Dual Contouring of Signed Distance Data

XIANA CARRERA, Columbia University, USA
NINGNA WANG, Columbia University, USA
CHRISTOPHER BATTY, University of Waterloo, Canada
ODED STEIN, Technion, Israel and University of Southern California, USA
SILVIA SELLÁN, Columbia University, USA

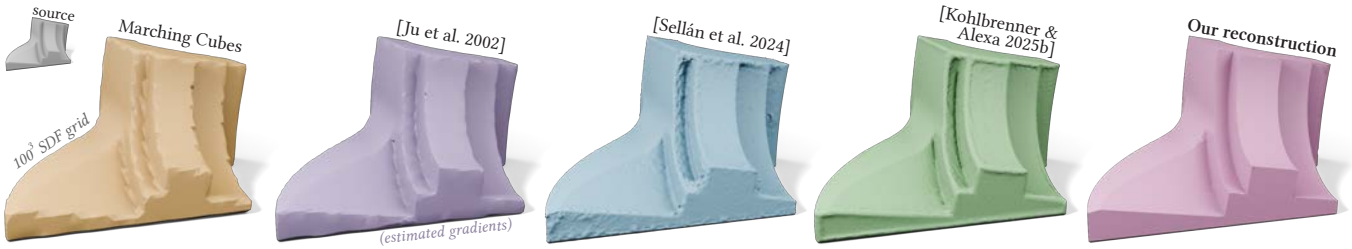

Fig. 1. We propose an algorithm to reconstruct surfaces with sharp features from discrete signed distance data without the need for gradient information.

We propose an algorithm to reconstruct explicit polygonal meshes from discretely sampled Signed Distance Function (SDF) data, which is especially effective at recovering sharp features. Building on the traditional *Dual Contouring of Hermite Data* method, we design and solve a quadratic optimization problem to decide the optimal placement of the mesh's vertices within each cell of a regular grid. Critically, this optimization relies solely on discretely sampled SDF data, without requiring arbitrary access to the function, gradient information, or training on large-scale datasets. Our method sets a new state of the art in surface reconstruction from SDFs at medium and high resolutions, and opens the door for applications in 3D modeling and design.

CCS Concepts: • **Computing methodologies** → **Mesh geometry models**; **Mesh models**.

Additional Key Words and Phrases: signed distance functions, mesh reconstruction, quadratic error functions, sharp features

ACM Reference Format:
Xiana Carrera, Ningna Wang, Christopher Batty, Oded Stein, and Silvia Sellán. 2026. Dual Contouring of Signed Distance Data. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3799902.3811116

Authors' Contact Information: Xiana Carrera, Columbia University, New York, USA, x.carrera@columbia.edu; Ningna Wang, Columbia University, New York, USA, ningna.wang@columbia.edu; Christopher Batty, University of Waterloo, Waterloo, Canada, christopher.batty@uwaterloo.ca; Oded Stein, Technion, Haifa, Israel and University of Southern California, Los Angeles, CA, USA, ostein@usc.edu; Silvia Sellán, Columbia University, New York, USA, silviasellan@cs.columbia.edu.

SIGGRAPH Conference Papers '26, Los Angeles, CA, USA

ACM ISBN 979-8-4007-2554-8/2026/07
https://doi.org/10.1145/3799902.3811116

1 Introduction

Signed Distance Functions (SDFs) measure the distance from any point in space to the boundary of a volume, using the sign to distinguish between the inside and the outside. Their ability to represent shapes with complex sharp features to arbitrary precision and combine them efficiently using Boolean operations have made them popular in applications from computer graphics to engineering, industrial design and manufacturing. In many of these, the SDF of a shape is sampled on a regular volumetric grid during the 3D modeling stage, the first in a larger pipeline with downstream tasks that require explicit surface representations (e.g., simulation). Thus, reconstructing polygonal meshes from discrete SDF samples while preserving geometric features is a critically important task.

Unfortunately, existing reconstruction methods cannot accurately reconstruct a shape's sharp features from a finite set of SDF samples alone. Recent work exploiting the global geometric information contained in every SDF sample has shown major improvements in reconstruction accuracy [Kohlbrenner and Alexa 2025b; Sellán et al. 2024]; however, they rely on smoothness priors that soften a shape's sharp edges and corners (see Figure 1). Indeed, most prior work that succeeds at recovering sharp features does so by placing additional assumptions on the input: data-driven methods use neural networks trained over large datasets to develop powerful priors, while the

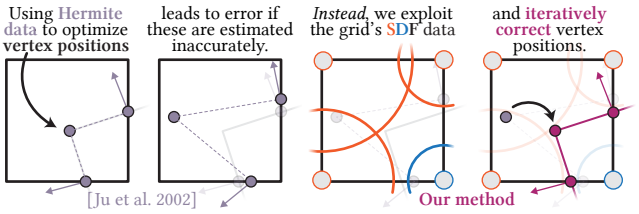

Fig. 2. Our work builds on the Dual Contouring method proposed by Ju et al. [2002]. Instead of relying on exact Hermite information, we estimate it first and progressively update it using discretely sampled SDF data.

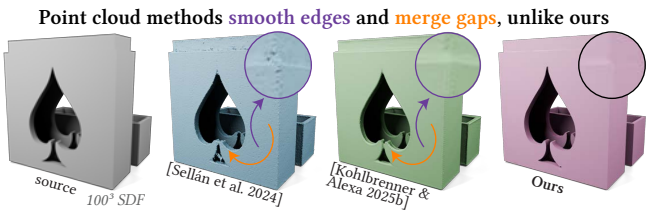

Fig. 3. Competing methods that apply Poisson reconstruction to an intermediate oriented point cloud suffer from over-smoothing. Our method preserves sharp edges and avoids merging close features.

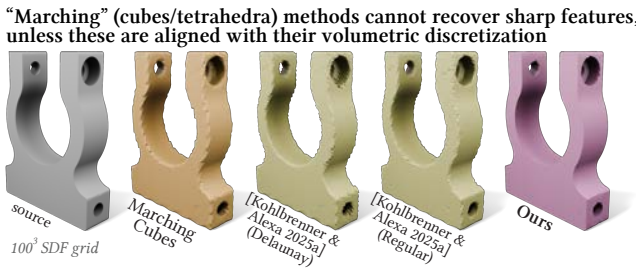

Fig. 4. Competing SDF reconstruction methods that rely on marching cubes or tetrahedra exhibit artifacts near sharp features, except when those features happen to align closely with the underlying elements of the volumetric mesh or grid. Our method exhibits no such bias.

traditional *Dual Contouring of Hermite Data* [Ju et al. 2002] assumes full knowledge of the surface's intersections with the grid edges and its local orientation at those points.

In this paper, we introduce a Dual Contouring algorithm designed to extract surfaces with sharp features from static sampled SDF data only. Like the original work by Ju et al. [2002], we assume that the samples are placed on the nodes of a regular grid, and design and solve a local quadratic energy minimization problem to select the optimal vertex placement within each cell. Unlike the original Dual Contouring, our algorithm does not rely on exact surface normals: instead, it uses the SDF information itself to iteratively refine the mesh's vertex positions (see Figure 2).

In contrast to a recent string of methods that improve reconstruction accuracy through complex optimizations that rely on the global information provided by SDF values away from the surface, our work uses a regular grid to build independent local optimizations that can be solved in parallel. While our method *can* exploit the global geometric information contained in SDF samples far from the zero level set, it is fundamentally agnostic to the amount of SDF data considered by each grid cell, which may be chosen to be higher or lower depending on computational constraints.

Through extensive qualitative and quantitative experiments, we demonstrate our method's superiority to prior work both in the reconstruction of sharp features and as a general reconstruction tool. By phrasing reconstruction as a series of local optimizations and removing the need for intermediary representations like point clouds, our method avoids artificial smoothing and can recover sharp features even when these are not aligned with the background grid (see Figure 1). Our reconstruction strategy approaches the quality provided by the original dual contouring [Ju et al. 2002], but without the need for exact Hermite data. Thus, our method sets a new state of the art in SDF reconstruction at medium and high resolutions, its benefits being most critical in applications in which precision and sharp feature recovery are particularly important, like manufacturing and industrial design.

2 Related Work

SDFs can accurately represent topologically and geometrically complex solid shapes while enabling flexible editing and reasonably fast rendering [Hart 1996]. Because of this, they have served as a useful tool in a myriad of scientific fields: from geology [Hayek et al. 2023] and medical imaging [Esposito et al. 2025] to computational fluid dynamics [Sethian and Smereka 2003], robotic path planning [Liu et al. 2022] and additive manufacturing [Brunton and Rmaileh 2021]. In computer graphics, SDFs have been used to represent the free surface of a fluid [Foster and Fedkiw 2001] or detect collisions between objects [Fuhrmann et al. 2003]; more recently, their representational power has been exploited in geometric deep learning and generative 3D modeling [Coiffier and Béthune 2024; Marschner et al. 2023; Park et al. 2019; Sharp and Jacobson 2022; Takikawa et al. 2021; Wang et al. 2023]. They have even been adopted as the geometric representation of choice by computational design companies like nTop [2019] and Metafold 3D [2026].

Despite the benefits of SDFs, they are ill-suited for many downstream tasks: from finite element simulation to texture mapping and real-time rendering, applications often rely on converting discretely sampled SDF solids into explicit representations like polygonal meshes on which these tasks can be efficiently performed. Thus, reconstructing meshes from SDFs has long been a critical research question in the geometry processing community. In the next two sections, we review the most relevant existing SDF isosurfacing algorithms, starting from those that (like ours) rely only on discretely sampled SDF data as input and ending with those that place additional assumptions on the input.

2.1 Reconstructing discretely sampled SDFs

A vast amount of prior research has been dedicated to the more general problem of reconstructing meshes from arbitrary implicit representations: to borrow the categorization proposed by De Araújo et al. [2015], these methods may opt for region-growing [Hilton et al. 1996], inflation/shrinkwrap schemes [Stander and Hart 1997], or, more commonly, spatial discretizations like regular grids (e.g., *Marching Cubes* [Lorensen and Cline 1987]). As shown by Sellán et al. [2023], these methods are by and large designed for general implicit representations; when applied to SDFs, they are not as accurate as purpose-made algorithms that exploit the distance information contained in each sample. We focus our discussion on the latter and refer the interested reader to the survey by De Araújo et al. [2015] for a comprehensive review of implicit isosurfacing.

Sellán et al. [2023] proposed interpreting global signed distance data as imposing a series of tangency constraints on the reconstructed surface, which they enforce weakly by minimizing an *SDF energy* through a mesh-based geometric gradient flow. This gradient

Fig. 5. The outstanding reconstruction quality of classic Dual Contouring depends on having *exact* Hermite data as input (center left). If approximate (finite difference) estimates are used, quality suffers noticeably (center right). Our method uses the SDF data to optimize for accurate Hermite data, nearly recovering the output quality achieved with exact data (far right).

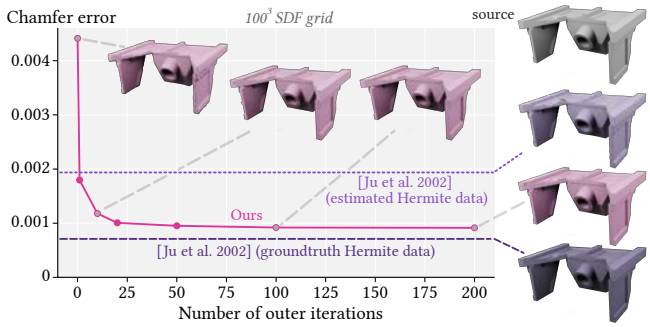

Fig. 6. Our method progressively corrects an estimated set of Hermite data until our reconstruction approaches the quality produced by the original Dual Contouring with groundtruth Hermite data.

flow includes a local remeshing step that smooths a surface's sharp features; more critically, it encounters singularities and fails to converge if the shape's topology is complex or not known beforehand.

To avoid singularities and support arbitrarily complex topologies, Sellán et al. [2024] proposed using SDF data to construct an intermediate point cloud representation, which can then be passed to an off-the-shelf point-based reconstruction algorithm [Kazhdan et al. 2006; Kazhdan and Hoppe 2013]. The crux of the method is in the positioning of the points: while Sellán et al. [2024] propose a complex optimization involving rasterization and iterative refinement, the follow-up work by Kohlbrenner and Alexa [2025b] uses Lie geometry to produce a valid, more regular point cloud. Regardless of how the point cloud is computed, these methods inherit the smoothness prior of the specific point cloud reconstruction algorithm, which causes smoothed corners and merged gaps (see Figures 1 and 3).

Exploiting the same insight, Kohlbrenner and Alexa [2025a] use SDF data to construct a volumetric regular tetrahedralization, from which an isosurface can be extracted using marching tetrahedra. More simple and computationally efficient than point and flow-based algorithms, it outperforms other spatial subdivision methods like Marching Cubes by grouping resolving power closer to a shape's important features. However, the edges of the regular tetrahedralization are not guaranteed to align with a shape's sharp features, leading to smoothed and chamfered corners (see Figure 4).

Our method builds on the observations underpinning recent methods for reconstruction from discretely sampled SDFs, addressing some of their main limitations. We extract the mesh's topology from a regular grid, avoiding singularities and unstructured remeshing. We do not rely on any intermediate representation, thus removing the need for smoothness priors (see Figure 3) and superlinear complexity. Finally, we use a dual spatial discretization, recovering sharp features irrespective of their grid alignment (see Figure 4).

2.2 Dual Contouring of Hermite Data

Mathematically, our method is most related to and greatly inspired by the *Dual Contouring of Hermite Data* algorithm proposed by Ju et al. [2002]. When combined with follow-up work improving stability and geometric properties [Ju and Udeshi 2006; Schaefer et al. 2007; Schaefer and Warren 2002; Trettner and Kobbelt 2020], it produces impressively accurate reconstructions with close-to-perfectly recovered sharp features (see Figure 5, left).

Its effectiveness is in part thanks to the format of its input: it requires exact knowledge of both the positions at which the grid edges intersect the unknown surface and the local normal directions there (from here onwards, we refer to such point-normal pairs as *Hermite data*). While this data can be extracted from a signed distance function if one has the ability to query the SDF at arbitrary spatial positions (e.g., through bisection along each edge combined with finite difference gradient estimation), it can only be loosely approximated if the SDF's values are only known on a discrete set of locations. If this estimation is inaccurate (as is often the case near an object's sharp features), the method loses its ability to reconstruct sharp features (see Figure 5, center-right).

Like Ju et al. [2002], we propose a contouring strategy that optimizes vertex placements within each cell using a quadratic error function (QEF). However, we do so without requiring exact Hermite data or access to the SDF function, and instead rely only on a discrete set of SDF samples. We exploit the information contained in these samples to iteratively correct an incorrect initial set of Hermite data, eventually recovering sharp features with nearly the same quality achieved by Ju et al. [2002], but without placing restrictive assumptions on the input (see Figure 6).

2.3 Neural SDF reconstruction

Neural techniques have had significant recent influence on isosurface extraction of SDF data. It is helpful to disambiguate between two different tasks that are nonetheless often referred to in the literature by the same name. Some methods are dedicated to meshing *continuous* SDFs represented (perhaps approximately) by neural networks [Binninger et al. 2025; Hwang and Sung 2024; Lei and Jia 2020; Liu et al. 2025; Remelli et al. 2020; Shen et al. 2023a; Stippel et al. 2025; Xie et al. 2022], while others concern themselves with learning data-driven priors to mesh *discretely sampled* SDFs.

The latter category of methods is most relevant to our setting. Neural Marching Cubes [Chen and Zhang 2021] and Neural Dual Contouring [Chen et al. 2022] update their classic counterparts by learning the placement of vertices from datasets of meshes, and using wider stencils of SDF data for inference compared to the classic methods. In Neural Dual Contouring, the role of the QEF is entirely replaced by a learned network, obviating the need for

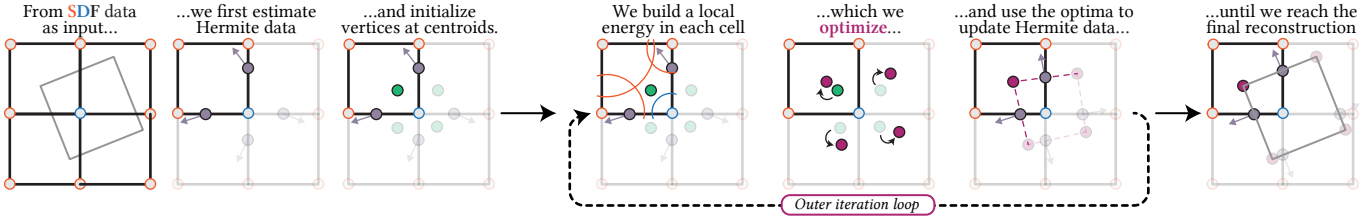

Fig. 7. An overview of the outer optimization loop of our method. After estimating Hermite data at each interesting edge, an initial configuration is calculated and progressively refined through local per-cell optimizations and global updates to the Hermite data.

Hermite input. Rather than replacing the QEF, the PoNQ method [Maruani et al. 2024] proposes a point-based surface representation augmented with QEF data. They learn to generate the PoNQ data using a network that accepts as input the entire N^3 SDF grid. Surface reconstruction is then performed using a Delaunay-based approach that tetrahedralizes the optimal QEF points (analogous to Dual Contouring vertices) and tags each tetrahedron as interior or exterior with a graph-cut strategy, ensuring a watertight and non-self-intersecting surface. PoNQ's predecessor VoroMesh [Maruani et al. 2023] uses a related Voronoi-based strategy, but without QEF data. As with any learning-based method, these approaches require significant training time over large data sets, and their results will be dependent on the characteristics of the training data.

2.4 Quadratic Error Functions in Computer Graphics

To produce accurate surface reconstructions with sharp feature edges, we design a tailor-made quadratic optimization problem that is solved to obtain the optimal vertex placement inside each grid cell. This puts our method, like the original by Ju et al. [2002], in the company of a vast array of geometry processing algorithms. Indeed, quadratic error functions have been used for tasks from mesh simplification [Garland and Heckbert 1998] and approximation [Thiery et al. 2013] to surface fairing [Legrand et al. 2019] and building simulation cages [Deng et al. 2011].

More relevant to our work, quadric error metrics have been used to reconstruct surfaces from signed [Liu et al. 2025] and unsigned [Zhang et al. 2023] distance data. While these methods produce impressive results, even in the presence of noise, they do so by exploiting the ability to sample the distance function at arbitrary spatial positions, enabling them to build the optimal quadratic error function. Our work considers a related but critically different task: namely, that of reconstructing surfaces with sharp features from discrete SDF samples *alone*, without access to more data or the ability to query the SDF function at additional locations.

3 Method

Our input is a uniform, regular grid equipped with the values $s_1, \ldots, s_n \in \mathbb{R}$ measuring the signed distance from an unknown surface Ω to each of the grid's vertices $\mathbf{u}_1, \ldots, \mathbf{u}_n \in \mathbb{R}^3$. In this section, we will introduce an iterative algorithm to compute a quad mesh $\mathcal{M} = (\mathcal{V}, \mathcal{Q})$ that approximates Ω.

As is customary in grid-based SDF reconstruction algorithms (e.g., [Ju et al. 2002; Lorensen and Cline 1987]), we begin by identifying the grid's *interesting* edges, i.e., those whose vertices differ in sign

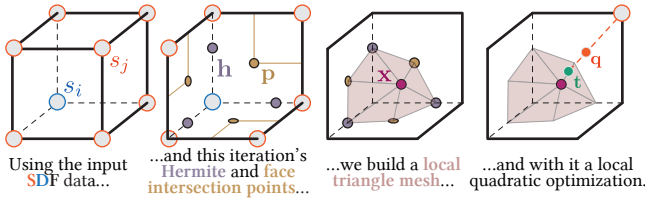

Fig. 8. In each iteration, we use the Hermite (edge) points and face intersection points to build a local mesh inside each cell, whose interior vertex we optimize to align with SDF values assigned to the cell.

and are thus known to intersect the true surface Ω. We then tag all cells containing any such edge as *interesting cells*, the set of which we will denote $C = c_1, \ldots, c_m$.

Following the contouring strategy proposed by Ju et al. [2002], we will now phrase the reconstruction problem as that of optimizing the placement of a single vertex $\mathbf{x}_i$ in each interesting cell c_i. Once the ideal vertex positions $\mathbf{x}_1, \ldots, \mathbf{x}_m$ have been found, the four vertices corresponding to the four cells containing each interesting edge can be joined together to form our mesh's quads $\boldsymbol{Q}$.

3.1 Initialization

To compute an initial guess for each cell's optimal vertex placement, we begin by looping over all interesting edges $e_i = [\mathbf{u}_{ia}, \mathbf{u}_{ib}]$. Given the difference in sign between the two edge's SDF values, linear interpolation allows us to estimate the point along the edge that intersects with the surface Ω,

$$\mathbf{h}_i^0 = (1-t)\mathbf{u}_{ia} + t\mathbf{u}_{ib}\,, \quad \text{where} \quad t = |s_{ia}|/(|s_{ia}| - |s_{ib}|)\,. \tag{1}$$

We then differentiate the trilinear interpolant to construct gradients over each interesting cell, $\nabla_{\text{tri}}(c_1, \mathbf{x}), \ldots, \nabla_{\text{tri}}(c_m, \mathbf{x})$. We evaluate the gradient at the Hermite point of edge e_i for each cell containing the edge and average them to estimate the Hermite *normal*:

$$\mathbf{n}_i^0 = \frac{\sum_{e_i \in c_j} \nabla_{\text{tri}}(c_j, \mathbf{h}_i^0)}{\left\| \sum_{e_i \in c_j} \nabla_{\text{tri}}(c_j, \mathbf{h}_i^0) \right\|}. \tag{2}$$

With this estimated information, one could directly apply the original Dual Contouring algorithm to obtain a possible reconstruction. In practice, we note that this strategy (which we refer to as *[Ju et al. 2002] (estimated Hermite data)* throughout the paper) fails to reconstruct the shape's features, as the estimates $\{(\mathbf{h}_i^0, \mathbf{n}_i^0)\}$ are least accurate near sharp gradient discontinuities (see Figure 5). Nonetheless, one may imagine that the vertex positions produced by their

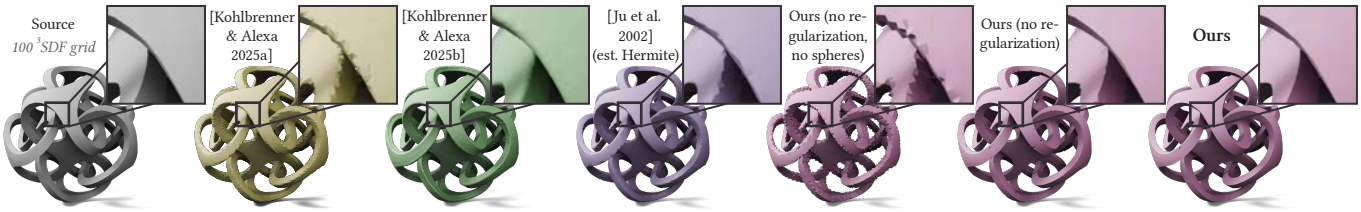

Fig. 9. To guide convergence in a highly non-convex setting, our algorithm incorporates two regularization terms in our linearized inner loop, one based on the estimated Hermite normals and the other a standard L_2 regularizer.

algorithm can serve as a helpful initialization for our method; unfortunately, this is not the case, as they often consist of irregular shapes with self-intersections that in practice can trap our iterative reconstruction in undesirable local minima.

Instead, we opt for a more regular initialization strategy. Once this first estimate for each edge's Hermite information has been calculated, we compute each interesting cell's *centroid* by averaging all Hermite points contained in its edges:

$$\mathbf{c}_i^0 = \frac{\sum_{\mathbf{h}_j^0 \in c_i} \mathbf{h}_j^0}{|\{\mathbf{h}_j^0 \in c_i\}|}. \tag{3}$$

We then initialize each cell's vertex to its centroid: $\mathbf{x}_i^0 = \mathbf{c}_i^0$. In the rest of this section, we will proceed iteratively to improve the positioning of this vertex. In particular, given the input SDF values and a prior iteration's cell vertices $\mathbf{x}_i^k$, we will construct an energy minimization problem whose solution provides an updated position $\mathbf{x}_i^{k+1}$. We refer to the repetition of this process as our algorithm's *outer* iterative loop, which we discuss in the next section.

3.2 Outer iterative loop

Given a prior iteration's vertex positions $\mathbf{x}_i^k$, we connect the four cell vertices corresponding to each interesting edge to form a quad mesh, which we refer to as this iteration's *global* mesh $\mathcal{M}^k$. Inspired by a long line of work from the geometry processing community proposing local-global optimization strategies [Sorkine and Alexa 2007], our goal now is to use the information contained in this mesh and the input SDF data to construct a *local* energy minimization problem that can be solved in parallel to obtain each cell's new vertex position.

Making each cell's optimization problem depend on all the input SDF data would be computationally prohibitive. Because of this, we start by arbitrarily triangulating the global mesh and computing the closest point on the mesh to each of the grid nodes $\mathbf{u}_j$. We identify which of the grid cells contains this closest point, and *assign* the tuple $(\mathbf{u}_j, s_j)$ to said cell. For robustness to outliers, we do not assign any sphere whose distance to the closest point is larger than the sum of the cell diagonal and $|s_i|$.

We will now quantify how much a given choice of cell vertex aligns with the SDF values $(\mathbf{u}_j, s_j)$ assigned to the cell. To achieve this, we will use the current iteration's vertex $\mathbf{x}_i$ and Hermite data $\mathbf{h}^k$ and construct a *local* triangle mesh that approximates the reconstruction's geometry at c_i without introducing explicit dependencies between cells (thus maintaining locality and parallelization).

By construction, the global mesh's edges will cross the planes of the interesting cell's faces: we compute and store these intersections or *face intersection points* $\mathbf{p}^k$. Each $\mathbf{p}^k$ is spawned by a specific quad in the mesh, and each quad corresponds to one of the grid's interesting edges. That edge has itself an associated Hermite point $\mathbf{h}^k$, which we say is $\mathbf{p}^k$'s *relevant Hermite point*. A cell's local triangle mesh $\mathcal{M}_i^k$ can then be constructed by joining together every face intersection point, its associated Hermite point, and the cell's vertex $\mathbf{x}$ (see Figure 8, left), inspired by the approach used by Occupancy-Based Dual Contouring [Hwang and Sung 2024].

The extent to which $\mathcal{M}_i^k$ agrees with the cell's assigned SDF data can be measured in terms of the local *distance energy*

$$E_d^{k,i}(\mathbf{x}) = \sum_{(\mathbf{u}_j, s_j) \in \mathcal{A}(c_i)} \left(|s_j| - d\left(\mathbf{u}_j, \mathcal{M}_i^k(\mathbf{x})\right)\right)^2 \tag{4}$$

where $\mathcal{A}(c_i)$ is the set of $(\mathbf{u}_j, s_j)$ tuples assigned to the cell c_i, and d measures the Euclidean distance from a point to a triangle mesh.

This energy is inspired by the *SDF energy* proposed by Sellán et al. [2023]; however, because it is applied to an open local mesh as opposed to the full reconstruction, it simply ignores the distance's sign. Because our method uses a regular grid discretization that only locally updates each cell's vertex position, our output topologically separates the positive SDF data from the negative one by construction, removing the added complexity of a signed energy treatment. By contrast, the method of Sellán et al. [2023] relies on a global geometric flow that optimizes an entire mesh starting from a large, encompassing initial guess (e.g., a sphere); it must therefore account for the distances' signs to ensure SDF samples $(\mathbf{u}_j, s_j)$ end up on the correct side of the surface. This need poses an additional challenge in their energy minimization, requiring a purpose-made metric of sign agreement based on the surface's normals.

Directly minimizing our distance energy will produce highly irregular solutions (see Figure 9): intuitively, this is because the union of the local meshes $\mathcal{M}_i^k$ is only a valid approximation of the global mesh for reasonably small displacements in $\mathbf{x}$. In particular, minimizing Equation 4 separately for all the vertices in a single quad may lead to optimal local meshes that are artificially bent in ways that are impossible to recreate with a single quadrilateral. Thus, we also include the *Hermite energy* (c.f. Ju et al. [2002]):

$$E_H^{k,i}(\mathbf{x}) = \sum_{e_j \in c_i} \left((\mathbf{x} - \mathbf{h}_j^k) \cdot \mathbf{n}_j^k \right)^2. \tag{5}$$

In our setting, this energy plays a dual role. First, given that each Hermite tuple $\{(\mathbf{h}_j^k, \mathbf{n}_j^k)\}$ is shared between all cells that will form

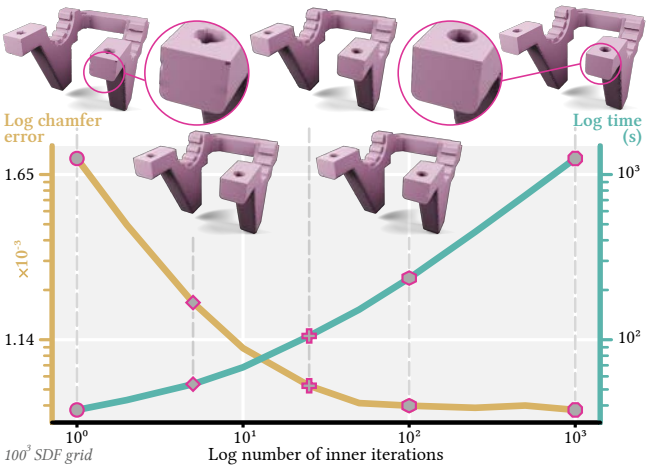

Fig. 10. Increasing the number of inner iterations progressively reduces the reconstruction error, eliminating local artifacts and improving feature fidelity until accuracy gains saturate, while runtime grows roughly linearly.

a quad together, it encourages a consistent normal orientation, ensuring that the distance energy is not satisfied through artificial bending in the local mesh. Crucially, this reduces the importance of the choice of triangulation in later steps of our method (L11 in Algorithm algorithm 1). Second, since our outer iterations are also designed to progressively improve the Hermite data itself, this energy helps mimic the sharp edge recovery behavior that Ju et al. [2002] achieved through exact Hermite data. The combination of Equations 4 and 5 lead to our total *local energy*

$$E^{k,i}(\mathbf{x}) = E_d^{k,i}(\mathbf{x}) + w_H^2 E_H^{k,i}(\mathbf{x}), \tag{6}$$

where w_H is a weight balancing the effect of the two terms (see Figure 20 for an ablation study on the effect of this parameter). In subsection 3.3, we will show how we turn this energy into an iterative quadratic problem (our *inner iterative loop*), enabling us to solve it in parallel for all cells during each outer iteration.

Once this energy has been optimized for every cell to obtain the optimal next vertex positions $\mathbf{x}_i^{k+1}$, our next and final step is to update the Hermite information in each interesting edge. For every four cells sharing a Hermite edge, we use Principal Component Analysis on the associated four vertices to obtain their best-fit plane, normal $\mathbf{n}$, and intersection with the edge $\mathbf{y}$. Then, we update the Hermite information at that edge using linear interpolation:

$$\mathbf{h}_j^{k+1} = \mathbf{h}_j^k + w_u(\mathbf{y} - \mathbf{h}_j^k), \quad \mathbf{n}_j^{k+1} = \frac{\mathbf{n} + w_u \mathbf{n}_j^k}{\|\mathbf{n} + w_u \mathbf{n}_j^k\|}, \tag{7}$$

where our *update weight* $w_u \in [0, 1]$ provides a tradeoff between stability and speed of convergence (see Figure 11 for an ablation). In the last iteration, we join together the final cell vertices $\mathcal{V} = \mathbf{x}_1, \ldots, \mathbf{x}_m$ to form our output quad mesh $\mathcal{M} = (\mathcal{V}, \mathcal{Q})$.

3.3 Inner iterative loop

We now describe how we minimize the local energy E^k for each cell in each iteration k of our outer loop (for clarity, we omit the

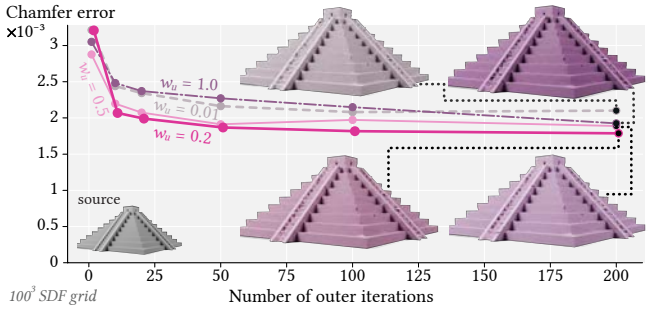

Fig. 11. Increasing the update weight w_u may not always improve accuracy when the number of outer iterations is low, but choosing very conservative values can make precision saturate too soon. We find intermediate values in the range 0.2-0.8 to generally yield the best results.

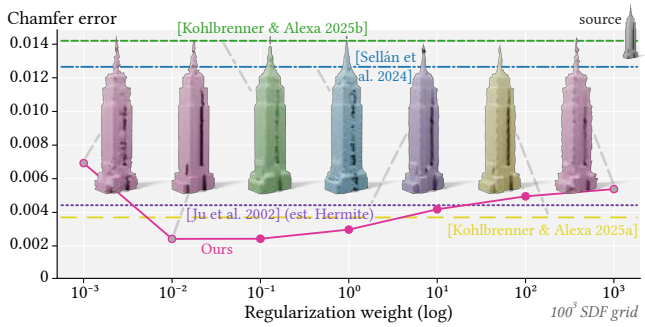

Fig. 12. *Start spreading the μ's*: our algorithm benefits from inter-iteration regularization; in practice, we recommend values of $\mu \in [10^{-2}, 10^{-1}]$.

superscript k from here onwards), a highly non-convex optimization problem difficult to optimize for directly with any guarantees. Like Sellán et al. [2023], we begin by interpreting each pair of SDF data $(\mathbf{u}_j, s_j)$ as a sphere centered at $\mathbf{u}_j$ with radius $|s_j|$ to which the local mesh must be tangent. Letting $\mathbf{t}_j$ be the closest point to $\mathbf{u}_j$ on the local mesh $\mathcal{M}_i$ and $\mathbf{q}_j$ the closest point to $\mathbf{t}_j$ on the surface of the sphere, one can linearize each term in the distance energy as

$$\left(|s_j| - d\left(\mathbf{u}_j, \mathcal{M}_i\right)\right)^2 \approx \left\|\mathbf{t}_j - \mathbf{q}_j\right\|^2. \tag{8}$$

This linearization is sufficient in global optimization settings, in which the interaction between different mesh vertices and even remeshing can aid one in escaping local minima [Sellán et al. 2023].

However, it is not enough in our local setting, in which a cell's vertex may be prevented from approaching an unsatisfied sphere because sliding tangentially with respect to others is penalized. Figure 14 gives an example in which the cell's vertex $\mathbf{x}$ cannot move towards $\mathbf{q}_3$ because it would cause $\mathbf{t}_1$ and $\mathbf{t}_2$ to slide away from their respective $\mathbf{q}_i$. To avoid this, we will only measure distance in the direction towards the sphere's center, and not tangential to it. Letting $\mathbf{d}_j$ be the (normalized) vector from $\mathbf{u}_j$ to $\mathbf{q}_j$, the distance energy becomes

$$\left(|s_j| - d(\mathbf{u}_j, \mathcal{M}_i)\right)^2 \approx \left((\mathbf{t}_j - \mathbf{q}_j) \cdot \mathbf{d}_j\right)^2. \tag{9}$$

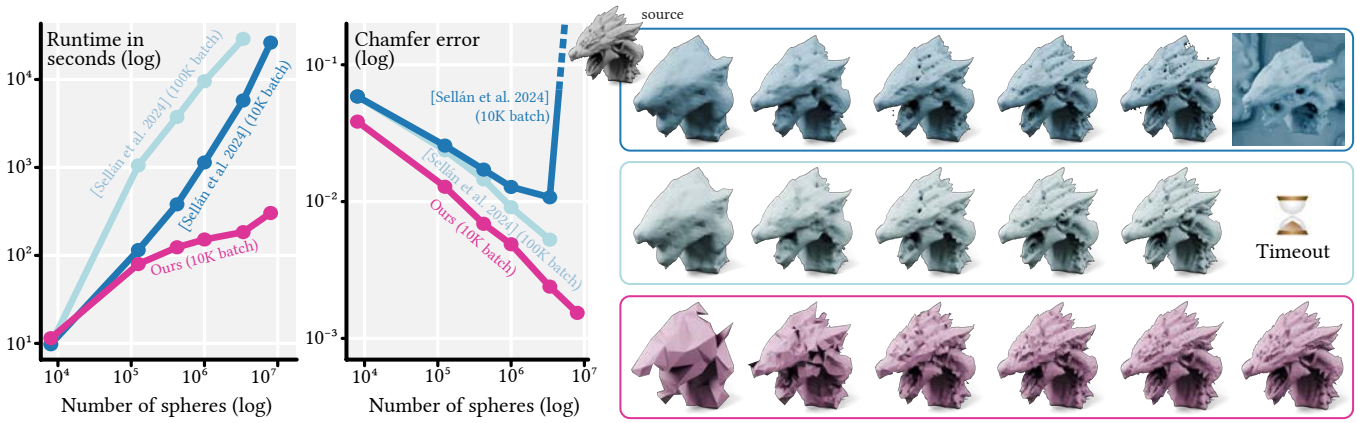

Fig. 13. We compare our method (pink) against Sellán et al. [2024] using their default batch size (10K, blue) for the number of spheres being considered, and an increased batch size (100K, cyan), which rapidly makes computational costs intractable. Not using a high enough batch size with their method can cause parts of the surface to be generated far away from the ground truth, fully enclosing the reconstruction (top row, rightmost).

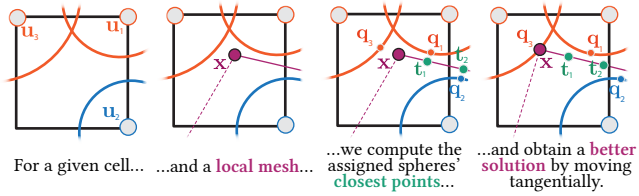

Fig. 14. An example in which allowing tangential displacements and penalizing only the radial distance toward the spheres' centers yields a more accurate vertex position.

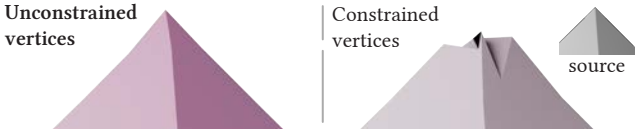

Fig. 15. Allowing vertices to move outside their corresponding cells helps reconstruct particularly sharp features.

Formally, this amounts to a first-order Taylor expansion of the distance in Equation 8. Writing $\mathbf{t}_j$ in terms of its barycentric coordinates in the local triangle mesh allows us to write the right hand side of the above approximation as a quadratic energy in $\mathbf{x}$:

$$\tilde{E_d}(\mathbf{x}) = \|A_d\mathbf{x} - b_d\|^2 \tag{10}$$

where A and b concatenate the contribution of each relevant tuple $(\mathbf{u}_j, s_j)$. To this linearized distance energy, we add the already-quadratic Hermite energy from Equation 5. We will minimize them together, starting from the previous iteration's cell vertex $\mathbf{x}^k = \mathbf{x}^{k,0}$ and iteratively updating the positions $\mathbf{t}_j, \mathbf{q}_j$ using each subsequent minimizer. Finally, we include a regularization term to compensate for the aggressiveness of our linearization:

$$\mathbf{x}^{k,r+1} = \arg\min_{\mathbf{x}} \tilde{E_d}^{k,r}(\mathbf{x}) + w_H^2 E_H^k(\mathbf{x}) + \mu\|\mathbf{x} - \mathbf{x}^{k,r}\|^2. \tag{11}$$

Whenever the distance between one $\mathbf{x}^{k,r}$ and the next $\mathbf{x}^{k,r+1}$ is lower than a chosen numerical threshold τ, or when a maximum number of iterations is reached, the algorithm returns to the outer loop with an updated $\mathbf{x}^k$. While we experimented with constraining $\mathbf{x}^{k,r+1}$ to always be inside the cell, we found that allowing it (and the face intersection points $\mathbf{p}$) to escape the cell helped with sharp feature recovery (see Figure 15).

4 Experiments & Results

Implementation. We developed our method in C++ and Python, using libigl [Jacobson et al. 2025] and Gpytoolbox [Sellán and Stein 2025] for common geometric processing subroutines (e.g., AABB trees for point-to-mesh distances) and OpenMP to perform our vertex optimizations in parallel for each cell. We report runtimes on a Macbook Pro laptop with an M4 Pro chip and 48 GB of memory, and render our results using BlenderToolbox [Liu 2023]. Our comparisons to the methods of Sellán et al. [2023], Sellán et al. [2024], Kohlbrenner and Alexa [2025b] and Kohlbrenner and Alexa [2025a] rely on the authors' official implementations. Our method's code is publicly available online[1], and we provide pseudocode in the **Supplementary Material**.

Experiments. Our algorithm's runtime is dominated by the quadratic optimization steps in our inner loop, whose computational cost scales linearly with the size of the input data. Fortunately, our algorithm is agnostic to the structure of this input: at low resolutions, one may want to exploit all the information contained in a regular grid (e.g., in Figure 16); at higher ones with more redundancy, one may wish to use only a narrow band around the zero level set (see Figure 17). If one wishes to exploit all the data in the input even in higher resolution settings, we can randomly batch the input data considered in each outer iteration: in Figures 13 and 19, we show that this strategy improves asymptotic complexity with minimal effect on accuracy. In all experiments, we set a maximum batch

[1] https://gatc.cs.columbia.edu/projects/dual-contouring-of-signed-distance-data.html

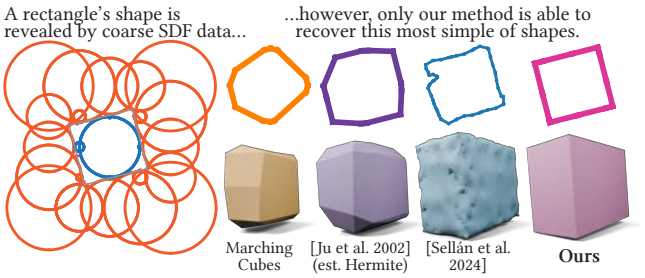

Fig. 16. Even at very low resolution (5^3), the sphere visualization of this SDF makes the shape of a source rectangle apparent (left). Our method easily recovers the rectangle (top row), and its counterpart 3D cuboid (bottom row), while other methods fail.

size of 200,000. As expected, our method's complexity also scales linearly with the number of inner and outer optimization iterations performed: in Figures 6 and 10, we show that diminishing returns are observed around 100 inner and outer maximum iterations, which we set as our default values.

Beyond its runtime, several other parameters affect our method's performance. As shown in Figure 20, it is most sensitive to the choice of w_H, which we fix to 0.02 in all our experiments. The two other major parameters of our algorithm are the update weight w_u and the regularization μ: in Figure 11 and Figure 12, we study the effect of these parameters, whose defaults we set to 0.2 and 0.1 respectively. Every result in our paper includes a randomly generated rotation, to control for the effect of axis alignment.

Comparisons & Applications. We carry out several quantitative and qualitative experiments to test the performance of our method, beginning with the one that inspired this research. In Figure 16, we attempt a deceptively simple task: to reconstruct a (2D or 3D) box from a coarse, discrete set of SDF samples alone, without differential or Hermite information. Surprisingly, no previously existing algorithm managed to successfully produce sharp corners, regardless of size and orientation (unless it perfectly aligns with the background grid). We used this example during the development of our algorithm; in its final form, it recovers the perfectly sharp box.

To evaluate our method in a setting closer to its real-world applications, we considered the randomly selected subset of the ABC dataset [Koch et al. 2019] released by Xu et al. [2024], and used it to produce SDF samples at resolutions 50^3, 100^3, 150^3 and 200^3. We pass these samples as input to our method as well as Marching Cubes [Lorensen and Cline 1987] and the methods of Ju et al. [2002] (with estimated Hermite data), Sellán et al. [2023], Sellán et al. [2024], Kohlbrenner and Alexa [2025a], and Kohlbrenner and Alexa [2025b]. As proposed by these authors, we quantify Hausdorff and chamfer error between groundtruth and output, as well as the SDF energy introduced by Sellán et al. [2023]. To evaluate accuracy in sharp feature recovery, we also use the *edge* chamfer error proposed by Chen and Zhang [2021].

We present our results in Table 2 (**Supplemental**) and Figure 21, which reveal our method to have the highest accuracy across resolutions and almost all metrics. Surprisingly, the second best scores are often achieved by the original Dual Contouring method with

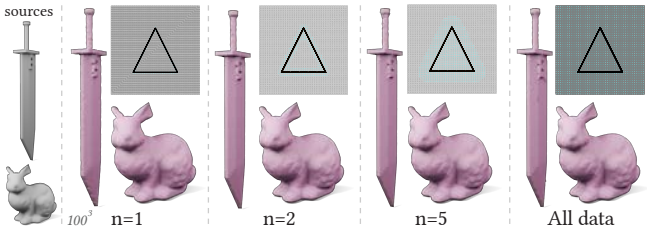

Fig. 17. Our algorithm successfully reconstructs the input even when using a limited amount of information. Here, we only consider the SDF data available at grid vertices whose distance from the ground truth is smaller than the length of the diagonal of a cell grid scaled by n.

Fig. 18. Sweeping a rigid rectangular plate along a swirling trajectory yields a complex shape with numerous sharp features. Our method produces the best reconstructed mesh compared to competing schemes.

estimated Hermite data, revealing that the high reported accuracy of prior point-cloud based work (e.g., [Kohlbrenner and Alexa 2025b; Sellán et al. 2024]), which was tested mostly on smooth organic shapes, may not carry over to cleaner CAD geometry with sharp features. In Figure 3, we show a qualitative result illustrating the failure cases of these algorithms. At the highest resolutions, *Reach for the Arcs*' [Sellán et al. 2024] runtime made it impracticable, and the method proposed by Kohlbrenner and Alexa [2025b] encountered crashes for a large subset of shapes. Therefore, at these resolutions, we compare only to the method by Kohlbrenner and Alexa [2025a], which our algorithm outperforms. In Figure 4, we qualitatively compare to this method, which fails to recover precise sharp features.

Our method's impressive performance on this dataset reveals that it is most accurate (in relation to other methods) in medium-to-high resolution settings, and on the broadly smooth shapes with sharp gradient discontinuities that dominate the ABC dataset. To test this intuition, we evaluate our method on the ten artist-designed smooth geometries used as a baseline by Sellán et al. [2024], and report the results in Table 3. The results of this specific dataset without sharp features are more mixed, varying based on resolution and metric. In rough terms, we observe our algorithm to perform on par with the best of the previously existing ones.

Our method will be most impactful in applications in which one wishes to reconstruct shapes with sharp features at a high resolution and it is impossible or exceedingly costly to estimate exact gradient information or densely sample the SDF. In Figure 18, we show one such example, in which each SDF sample is painstakingly computed

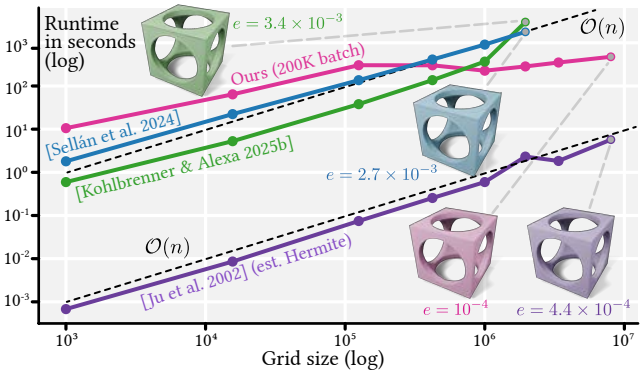

Fig. 19. While prior point-based methods scale superlinearly or linearly with the size of the SDF data (even the work by Sellán et al. [2024], which includes constant-size batching), our random asymptotic complexity is bounded by the size of the batch.

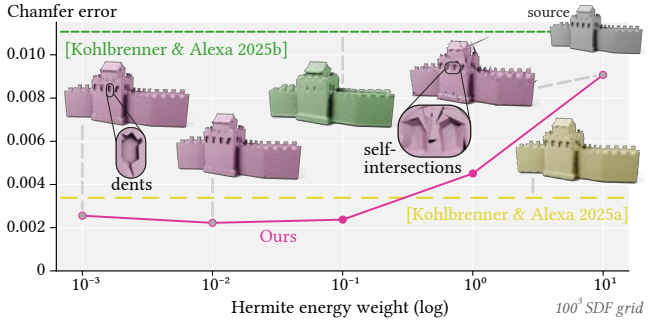

Fig. 20. Effect of the Hermite energy weight w_H on the fidelity of our reconstruction. Large values make the method unstable and cause spikes to appear. We set a default value of 0.02.

by taking the minimum over tens of thousands of positions along an object's trajectory. In this context, our method recovers sharp edges without the need for binary search of gradient computation, considerably speeding existing swept volume computation frameworks [Sellán et al. 2021]. For this result, we follow the lead of Sellán et al. [2023] and do not impose tangency constraints for the *pseudoSDF* samples on the inside of the swept volume [Marschner et al. 2023].

Limitations & Conclusion. We have introduced an algorithm to reconstruct polygonal meshes from discretely sampled signed distance functions. Building on the original work by Ju et al. [2002], we proposed a Dual Contouring strategy that begins by estimating Hermite information on a grid's edges and progressively corrects it by exploiting the information contained in the SDF data. Unlike all prior work on this task, our algorithm is able to faithfully reconstruct a shape's sharp edges through the SDF information, without the need for gradient information or data-driven priors.

Our method inherits some limitations from the original Dual Contouring. For example, we recover sharp features by optimizing the position of a vertex associated with each of the grid's cells. Often, the most accurate reconstruction relies on this vertex escaping outside the bounds of the cell, which can cause self-intersections and face flips (see Figure 20). Similarly, as we study in Table 1, our method inherits the original Dual Contouring's sensitivity to noise.

To obtain accurate reconstructions efficiently, our method relies on a heavily non-convex energy, whose minimizer is only approximated numerically through a local-global iterative optimization. In challenging cases, this approximation can yield isolated dents along feature curves (see Figure 20). We conjecture that they might be alleviated through a more complex global mesh update that enforces sparsified regularity in each iteration's Hermite data.

After optimizing the vertex corresponding to each cell, we connect vertices in adjacent cells to output a final quad mesh. We made this algorithmic choice in the interest of simplicity, and to draw a clear analogy between our method and the original work by Ju et al. [2002]. However, our work could likely benefit from orthogonal-yet-complementary improvements to the original Dual Contouring, e.g., by enforcing manifold outputs [Schaefer et al. 2007], which would not require major fundamental changes to our method. More interestingly, our algorithm's outer loop relies on triangulating the prior iteration's quads to build the next optimization problem. While we followed an arbitrary strategy (taking the first diagonal), future work may consider using the SDF data to optimize the choice of triangulation itself as suggested recently by Shen et al. [2023b]. Similarly, while our implementation assumed signed distance samples to lie on a regular grid, future work may consider extensions of our method to adaptive data structures like octrees.

Finally, this paper sits on the shoulders of recent work that has reawakened our community's interest in surface reconstruction from discrete SDF samples. By focusing our study on the recovery of geometric sharp features, we have contributed to this growing area of research that has revealed itself to possess both impressive mathematical richness and practical importance. As the literature on this topic continues to expand, we also hope to see future work dedicated to building datasets, baselines, and open challenges that can help guide SDF reconstruction towards a healthy maturity.

Acknowledgments

The Geometry and the City lab at Columbia University is supported by generous gifts from nTop, Adobe, Dandy, and Braid Technologies, as well as by a sponsored research project from Dreamsports and the Columbia Engineering Interdisciplinary Research Fund. The third author acknowledges the generous support from the Natural Sciences and Engineering Research Council of Canada (Grant RGPIN-2021-02524). The fourth author acknowledges the generous support from the National Science Foundation (award #2335493) and a gift from Adobe.

We thank the authors of the 3D models used throughout this paper for making them available for academic use. Figures in this work contain the playing card box [SometimesCake 2012], mechanical toy piece [Ellindsey 2013], metatron [Bathsheba Grossman 2012], Empire State [demonofthenorth 2014], Great Wall of China [MJCbull 2012], dragon head [LosoPower 2022], bunny [The Stanford 3D Scanning Repository 1994], sword [BlyncStudio 2015], dualstrusion ball in cube [etrohn 2012], Chichen Itza [ChapinEagle02 2015], and car piece [barspin 2013].

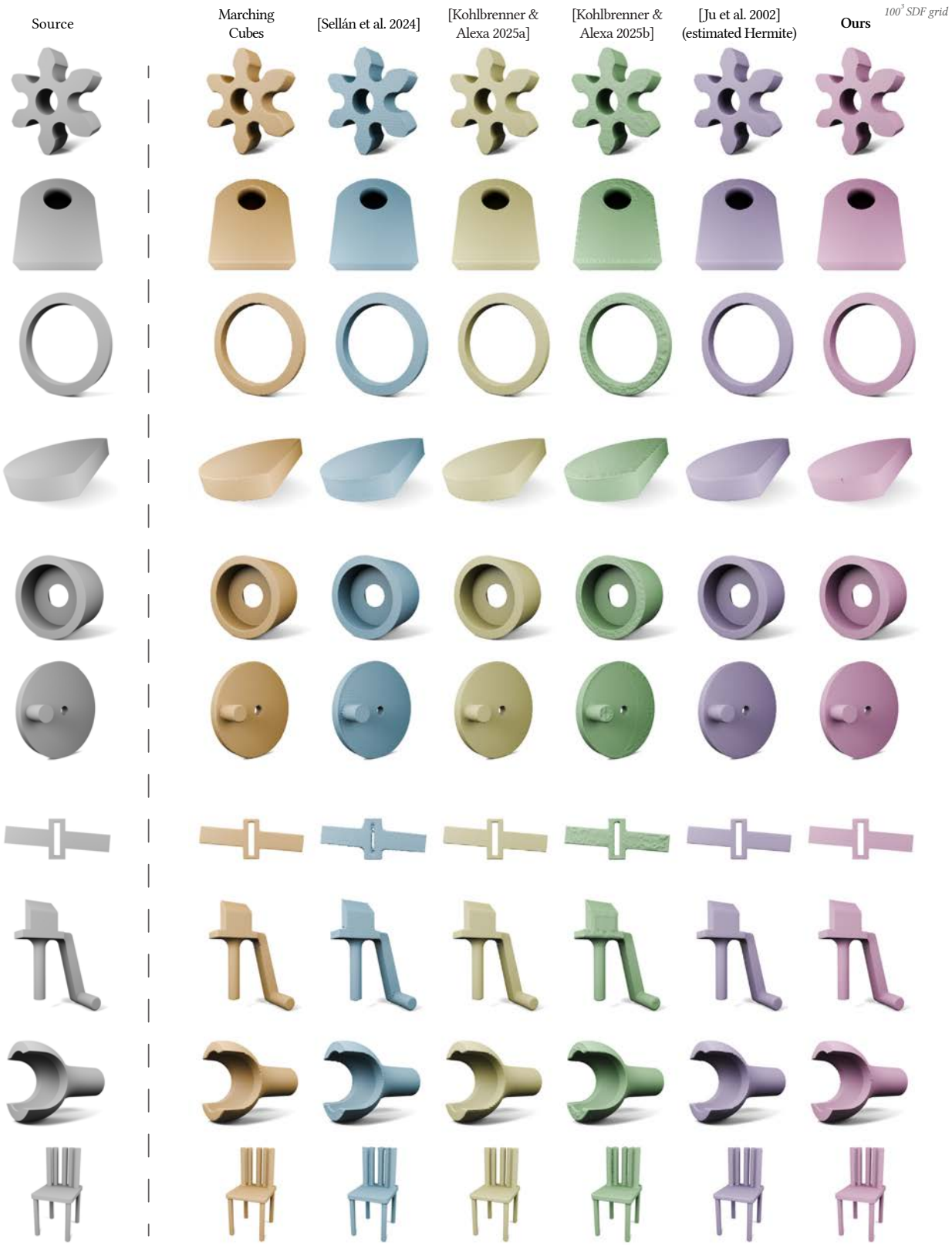

Fig. 21. A comparison against other methods on a subset of the ABC dataset, at a resolution of 100^3.

References

barspin. 2013. Thingi10K model 1458674. https://ten-thousand-models.appspot.com/detail.html?file_id=1458674.

Bathsheba Grossman. 2012. metratron. https://www.thingiverse.com/thing:16673.

Alexandre Binninger, Ruben Wiersma, Philipp Herholz, and Olga Sorkine-Hornung. 2025. TetWeave: Isosurface Extraction using On-The-Fly Delaunay Tetrahedral Grids for Gradient-Based Mesh Optimization. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–19.

BlyncStudio. 2015. Buster Sword. https://www.thingiverse.com/thing:759413.

Alan Brunton and Lubna Abu Rmaileh. 2021. Displaced Signed Distance Fields for Additive Manufacturing. *ACM Trans. Graph.* 40, 4, Article 179 (2021), 13 pages. doi:10.1145/3450626.3459827

ChapinEagle02. 2015. The Chichen Itza pyramid. https://www.thingiverse.com/thing:675795.

Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. 2022. Neural dual contouring. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13.

Zhiqin Chen and Hao Zhang. 2021. Neural marching cubes. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–15.

Guillaume Coiffier and Louis Béthune. 2024. 1-Lipschitz Neural Distance Fields. In *Computer Graphics Forum*, Vol. 43. Wiley Online Library, e15128.

Bruno Rodrigues De Araújo, Daniel S Lopes, Pauline Jepp, Joaquim A Jorge, and Brian Wyvill. 2015. A survey on implicit surface polygonization. *ACM Computing Surveys (CSUR)* 47, 4 (2015), 1–39.

demonofthenorth. 2014. Empire State Building. https://www.thingiverse.com/thing:421017.

Zheng-Jie Deng, Xiao-Nan Luo, and Xiao-Ping Miao. 2011. Automatic cage building with quadric error metrics. *Journal of Computer Science and Technology* 26, 3 (2011), 538–547.

Ellindsey. 2013. Thingi10K model 375275. https://ten-thousand-models.appspot.com/detail.html?file_id=375275.

Salvatore Esposito, Daniel Rebain, Arno Onken, Changjian Li, and Oisin Mac Aodha. 2025. VesselSDF: Distance Field Priors for Vascular Network Reconstruction. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 674–683.

etrohn. 2012. Thingi10K model 69079. https://ten-thousand-models.appspot.com/detail.html?file_id=69079.

Nick Foster and Ronald Fedkiw. 2001. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 23–30.

Arnulph Fuhrmann, Gerrit Sobotka, and Clemens Groß. 2003. Distance fields for rapid collision detection in physically based modeling. In *Proceedings of GraphiCon*, Vol. 2003. 58–65.

Michael Garland and Paul S Heckbert. 1998. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*. IEEE, 263–269.

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.

Jorge N Hayek, Dave A May, Casper Pranger, and Alice-Agnes Gabriel. 2023. A diffuse interface method for earthquake rupture dynamics based on a phase-field model. *Journal of Geophysical Research: Solid Earth* 128, 12 (2023), e2023JB027143.

Adrian Hilton, Andrew J Stoddart, John Illingworth, and Terry Windeatt. 1996. Marching triangles: range image fusion for complex object modelling. In *Proceedings of 3rd IEEE international conference on image processing*, Vol. 2. IEEE, 381–384.

Jisung Hwang and Minhyuk Sung. 2024. Occupancy-Based Dual Contouring. In *SIGGRAPH Asia 2024 Conference Papers*. 1–11.

Alec Jacobson, Daniele Panozzo, et al. 2025. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. 2002. Dual contouring of Hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 339–346.

Tao Ju and Tushar Udeshi. 2006. Intersection-free contouring on an octree grid. In *Proceedings of Pacific graphics*, Vol. 2006.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, Vol. 7. 0.

Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson surface reconstruction. *ACM Transactions on Graphics (ToG)* 32, 3 (2013), 1–13.

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)*.

Maximilian Kohlbrenner and Marc Alexa. 2025a. Isosurface Extraction for Signed Distance Functions using Power Diagrams. In *Computer Graphics Forum*. Wiley Online Library, e70037.

Maximilian Kohlbrenner and Marc Alexa. 2025b. A Polyhedral Construction of Empty Spheres in Discrete Distance Fields. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*. 1–10.

Hélene Legrand, Jean-Marc Thiery, and Tamy Boubekeur. 2019. Filtered quadrics for high-speed geometry smoothing and clustering. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 663–677.

Jiabao Lei and Kui Jia. 2020. Analytic marching: An analytic meshing solution from deep implicit surface networks. In *International Conference on Machine Learning*. PMLR, 5789–5798.

Hsueh-Ti Derek Liu. 2023. BlenderToolbox. https://github.com/HTDerekLiu/BlenderToolbox.

Longdu Liu, Hao Yu, Shiqing Xin, Shuangmin Chen, Hongwei Lin, Wenping Wang, and Changhe Tu. 2025. Direct Extraction of High-Quality and Feature-Preserving Triangle Meshes from Signed Distance Functions. In *International Conference on Computational Visual Media*. Springer, 113–126.

Puze Liu, Kuo Zhang, Davide Tateo, Snehal Jauhri, Jan Peters, and Georgia Chalvatzaki. 2022. Regularized Deep Signed Distance Fields for Reactive Motion Generation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 6673–6680. doi:10.1109/IROS47612.2022.9981456

William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *Computer Graphics*.

LosoPower. 2022. Monster Hunter Rathalos Head. https://www.thingiverse.com/thing:5739684.

Zoë Marschner, Silvia Sellán, Hsueh-Ti Derek Liu, and Alec Jacobson. 2023. Constructive Solid Geometry on Neural Signed Distance Fields. In *SIGGRAPH Asia 2023 Conference Papers*. 1–12.

Nissim Maruani, Roman Klokov, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. 2023. VoroMesh: Learning watertight surface meshes with Voronoi diagrams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14565–14574.

Nissim Maruani, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. 2024. PoNQ: a neural qem-based mesh representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3647–3657.

Metafold 3D. 2026. Metafold Developer Docs: Core Concepts. https://docs.metafold3d.com/core-concepts. Accessed: 2026-01-18.

MJCbull. 2012. Great Wall of China. https://www.thingiverse.com/thing:22360.

nTop. 2019. Implicits and Fields for Beginners. https://www.ntop.com/resources/blog/implicits-and-fields-for-beginners/. Accessed: 2026-01-18.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 165–174.

Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. 2020. MeshSDF: Differentiable iso-surface extraction. *Advances in Neural Information Processing Systems* 33 (2020), 22468–22478.

Scott Schaefer, Tao Ju, and Joe Warren. 2007. Manifold dual contouring. *IEEE Transactions on Visualization and Computer Graphics* 13, 3 (2007), 610–619.

Scott Schaefer and Joe Warren. 2002. Dual contouring: The secret sauce. *Technical Report* 2, 408 (2002).

Silvia Sellán, Noam Aigerman, and Alec Jacobson. 2021. Swept volumes via spacetime numerical continuation. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–11.

Silvia Sellán, Christopher Batty, and Oded Stein. 2023. Reach For the Spheres: Tangency-Aware Surface Reconstruction of SDFs. In *SIGGRAPH Asia 2023 Conference Papers*. Article 73, 11 pages.

Silvia Sellán, Yingying Ren, Christopher Batty, and Oded Stein. 2024. Reach for the arcs: Reconstructing surfaces from SDFs via tangent points. In *ACM SIGGRAPH 2024 Conference Papers*. 1–12.

Silvia Sellán and Oded Stein. 2025. gpytoolbox: A Python Geometry Processing Toolbox. https://gpytoolbox.org/.

James A Sethian and Peter Smereka. 2003. Level set methods for fluid interfaces. *Annual review of fluid mechanics* 35, 1 (2003), 341–372.

Nicholas Sharp and Alec Jacobson. 2022. Spelunking the deep: Guaranteed queries on general neural implicit surfaces via range analysis. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–16.

Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. 2023a. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–16.

Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. 2023b. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–16.

SometimesCake. 2012. Thingi10K model 108613. https://ten-thousand-models.appspot.com/detail.html?file_id=108613.

Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*. 109–116.

Barton T. Stander and John C. Hart. 1997. Guaranteeing the Topology of an Implicit Surface Polygonization for Interactive Modeling. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. 279–286. doi:10.1145/258734.258868

Christian Stippel, Felix Mujkanovic, Thomas Leimkühler, and Pedro Hermosilla. 2025. Marching Neurons: Accurate Surface Extraction for Neural Implicit Shapes. *ACM Transactions on Graphics (TOG)* 44, 6 (2025), 1–12.

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. (2021).

The Stanford 3D Scanning Repository. 1994. Stanford Bunny. http://graphics.stanford.edu/data/3Dscanrep/.

Jean-Marc Thiery, Émilie Guy, and Tamy Boubekeur. 2013. Sphere-meshes: Shape approximation using spherical quadric error metrics. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–12.

Philip Trettner and Leif Kobbelt. 2020. Fast and robust qef minimization using probabilistic quadrics. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 325–334.

Zixiong Wang, Yunxiao Zhang, Rui Xu, Fan Zhang, Peng-Shuai Wang, Shuangmin Chen, Shiqing Xin, Wenping Wang, and Changhe Tu. 2023. Neural-singular-hessian: Implicit neural representation of unoriented point clouds by enforcing singular hessian. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–14.

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2022. Neural fields in visual computing and beyond. In *Computer graphics forum*, Vol. 41. Wiley Online Library, 641–676.

Rui Xu, Longdu Liu, Ningna Wang, Shuangmin Chen, Shiqing Xin, Xiaohu Guo, Zichun Zhong, Taku Komura, Wenping Wang, and Changhe Tu. 2024. CWF: consolidating weak features in high-quality mesh simplification. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–14.

Congyi Zhang, Guying Lin, Lei Yang, Xin Li, Taku Komura, Scott Schaefer, John Keyser, and Wenping Wang. 2023. Surface extraction from neural unsigned distance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 22531–22540.

A Pseudocode

Algorithm 1: Dual Contouring of signed distance data

function sdf_dual_contouring($c_1, \ldots, c_m, \mathbf{u}_1, \ldots, \mathbf{u}_n, s_1, \ldots, s_n$)
  Identify interesting edges and cells using signs of s_i
  foreach *interesting edge* e **do**
    Estimate $\mathbf{h}^0, \mathbf{n}^0$ using Equations 1, 2
  foreach *interesting cell* c_i **do**
    $\mathbf{c}_i \leftarrow$ average of Hermite data contained in c_i
    $\mathbf{x}_i^0 \leftarrow \mathbf{c}_i$
  for $k = 0, \ldots,$ *max_outer_iter* **do**
    $\mathcal{M}^k \leftarrow$ connect vertices to form global mesh
    $\mathbf{p}_1, \cdots \leftarrow$ intersect edges of $\mathcal{M}^k$ with cell faces to find face intersection points
    Triangulate global mesh $\mathcal{M}^k$
    foreach *SDF data point* $(\mathbf{u}_i, s_i)$ **do**
      Find closest point to $\mathbf{u}_i$ on $\mathcal{M}^k$
      Find cell c_j closest point belongs to
      Assign $(\mathbf{u}_i, s_i)$ to c_j
    foreach *cell* c_i **do** // parallel
      $\mathbf{x}^{k,0} \leftarrow \mathbf{x}^k$
      for $r = 1, \ldots,$ *max_inner_iters* **do**
        $\mathcal{M}_i^{k,r} \leftarrow$ build local mesh using $\mathbf{x}^{k,r}$
        $\mathbf{t}_1^r, \mathbf{q}_1^r, \cdots \leftarrow$ compute closest points on local mesh to assigned $\mathbf{u}$
        $\mathbf{x}^{k,r+1} \leftarrow$ solve quadratic minimization problem
        if $\|\mathbf{x}^{k,r+1} - \mathbf{x}^{k,r}\| \leq$ *tol* **then**
          break
      $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^{k,r_{\text{last}}}$
    foreach *interesting edge* **do**
      $\mathbf{h}^{k+1}, \mathbf{n}^{k+1} \leftarrow$ update Hermite data using Equation 7
  $\mathcal{M} \leftarrow$ connect vertices to form final global mesh
  return $\mathcal{M}$

B Tables

Tables 1, 2 and 3 show numerical results for the large-scale quantitative results explained in section 4.

Table 1. Quantitative comparison under increasing noise (averaged over 10 shapes, at resolution 100^3). Best scores are bold; second best are underlined.

Method	Chamfer error ($\times 10^{-3}$)	Hausdorff error ($\times 10^{-2}$)	Edge chamfer error
Noise = 0.001			
[Lorensen and Cline 1987]	4.552	2.448	0.2272
[Ju et al. 2002] (est. Hermite)	3.698	2.129	0.1253
[Kohlbrenner and Alexa 2025a]	3.611	**2.107**	0.1468
[Kohlbrenner and Alexa 2025b]	4.808	2.448	0.3137
Ours	**2.683**	2.651	**0.04688**
Noise = 0.005			
[Lorensen and Cline 1987]	6.338	2.117	**0.08841**
[Ju et al. 2002] (est. Hermite)	6.543	5.037	0.1022
[Kohlbrenner and Alexa 2025a]	**3.612**	**2.109**	0.1401
[Kohlbrenner and Alexa 2025b]	9.734	3.3	0.2394
Ours	5.104	4.816	0.1027
Noise = 0.01			
[Lorensen and Cline 1987]	11.8	3.713	0.1147
[Ju et al. 2002] (est. Hermite)	12.8	6.927	**0.1124**
[Kohlbrenner and Alexa 2025a]	**3.614**	**2.11**	0.1423
[Kohlbrenner and Alexa 2025b]	24.32	4.093	0.2183
Ours	10.79	8.729	0.1138

C Quadratic minimization

We now give additional details on how we solve the minimization problem in the inner loop. At a given inner iteration r of a fixed outer iteration, we optimize the position of the cell vertex by solving

$$\mathbf{x}^{r+1} = \arg\min_{\mathbf{x}} w_H^2 \sum_{e_j \in c_i} \left((\mathbf{x} - \mathbf{h}_j) \cdot \mathbf{n}_j\right)^2 + \|A_d^r \mathbf{x} - b_d^r\|^2 + \mu \|\mathbf{x} - \mathbf{x}^r\|^2. \quad (12)$$

This is a quadratic minimization problem, which can be expressed in the form

$$\mathbf{x}^{r+1} = \arg\min_{\mathbf{x}} \|Q\mathbf{x} - c\|^2 = \arg\min_{\mathbf{x}} \mathbf{x}^T Q^T Q \mathbf{x} - 2c^T Q \mathbf{x} + c^T c, \quad (13)$$

where Q is a matrix of size $(N_H + N_d + 3) \times 3$ (with N_H being the number of edges with Hermite data of cell c_i, and N_d, the number of SDF data points assigned to c_i), and c is a vector of size $N_H + N_d + 3$.

In particular, Q is built by vertically stacking the following three matrices:

$$Q_{dc} = w_H \begin{bmatrix} \mathbf{n}_1^T \\ \mathbf{n}_2^T \\ \vdots \\ \mathbf{n}_{N_H}^T \end{bmatrix} \in \mathbb{R}^{N_H \times 3}, \quad Q_d = A_d^r \in \mathbb{R}^{N_d \times 3}, \quad Q_\mu = \sqrt{\mu} I_3 \in \mathbb{R}^{3 \times 3}, \quad (14)$$

where each row of Q_d stores $\alpha \cdot \mathbf{d}^T$, denoting by α the barycentric coordinate of $\mathbf{t}_j^r$ with respect to $\mathbf{x}^r$. Similarly, c is built by vertically stacking the following three vectors:

$$c_{dc} = w_H \begin{bmatrix} \mathbf{n}_1^T \mathbf{h}_1 \\ \mathbf{n}_2^T \mathbf{h}_2 \\ \vdots \\ \mathbf{n}_{N_H}^T \mathbf{h}_{N_H} \end{bmatrix} \in \mathbb{R}^{N_H}, \quad c_d = b_d^r \in \mathbb{R}^{N_d}, \quad c_\mu = \sqrt{\mu} \mathbf{x}^r \in \mathbb{R}^3. \quad (15)$$

Each row of c_d then stores $q_j \cdot \mathbf{d}_j - \beta \mathbf{h} \cdot \mathbf{d}_j - \gamma \mathbf{p} \cdot \mathbf{d}_j$, where β and γ are the barycentric coordinates of $\mathbf{t}_j^r$ corresponding to $\mathbf{h}_j$ and $\mathbf{p}_j$, respectively. When $\alpha = 0$, we set $\alpha = 1$, $\beta = 0$ and $\gamma = 0$, avoiding numerical locking by encouraging the vertex to move towards or away from the sphere. Finally, we solve the system following the same approach as Ju et al. [2002] (see their Sec. 2.3).

D Additional results

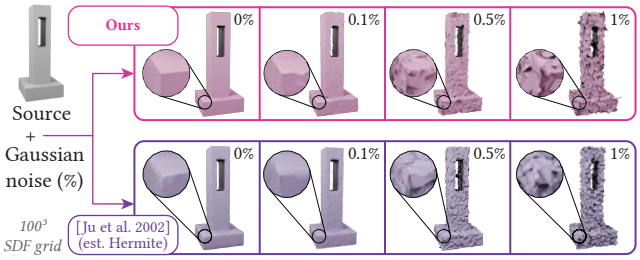

Fig. 22. Comparison of the results of our method and [Ju et al. 2002] under different noise levels.

Table 2. A quantitative comparison of error metrics (averaged over 55 random shapes from the ABC dataset), across three different resolutions. Best scores are bold; second best are underlined. Our method yields the best result in all but two instances.

Method	Chamfer error ($\times 10^{-3}$)	Hausdorff error ($\times 10^{-2}$)	Edge chamfer error	SDF energy ($\times 10^{-5}$)	#Vertices
Resolution 50^3					
[Lorensen and Cline 1987]	7.586	3.692	0.3636	13.34	3.4k
[Ju et al. 2002] (est. Hermite)	4.71	2.937	0.356	8.611	**3.4k**
[Sellán et al. 2024]	4.976	**2.88**	0.1069	**0.4781**	151k
[Kohlbrenner and Alexa 2025a]	5.063	3.033	0.2996	8.528	27k
[Kohlbrenner and Alexa 2025b]	13.04	3.517	0.4066	3.826	73k
Ours	**2.958**	2.97	**0.03787**	1.074	**3.4k**
Resolution 100^3					
[Lorensen and Cline 1987]	2.62	1.81	0.4171	3.061	**14k**
[Ju et al. 2002] (est. Hermite)	1.589	1.372	0.3502	1.866	**14k**
[Sellán et al. 2024]	30.8	6.284	0.1143	130.8	274k
[Kohlbrenner and Alexa 2025a]	1.843	1.492	0.274	1.812	123k
[Kohlbrenner and Alexa 2025b]	7.242	2.216	0.4466	2.321	163k
Ours	**0.7788**	**1.221**	**0.02622**	**0.05776**	**14k**
Resolution 150^3					
[Lorensen and Cline 1987]	1.329	1.107	0.3256	1.175	32k
[Ju et al. 2002] (est. Hermite)	0.8344	0.8513	0.3139	0.7912	**32k**
[Kohlbrenner and Alexa 2025a]	1.118	0.9435	0.2608	0.7778	281k
Ours	**0.4432**	**0.7037**	**0.01684**	**0.241**	**32k**
Resolution 200^3					
[Lorensen and Cline 1987]	1.059	0.9969	0.3003	1.115	57k
[Ju et al. 2002] (est. Hermite)	0.7411	0.8208	0.2929	0.9148	**57k**
[Kohlbrenner and Alexa 2025a]	1.037	0.8769	0.1999	0.9096	496k
Ours	**0.4576**	**0.6741**	**0.01599**	**0.4359**	**57k**

Table 3. Results from our method and related work on 10 smooth shapes used for evaluation by Sellán et al. [2024].

Method	Chamfer error ($\times 10^{-3}$)	Hausdorff error ($\times 10^{-2}$)	Edge chamfer error	SDF energy ($\times 10^{-5}$)	#Vertices
Resolution 50^3					
[Lorensen and Cline 1987]	11.47	8.033	0.1897	29.06	2.5k
[Ju et al. 2002] (est. Hermite)	10.23	8.803	0.1842	28.75	**2.5k**
[Sellán et al. 2024]	11.35	**5.433**	0.2002	**0.1906**	109k
[Kohlbrenner and Alexa 2025a]	9.813	7.648	**0.1403**	25.79	23k
[Kohlbrenner and Alexa 2025b]	13.27	5.6	0.2919	0.3155	97k
Ours	**9.479**	6.58	0.1601	9.801	**2.5k**
Resolution 100^3					
[Lorensen and Cline 1987]	4.579	5.233	0.1582	9.785	11k
[Ju et al. 2002] (est. Hermite)	4.206	6.402	0.1402	9.55	**11k**
[Sellán et al. 2024]	6.039	4.698	0.1696	**0.1275**	201k
[Kohlbrenner and Alexa 2025a]	3.852	5.224	**0.1092**	8.721	121k
[Kohlbrenner and Alexa 2025b]	5.445	**3.444**	0.3845	0.1496	189k
Ours	**3.804**	4.745	0.1457	6.346	**11k**
Resolution 150^3					
[Lorensen and Cline 1987]	2.99	4.375	0.1462	7.618	25k
[Ju et al. 2002] (est. Hermite)	2.681	5.413	0.1245	7.51	**25k**
[Kohlbrenner and Alexa 2025a]	**2.477**	4.241	0.1236	7.324	306k
Ours	2.811	**3.95**	**0.1064**	**6.072**	**25k**