mathematika* 17, 2 (1970), 179–184.

Quentin Merigot, Maks Ovsjanikov, and Leonidas J. Guibas. 2011. Voronoi-Based Curvature and Feature Estimation from Point Clouds. *IEEE Transactions on Visualization and Computer Graphics* 17, 6 (June 2011), 743–756.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4460–4470.

Gal Metzer, Rana Hanocka, Denis Zorin, Raja Giryes, Daniele Panozzo, and Daniel Cohen-Or. 2021. Orienting point clouds with dipole propagation. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.

N. J. Mitra, A. Nguyen, and L. Guibas. 2004. Estimating Surface Normals in Noisy Point Cloud Data. In *special issue of International Journal of Computational Geometry and Applications*, Vol. 14. 261–276.

B. S. Morse, T. S. Yoo, P. Rheingans, D. T. Chen, and K. R. Subramanian. 2001. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Proceedings International Conference on Shape Modeling and Applications*. 89–98.

Patrick Mullen, Fernando De Goes, Mathieu Desbrun, David Cohen-Steiner, and Pierre Alliez. 2010. Signing the unsigned: Robust surface reconstruction from raw pointsets. *Computer Graphics Forum* 29, 5 (2010), 1733–1741.

Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. 2003a. Multi-level partition of unity implicits. In *ACM Transactions on Graphics (TOG)*, Vol. 22. ACM, 463–470.

Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. 2003b. A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions. In *Shape Modeling International, 2003.* IEEE, 153–161.

A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. 2009. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum* 28, 2 (2009), 493–501.

Rongjiang Pan and Vaclav Skala. 2012. Surface Reconstruction with higher-order smoothness. *The Visual Computer* 28, 2 (2012), 155–162.

Roi Poranne, Craig Gotsman, and Daniel Keren. 2010. 3D surface reconstruction using a generalized distance function. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 2479–2491.

M. Samozino, M. Alexa, P. Alliez, and M. Yvinec. 2006. Reconstruction with Voronoi Centered Radial Basis Functions. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Cagliari, Sardinia, Italy). 51–60.

Conrad Sanderson and Ryan Curtin. 2016. Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software* 1, 2 (2016), 26.

Nico Schertler, Bogdan Savchynskyy, and Stefan Gumhold. 2017. Towards Globally Optimal Normal Orientations for Large Point Clouds. *Comput. Graph. Forum* 36, 1 (Jan. 2017), 197–208.

Bernhard Schölkopf, Joachim Giesen, and Simon Spalinger. 2004. Kernel Methods for Implicit Surface Modeling. In *Proceedings of the 17th International Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada) *(NIPS'04)*. 1193–1200.

Chen Shen, James F. O'Brien, and Jonathan R. Shewchuk. 2004. Interpolating and Approximating Implicit Surfaces from Polygon Soup. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 896–904.

Robin Sibson. 1980. A vector identity for the Dirichlet tessellation. In *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 87. Cambridge University Press, 151–155.

Robin Sibson. 1981. A brief description of natural neighbour interpolation. *Interpreting multivariate data* (1981), 21–36.

Gabriel Taubin. 2012. Smooth signed distance surface reconstruction and applications. In *Iberoamerican Congress on Pattern Recognition*. Springer, 38–45.

Greg Turk and James F. O'Brien. 2002. Modelling with Implicit Surfaces That Interpolate. *ACM Trans. Graph.* 21, 4 (Oct. 2002), 855–873.

C. Walder, O. Chapelle, and B. Schölkopf. 2005. Implicit Surface Modelling as an Eigenvalue Problem. In *Proceedings of the 22nd International Conference on Machine Learning*. ACM, 937–944.

Christian Walder, Olivier Chapelle, and Bernhard Schölkopf. 2007. Implicit surfaces with globally regularised and compactly supported basis functions. In *Advances in Neural Information Processing Systems*. 273–280.

Jun Wang, Zhouwang Yang, and Falai Chen. 2011. A variational model for normal computation of point clouds. *The Visual Computer* 28 (2011), 163–174.

David F Watson. 1981. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The computer journal* 24, 2 (1981), 167–172.

Holger Wendland. 1995. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in computational Mathematics* 4 (1995), 389–396.

Holger Wendland. 2004. *Scattered data approximation*. Vol. 17. Cambridge university press.

Walter Wohlkinger, Aitor Aldoma, Radu B Rusu, and Markus Vincze. 2012. 3dnet: Large-scale object class recognition from cad models. In *2012 IEEE international conference on robotics and automation*. IEEE, 5384–5391.

Zhang Xianyi, Wang Qian, and Zhang Yunquan. 2012. Model-driven level 3 BLAS performance optimization on Loongson 3A processor. In *2012 IEEE 18th international conference on parallel and distributed systems*. IEEE, 684–691.

Dong Xiao, Zuoqiang Shi, Siyu Li, Bailin Deng, and Bin Wang. 2023. Point normal orientation and surface reconstruction by incorporating isovalue constraints to Poisson equation. *Computer Aided Geometric Design* 103 (2023), 102195.

Rui Xu, Zhiyang Dou, Ningna Wang, Shiqing Xin, Shuangmin Chen, Mingyan Jiang, Xiaohu Guo, Wenping Wang, and Changhe Tu. 2023. Globally consistent normal orientation for point clouds by regularizing the winding-number field. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–15.

Hong-Kai Zhao, Stanley Osher, and Ronald Fedkiw. 2001. Fast surface reconstruction using the level set method. In *Proceedings IEEE Workshop on Variational and Level Set Methods in Computer Vision*. IEEE, 194–201.

Deyun Zhong, Liguan Wang, and Lin Bi. 2020. A Fast Solution for the Generalized Radial Basis Functions Interpolant. *IEEE Access* 8 (2020), 80148–80159.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).

A Proof of Proposition 5.1

Proof. We first verify the interpolation of values S and linear reproduction. The Lagrange property of w_i and the interpolation of f_{S_i,G_i} at x_i implies $\tilde{f}_{S,G}(x_i) = f_{S_i,G_i}(x_i) = s_i$. Furthermore, if S and G are derived from some linear function f, each f_{S_i,G_i} is the same as f due to linear reproduction of Duchon's interpolant. As w_i form a partition of unity, $\tilde{f}_{S,G}$ reproduces the same linear function f.

We next prove the smoothness of $\tilde{f}_{S,G}$ and interpolation of gradients G. In one dimension ($d = 1$), it is easy to verify that $\tilde{f}_{S,G}$ is identical to the global Duchon's interpolant $f_{S,G}$, which is C^1 continuous and interpolates the prescribed gradients. For $d \geq 2$, we know that $w_i(x)$ have at least C^1 continuity within the convex hull of Y except at X. Since Duchon's interpolants are C^1, this implies that $\tilde{f}_{S,G}$ is C^1 away from X. It remains to show that the gradient of $\tilde{f}_{S,G}$ exists at each point $x_i \in X$ and equals g_i.

For notational simplicity, we replace $f_{S,G}$ by f, $\tilde{f}_{S,G}$ by $\tilde{f}$, and f_{S_i,G_i} by f_i. Let $I(x)$ be the indices of those x_i that are the natural neighbors of x in $\{x\} \cup X \cup Y$. Using the product rule,

$$D\tilde{f}(x) = \sum_{j\in I(x)} Dw_j(x) f_j(x) + \sum_{j\in I(x)} Df_j(x) w_j(x).$$

Due to the Lagrange property of w_j, and since $f_j(x_i) = s_i$ and $Df_j(x_i) = g_i$ for every $j \in I(x_i)$,

$$\begin{aligned} D\tilde{f}(x_i) &= s_i \sum_{j\in I(x_i)} Dw_j(x_i) + Df_i(x_i) \\ &= s_i D\left(\sum_{j\in I(x_i)} w_j(x_i)\right) + g_i \end{aligned}$$

for $i = 1, \ldots, n$. Due to partition of unity, $\sum_{j\in I(x)} w_j(x)$ is constant (0) for any x in the convex hull of Y, and hence its derivative exists and is zero everywhere (including $x = x_i$). Therefore the first term on the rhs vanishes and we conclude $D\tilde{f}(x_i) = g_i$. □

B Proof of Proposition 6.1

Proof. We draw upon a result from [Huang et al. 2019] (Proposition 4.1) about the matrix J_i in Equation 10 for subset X_i of X. A square matrix of length $(d+1)|X_i|$, J_i can be written in a block form,

$$J_i = \begin{pmatrix} J_{i,00} & J_{i,01} \\ J_{i,01}^T & J_{i,11} \end{pmatrix},$$

where $J_{i,00}$ is a square matrix with length $|X_i|$. Then matrix $\hat{J}_i$ for the corresponding subset $\hat{X}_i$ of $\hat{X}$ has the form:

$$\hat{J}_i = \begin{pmatrix} \sigma^{-3} J_{i,00} & \sigma^{-2} J_{i,01} \\ \sigma^{-2} J_{i,01}^T & \sigma^{-1} J_{i,11} \end{pmatrix}.$$

Building on this result, and let $\hat{S} = \sigma S, \hat{G} = G$, and $\hat{\lambda} = \sigma^3\lambda$, we can relate the objective of Equation 10 w.r.t $\{\hat{X}, \hat{\lambda}\}$ and $\{X, \lambda\}$ as:

$$\hat{S}^T\hat{S} + \hat{\lambda}\sum_i \begin{pmatrix} \hat{S}_i^T & \hat{G}_i^T \end{pmatrix} \hat{J}_i \begin{pmatrix} \hat{S}_i \\ \hat{G}_i \end{pmatrix} = \sigma^2 (S^T S + \lambda \sum_i \begin{pmatrix} S_i^T & G_i^T \end{pmatrix} J_i \begin{pmatrix} S_i \\ G_i s \end{pmatrix})$$

As a result, if $\{S^*, G^*\}$ minimizes the objective w.r.t. $\{X, \lambda\}$, then $\{\hat{S}^* = \sigma S^*, \hat{G}^* = G^*\}$ minimizes the objective w.r.t. $\{\hat{X}, \hat{\lambda}\}$. To prove the second half of the proposition, [Huang et al. 2019] (Proposition 4.1) shows that $f_{\hat{S},\hat{G}}(x) = \sigma f_{S,G}(x/\sigma)$ for any $x \in \mathbb{R}^d$. On the other hand, the blending weights w_i, like NNC, are invariant under uniform scaling. We therefore derive:

$$\begin{aligned} \tilde{f}_{\hat{S}^*,\hat{G}^*}(x) &= \sum_i w_i(x) f_{\hat{S}_i^*,\hat{G}_i^*}(x) \\ &= \sigma \sum_i w_i(x) f_{S_i^*,G_i^*}(x/\sigma) \\ &= \sigma \tilde{f}_{S^*,G^*}(x/\sigma) \end{aligned}$$

□

C Initialization

To initialize the NN-VIPSS optimization (Equation 10), we first compute Hermite data at each point $x_i \in X$ by solving a simplified variational problem, similar to VIPSS, over the subset X_i. Instead of constraining *all* vectors g_j for $x_j \in X_i$ to be unit vectors, we only constrain one vector, g_i, to be a unit vector. As we shall see, the simplification reduces the variational problem to computing the eigenvector of a small $d \times d$ matrix.

For generality and ease of notation, we will formulate the problem as follows: given a point set $X = \{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$, we seek a smooth implicit function whose zero level set approximates X and whose gradient at x_1 has unit length. We can represent the function as an interpolant $f_{S,g}$ of scalars $S = \{s_1, \ldots, s_n\}$, one at each point, and a single vector g at x_1. The problem can be formulated as finding $\{S, g\}$ that

$$\begin{aligned} &\text{Minimizes:} && S^T S + \lambda\, E(f_{S,g}) \\ &\text{Subject to:} && g^T g = 1. \end{aligned} \tag{13}$$

Similar to Duchon's interpolant, a smooth interpolant $f_{S,g}$ can be defined using an extended form of HRBF (see [Wendland 2004], Chapter 16.2) as

$$f_{S,g}(\mathbf{x}) = \sum_{i=1}^{n} a_i \phi(x, x_i) + b^T D^{0,1}\phi(x, x_1) + p^T X + q$$

where ϕ is the triharmonic kernel. The coefficients $a_i \in \mathbb{R}, b \in \mathbb{R}^d, p \in \mathbb{R}^d, q \in \mathbb{R}$ can be uniquely determined by a similar linear system that enforces interpolation ($f_{S,g}(x_i) = s_i$ for all i and $Df_{S,g}(x_1) = g$) and additional constraints ($\sum_i a_i = 0$ and $\sum_i a_i x_i + b = 0$):

$$M \begin{pmatrix} A \\ b \\ p \\ q \end{pmatrix} = \begin{pmatrix} S \\ g \\ 0 \\ 0 \end{pmatrix},$$

where the interpolation matrix M has length $n + 2d + 1$. Duchon's energy of $f_{S,g}$ bears a similar form as that of Duchon's interpolant (Equation 6):

$$E(f_{S,g}) = \begin{pmatrix} S^T & g^T \end{pmatrix} J \begin{pmatrix} S \\ g \end{pmatrix}, \tag{14}$$

where J is the top-left block of length $n + d$ in M^{-1}. We will write J in a block form as

$$J = \begin{pmatrix} J_{00} & J_{01} \\ J_{01}^T & J_{11} \end{pmatrix},$$

where J_{00} is a square matrix with length n. Substituting the energy definition in Equation 14 into the objective of Equation 13 yields the quadratic objective:

$$S^T S + \lambda \begin{pmatrix} S^T & g^T \end{pmatrix} J \begin{pmatrix} S \\ g \end{pmatrix}.$$

For a given g, this objective is a quadratic function of S, whose minimum is achieved at

$$S = -\lambda(I_n + \lambda J_{00})^{-1} J_{01}\, g, \tag{15}$$

where I_n is the identity matrix of length n, and the minimum objective has the form $g^T H g$, where H is a $d \times d$ matrix of the form:

$$H = J_{11} - \lambda J_{01}^T (I_n + \lambda J_{00})^{-1} J_{01}.$$

Therefore, solving the problem of Equation 13 amounts to minimizing $g^T H g$ subject to $g^T g = 1$. By the Rayleigh Quotient Theorem, the minimizer g^* is the eigenvector of H with the smallest eigenvalue. The scalars S^* minimizing Equation 13 can then be obtained from g^* via Equation 15.