Mesh Slicing Along Isolines of Surface-Based Functions

Lei Wang[†1], Xudong Wang[†1], Wensong Wang[1], Shuangmin Chen[‡2], Shiqing Xin[1], Changhe Tu[1], Wenping Wang[3]

[1]Shandong University, China
[2]Qingdao University of Science and Technology, China
[3]Texas A&M University, USA

Abstract

There are numerous practical scenarios where the surface of a 3D object is equipped with varying properties. The process of slicing the surface along the isoline of the property field is a widely utilized operation. While the geometry of the 3D object can typically be approximated with a piecewise linear triangle mesh, the property field f might be too intricate to be linearly approximated at the same resolution. Arbitrarily reducing the isoline within a triangle into a straight-line segment could result in noticeable artifacts. In this paper, we delve into the precise extraction of the isoline of a surface-based function f for slicing the surface apart, allowing the extracted isoline to be curved within a triangle. Our approach begins by adequately sampling Steiner points on mesh edges. Subsequently, for each triangle, we categorize the Steiner points into two groups based on the signs of their function values. We then trace the bisector between these two groups of Steiner points by simply computing a 2D power diagram of all Steiner points. It's worth noting that the weight setting of the power diagram is derived from the first-order approximation of f. Finally, we refine the polygonal bisector by adjusting each vertex to the closest point on the actual isoline. Each step of our algorithm is fully parallelizable on a triangle level, making it highly efficient. Additionally, we provide numerous examples to illustrate its practical applications.

CCS Concepts

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

1. Introduction

In the field of digital geometry processing, the geometry of a 3D object often incorporates specific attributes, such as curvature fields [MWS*21]. A common approach to examine variations on the surface is through the visualization of the isolines of the property field. The precise extraction of these isolines is crucial for a wide range of applications, not limited to visualization [BJB*11, YMS*21] and shape analysis [WS19].

When dealing with surfaces represented by triangle meshes, a typical method to capture the varying property field is to treat the change within each triangle as linear, recording the values at the mesh vertices. However, the property field f may be too intricate to be adequately approximated by such a linear model at the same resolution. As a result, reducing the isoline within a triangle to a mere straight-line segment can lead to considerable inaccuracies and visual distortions.

As illustrated in Figure 1, we select sphere as the base surface $\mathcal{M}$ and proceed to extract the isoline for $f(\mathbf{x} \in \mathcal{M}) = (x^2+y^2-1)^3 - x^2y^3$. When we assume that the change of f within each triangle is linear, this leads to significant deviations at the common edges between two adjacent triangles, as depicted in Figure 1 (a). A practical strategy to mitigate this issue involves subdividing the triangles to enhance the resolution of the extracted isolines, as demonstrated in Figure 1 (b). Nonetheless, this technique is still prone to visual irregularities at points where isolines exhibit cusps.

In this research, we introduce a straightforward yet effective method for the precise extraction of the isoline at value c. Our approach initiates with the sampling of Steiner points along the mesh edges. These Steiner points are subsequently sorted into two categories based on the sign of their function values. Utilizing the power diagram as the analytical framework, we define the boundary curve that differentiates the two groups of Steiner points. It's critical to note that the power diagram's weights are determined by the first-order approximation of $f(\mathbf{x})$. This boundary curve is then refined by adjusting each vertex to satisfy $f(\mathbf{x}) = c$. As showcased in Figure 1 (c), our method allows highly curved isolines within a triangle and accurately delineates the cusps, thereby outperforming current techniques. The efficiency of our algorithm is significantly boosted as each procedure can be parallelized at the level of individual triangles. We also provide a range of examples to illustrate its versatile applications.

Our contributions are articulated through four primary aspects:

† Equal contribution
‡ Corresponding author

DOI: 10.2312/pg.20241283

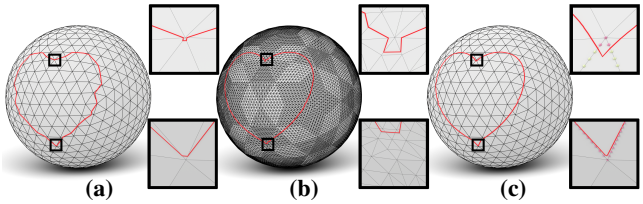


Figure 1: *Comparison between our approach and existing methods. (a) When the change of f within each triangle is assumed to be linear, the isolines extracted are notably inaccurate. (b) Even with increased resolution by subdividing triangles, visual irregularities persist at points where isolines develop cusps. (c) Our method excels by permitting highly curved isolines within a triangle and precisely delineating cusps, thus surpassing current techniques.*

- We introduce a methodology for precise tracing of the isolines of a function defined on a surface. This method fundamentally separates the approximation of the surface-based function from the piecewise linear representation of the geometry.
- We recast the challenge of isoline extraction into a solvable power diagram problem, with the weights established by the first-order approximation of f.
- We design a novel feature-preserving fine-tuning framework with the potential to be integrated into other isoline extraction methods.
- We innovatively implement our algorithm in a range of applications, notably in slicing polygonal surfaces.

2. Related Work

2.1. Isolines and Applications

Isolines are defined as curves within a scalar field (such as geodesic distance field) that connect points of identical value. These are also known as level sets, formally represented by

$$L_f(c) = \{\mathbf{x} \,|\, f(\mathbf{x}) = c\}.$$

In the context of this study, we explore contouring isolines on a three-dimensional triangle mesh, essentially treating it as a two-dimensional problem where each triangle represents a discrete unit.

Generally, scalar fields within a triangle are typically considered linear, rendering the isoline a straight segment. Therefore, common methods for generating these contours rely on linear interpolation, which estimates values at intermediate points based on nearby data. Nevertheless, the triangle mesh only discretizes the shape, implying that the scalar field can exhibit different resolutions within. Specifically, isolines are permitted to curve within each triangle. Even when triangles are subdivided to increase resolution, visual inconsistencies, such as cusps in the isolines, may remain.

Surface Reconstruction. The most similar 3D task to contouring isolines is surface reconstruction from signed distance fields (SDFs), with the classic isosurfacing approach being Marching Cubes [LC87] and its variants. It divides space into a lattice of cubes, determining shapes from the scalar values at the corners of these cubes. This approach, however, can lead to topological challenges and feature loss. Approaches like dual contouring [JLSW02,KBSS01] improve sharp feature recovery by incorporating surface normals. Unfortunately, discrete SDFs lack the accurate gradient information and finite difference estimates yield poor results. With the help of neural networks, [CZ21] and [CTFZ22] achieve better reconstructions by training on a large dataset of SDFs.

Geodesic Distance Field. A common example involves defining the scalar field on a surface through geodesic distances. We provide a brief overview of notable work utilizing geodesic fields and their interesting applications. Liu et al. [LCT11] introduced effective algorithms for precise geodesic line extraction, building on Surazhsky et al.'s [SSK*05] implementation of the MMP algorithm [MMP87]. Their technique begins with the calculation of a geodesic distance field via MMP. It maintains all 'windows' propagated along each mesh edge, then recursively subdivides each edge into monotone segments, searching for arc segments within each re-tessellated triangle. Despite its effectiveness, the computational demand of this method is substantial. Unlike traditional methods that utilize window propagation, a simple yet effective method for computing geodesic distances on triangle meshes has been proposed in [MXT*21]. This approach constructs a Steiner-point graph, which partitions the surface into mutually exclusive tracks, ensuring accurate isoline tracing. Recently, Mancinelli et al.'s [MLP21, MP22] utilized geodesic distance field to compute cut locus and vector graphics on a surface.

2.2. Boolean Operations

A possible approach to extract isoline involves treating the process as an boolean intersection. Mesh booleans generally encompass vertex- or plane-based methodologies. Vertex-based methods commonly involve computing intersections of the input meshes and subsequently determining the inclusion of resulting polygons in the output [BGF15,ZGZJ16,CLSA20]. On the other hand, plane-based boolean approaches [NAT90] represent input solids via implicit BSP trees.

One significant challenge in mesh boolean operations stems from stability issues arising from inconsistencies caused by floating-point rounding. Recently, a robust and interactive boolean method called EMBER was presented [TNWK22]. This algorithm is based on the use of homogeneous integer numbers to represent point coordinates exactly, but can only be considered exact if input and output are in integer homogeneous coordinates subject to the fixed-width precision limits.

Another common problem in most mesh boolean algorithms is determining whether elements are part of the result or not. Apart from past boolean methods using surface walking [Att14, SJK10], or generalized winding number [JKSH13], a novel inside/outside classification system based on exact ray casting was proposed [CPAL22], but it may time-consuming on very high resolution meshes.

Furthermore, mesh boolean methods commonly impose specific requirements on the input mesh. For instance, the criterion outlined in [ZGZJ16] necessitates that the input mesh be both manifold and well-oriented, while [CPAL22] stipulates that the input mesh must be manifold, watertight, self-intersections free, and well-oriented.

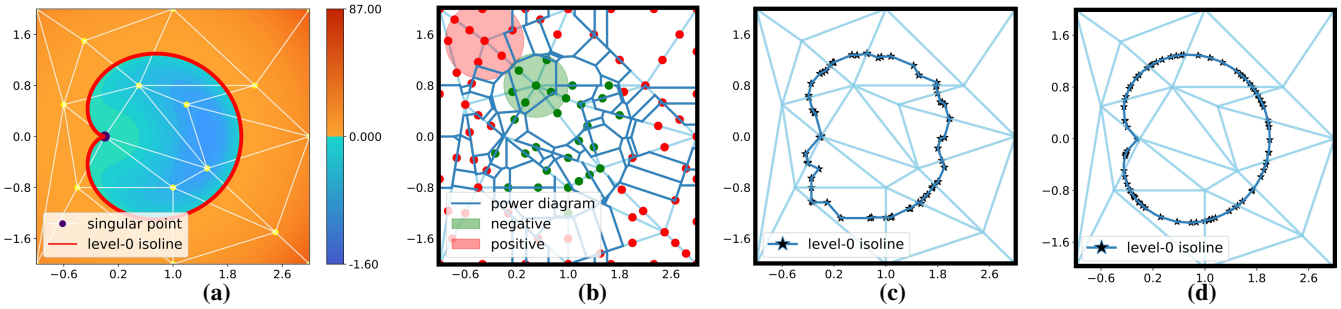


Figure 2: *We illustrate the workflow of our algorithm using a 2D example: (a) The function f is represented using a color-coded scheme, with the red curve indicating the actual zero isoline we aim to extract. (b) Through a selection of sample points as sites, we categorize them into two groups based on their sign: red sites and green sites. (c) By examining the power diagram of these sites, we identify edges that act as bisectors between the two groups, providing an initial approximation of the isoline. It's worth noting that the weight setting can be promptly estimated by considering the gradients of f. (d) A refinement process fine-tunes the positions of points to closely match the isolines of f.*

Therefore, even though boolean intersection can serve as a tool for the task of tracing isolines, it is susceptible to various factors, making it not entirely robust. In contrast, our method circumvents the explicit acquisition of meshes, ensuring applicability across a wide range of scenarios.

2.3. Power Diagram

Voronoi diagram [AK00], also known as Voronoi tessellation or Voronoi partition, is a partitioning of a plane or space $\Omega \subset \mathbb{R}^{\mathbb{N}}$ into a set of disjoint subregions $V(\mathbf{S}) = \bigcup_{i=1}^{n} V(\mathbf{s}_i)$ based on the proximity to a set of sites $\mathbf{S} = \{s_i\}_{i=1}^{n}$. Each Voronoi cell $V(\mathbf{s}_i)$ is defined as

$$V(\mathbf{s}_i) = \{\mathbf{x} \in \Omega \,|\, dis(\mathbf{x}, \mathbf{s}_i) \leq dis(\mathbf{x}, \mathbf{s}_j), \forall j \neq i\}.$$

Power diagram [Aur87] is a type of weighted Voronoi diagram, where each site $\mathbf{s}_i$ is associated with a non-negative scalar value w_i, the *weight* of $\mathbf{s}_i$. The power cell $P(\mathbf{s}_i)$ of $\mathbf{s}_i$ is defined as

$$P(\mathbf{s}_i) = \{\mathbf{x} \in \Omega \,|\, dis(\mathbf{x}, \mathbf{s}_i)^2 - w_i \leq dis(\mathbf{x}, \mathbf{s}_j)^2 - w_j, \forall j \neq i\}.$$

Power diagrams are widely utilized in the field of geometry processing. A seminal work is the power crust algorithm [ACK01], which samples points from the surface of a three-dimensional object to generate a surface mesh and an approximate medial axis. Another notable advancement is the concept of the restricted power diagram (RPD), which involves computing the power diagram on a surface by determining the intersections between a 3D power cell and the original surface, as detailed by [YLL*09]. Furthermore, the application of restricted power diagrams has been extended to the computation of the medial axis transform while preserving features, as demonstrated by [WWWG22].

3. Our Algorithm

3.1. Algorithm Overview

Given a triangle mesh surface with a surface-based function f, the objective of this paper is to extract the isoline of f at a specified value, typically 0. The output is desired to be as accurate as possible, allowing for curved changes within a triangle. We utilize Figure 2 to illustrate the workflow of our algorithm, with the red curve indicating the actual zero isoline we aim to extract.

Initially, our algorithm begins by uniformly sampling Steiner points along the edges of $\mathcal{M}$. These points are then classified into two groups based on the sign of their function values. Each site determines a circle touching the actual isoline, with the radius quickly estimated by considering the gradients of f. By examining the power diagram of these sites, we identify the bisector between the two groups as an initial approximation of the actual isoline. Finally, a lightweight refinement process adjusts the positions of points to align them with the isoline of f. We shall elaborate these steps in detail in the following subsections.

3.2. Insight of Utilizing Power Diagram

The key insight of our method is (see Figure 3 left) that each sample $\mathbf{x}$ on the surface corresponds to a circle, centered at $\mathbf{x}$ and with radius equal to $|f(\mathbf{x}) - c|$, representing the closest distance to the actual isoline. By definition, the true isoline lies between positive ($f(\mathbf{x}) > c$) and negative ($f(\mathbf{x}) < c$) regions, touching these circles tangentially, strictly enclosing negative values and excluding positive ones.

Following this observation, we utilize the power diagram tool to infer isolines. This is because each site $\mathbf{s}_i$ can be conceptually associated with a circle of radius $\sqrt{w_i}$. When the weight is set to the squared distance to the true isoline, the circles from the two groups of sites (negative/positive) are clearly separated from the isoline. The power diagram then forms a polygonal curve that closely matches the actual isoline without losing any features (see Figure 3 right).

Unlike piecewise linear approximations, our method adheres to the unique and fundamental properties of level-sets, avoiding potential detriment, oversmoothing and feature loss. Additionally, it remains independent of mesh resolution and function, and can incorporate new samples to enhance extraction.

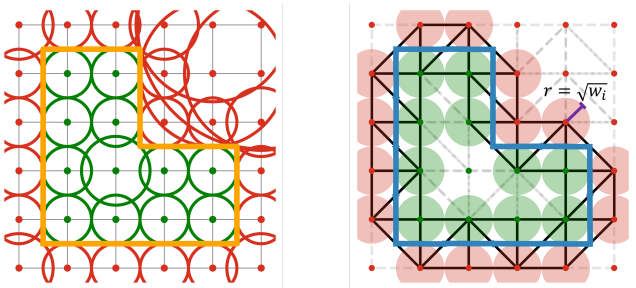


Figure 3: *Left: The actual isoline (orange) conforms to the constraints of all spheres from negative samples (green) and positive samples (red) tangent to it. Right: Sites are classified into two groups based on the signs of their function values. Our research is motivated by the observation that the power diagram of these sites, with an appropriate weight, induces a polygonal curve (blue) that approximates the actual isoline.*

3.3. Weight Setting

Let $\mathbf{s}_i$ represent one of the sites, and $\mathbf{s}_i'$ denote the closest point on the isoline, satisfying $f(\mathbf{s}_i') = c$, where c is the user-specified value.

Suppose that f is differentiable and the gradient does not vanish at $\mathbf{s}_i$. Based on the first-order approximation, we have

$$c = f(\mathbf{s}_i') \approx f(\mathbf{s}_i) + \nabla f(\mathbf{s}_i) \cdot (\mathbf{s}_i' - \mathbf{s}_i). \tag{1}$$

By setting $\mathbf{s}_i' = \mathbf{s}_i + \alpha_i \nabla f(\mathbf{s}_i)$, it is easy to find α_i:

$$\alpha_i = \frac{c - f(\mathbf{s}_i)}{\|\nabla f(\mathbf{s}_i)\|^2}. \tag{2}$$

The distance from $\mathbf{s}_i$ to the isoline can be estimated by

$$\|\mathbf{s}_i' - \mathbf{s}_i\| = \|\alpha_i \nabla f(\mathbf{s}_i)\| = \frac{|c - f(\mathbf{s}_i)|}{\|\nabla f(\mathbf{s}_i)\|}. \tag{3}$$

Considering that the power diagram is defined based on the squared distance, we set the weight of $\mathbf{s}_i$ as

$$w_i = \frac{|c - f(\mathbf{s}_i)|^2}{\|\nabla f(\mathbf{s}_i)\|^2}. \tag{4}$$

To this end, each site is given a suitable weight. The calculation of the power diagram can be effectively performed using existing solvers, such as CGAL [The24]. Each edge in the power diagram is dual to a line segment that connects two adjacent sites. By examining all the edges, we are able to identify the edges whose contributing sites are from different groups. Such edges form an approximate representation of the underlying isoline.

Remark. If $f(\mathbf{s}_i)$ is very close to c, we exclude $\mathbf{s}_i$ before dividing the sites into two groups. Additionally, Eq. (4) assumes that the gradient does not vanish at $\mathbf{s}_i$. Therefore, if there exist some sites that have almost vanishing gradients, we can estimate the distance between $\mathbf{s}_i$ and the isoline in an alternative way or exclude $\mathbf{s}_i$ from the sites.

3.4. Refinement

In this step, we need to examine each vertex $\mathbf{p}$ of the approximate isoline extracted in the previous step and fine-tune it to a new position such that $f(\mathbf{p}_{\text{new}}) = c$.

We can formulate it into the following constrained optimization problem:

$$\begin{aligned} \min \quad & \|\mathbf{p}_{\text{new}} - \mathbf{p}\|^2 \\ \text{s.t.} \quad & f(\mathbf{p}_{\text{new}}) = c. \end{aligned} \tag{5}$$

We use the following iterative process to achieve the optimum:

$$\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} + \frac{c - f(\mathbf{p}^{(k)})}{\|\nabla f(\mathbf{p}^{(k)})\|^2} \nabla f(\mathbf{p}^{(k)}), \tag{6}$$

until the value of $f(\mathbf{p}^{(k)})$ is sufficiently close to c ($1e^{-6}$ by default). As illustrated in Figure 4, this iterative process exhibits a high convergence rate.

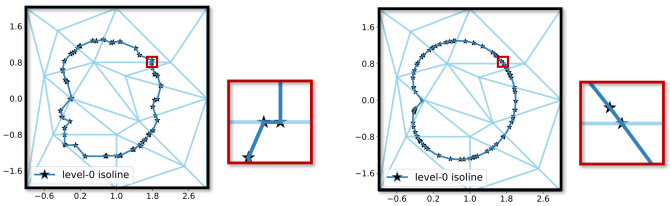


Figure 4: *The effective refinement process mitigates inconsistency issues, resulting in a more accurate approximation.*

Remark. Recall that before the fine-tuning process, there are some vertices of the approximate isoline located on mesh edges. For these vertices, we constrain the positional refinement on mesh edges. Otherwise, a vertex may result in two different positions upon optimization.

3.5. Treatment of Singular Points

The surface-based function f may have singular points (i.e., where the gradient vanishes). Typically, these points correspond to feature points on the isoline. Although they can be recovered with the help of the power diagram, they cannot be optimized through Eq. 6 to correctly distribute them to their true positions.

To address this, we introduce an extra optimization process specifically for the singular points $\mathbf{P}_s$:

$$\begin{aligned} \min \quad & \sum_{\mathbf{p}_s \in \mathbf{P}_s} \|(\mathbf{p}_s - \mathbf{p}^1) \cdot \nabla \mathbf{p}^1\|^2 + \|(\mathbf{p}_s - \mathbf{p}^2) \cdot \nabla \mathbf{p}^2\|^2 \\ \text{s.t.} \quad & f(\mathbf{p}_s) = c \quad \text{for } \forall \mathbf{p}_s \in \mathbf{P}_s, \end{aligned} \tag{7}$$

where $\mathbf{p}^1$ and $\mathbf{p}^2$ are the points preceding and following $\mathbf{p}_s$ on the isoline.

The fine-tuning process ensures that the singular points and their neighboring points are accurately positioned, we give an illustrative example as shown in Figure 5. Notably, compared to the traditional technique, our initial output via power diagram already recovers the feature point, thereby facilitating the optimization process. This underscores the importante role of power diagrams in detail recovery.

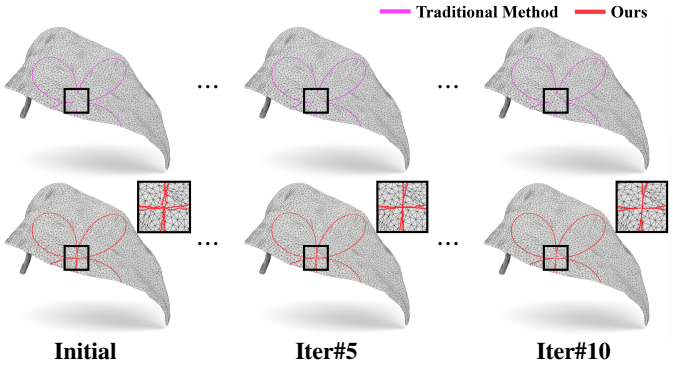


Figure 5: *The first and second row respectively shows the results (on the leaf model with 16K faces) of applying our refinement and singular points tuning processes to the initial outputs from our method and the traditional piecewise linear approximation method. Under the same number of iterations, our method produces superior visualized results (the highlighted window demonstrates our algorithm's capability to recover high-fidelity geometry details).*

4. Results

4.1. Experimental Settings

In this study, we start with the premise that the function in question is defined on a surface. It's conceivable that our algorithm would be equally effective for a function defined in three dimensions, that is, $f = f(x,y,z)$. In such scenarios, $f(x,y,z) = c$ defines a 3D curved surface, and the challenge of isoline extraction transforms into identifying the intersection curve between the base surface and $f(x,y,z) = c$. To validate the accuracy of our algorithm, we compare our approach with a boolean intersection-based method [ZGZJ16], chosen for its ease in tracing isolines. Then we test our algorithm on more complex examples to validate robustness where boolean intersection method failed. Furthermore, we performed a comparative experiment with piecewise linear approximation method (PLM) to demonstrate the effectiveness of our algorithm in recovering feature points.

All the 3D meshes we work with are normalized within the range $[-1,1]^3$. The functions utilized in our study are detailed in Table 1. For the computation of a robust and high-quality power diagram, we employ CGAL [The24], inputting a collection of sites and their corresponding weights. Additionally, we make use of the boolean intersection capabilities offered by libigl [JP*18]. Given the challenge of performing boolean intersections directly between the implicit function and the input mesh, it becomes necessary to first transform the implicit function into an explicit mesh. This is accomplished using the Marching Cubes algorithm [LC87], an implementation of which is also provided by libigl. All experiments are performed on a PC with a 32-core Intel CoreTM i9-13900K CPU clocked at 3.0 GHz and 64 GB of memory.

For a quantitative assessment, we calculate the average absolute error E_{avg} and max absolute error E_{max} with the ground truth using the following formulas:

$$E_{\text{avg}} = \frac{1}{n(\mathbf{P}_s)} \sum_{\mathbf{p}\in\mathbf{P}_s} |f(\mathbf{p}) - c|$$

$$E_{\text{max}} = \max_{\forall \mathbf{p}\in\mathbf{P}_s} |f(\mathbf{p}) - c|$$

where $\mathbf{p} \in \mathbf{P}_s$ is an arbitrary point on L_c, and $n(\mathbf{P}_s)$ denotes the total number of points.

4.2. Accuracy

In this section, we perform tests on various freeform meshes and smooth functions (referenced in Table 1) to assess the precision of our algorithm. For boolean intersection operations, we adjust the grid resolution in the Marching Cubes algorithm to 256, and for our approach, we specify the number of sampling points on each mesh edge as 10.

Based on the data presented in Figure 6 and Table 4.1, it's observed that the majority of the average absolute errors (E_{avg}) and the maximum absolute errors (E_{max}) obtained through our method are lower than those resulting from the boolean intersection approach. This indicates that our algorithm is capable of producing results that are comparable to those of the more time-intensive boolean intersection method.

4.3. Robustness

The boolean intersection method is subject to a number of limitations. As indicated by [ZGZJ16], it struggles with non-watertight meshes, meshes with inconsistent orientations, and self-intersecting meshes. In addition to these considerations, we carried out experiments using various functions, as outlined in Table 1. These functions are distinguished by their intricate structures and the presence of numerous critical or singular points. This complexity implies that, once transformed into an explicit mesh, they may not be amenable to boolean intersection operations.

Table 1: *The table presents all the functions utilized in the experiment, along with their expressions and an indication of whether they are smooth.*

Function	Expression	Smoothness
Ellipsoid	$(x/2)^4 + (y/2)^4 + (z/2)^4$	✓
Torus	$\sqrt{(8x^2+8z^2-2)^2+8y^2}$	✓
Bean	$(x^2+y^2+z^2-1)^2 - z^2 - 2x$	✓
Klein Bottle	$(x^2+y^2+z^2+2y-1)*$ $((x^2+y^2+z^2-2y-1)^2-8z^2)+16xz$ $(x^2+y^2+z^2-2y-1)$	✘
Conic	$\sqrt{x^2+y^2}-z$	✘
Quatrefoil Curve	$(x^2+y^2)^3 - x^2y^2$	✘
Mobius	$-y-2xz+x^2y-2x^2z+y^3-2y^2z+yz^2$	✓
Cross torus	$(30x^2+30y^2+30z^2+1)^2-$ $16\cdot(30x^2+30y^2)$	✓
Stellated octahedron	$(4x^2+4y^2+4z^2)^2 - 3\cdot(4x^2-4y^2)$	✓
Star	$x^{\frac{2}{3}}+y^{\frac{2}{3}}$	✘

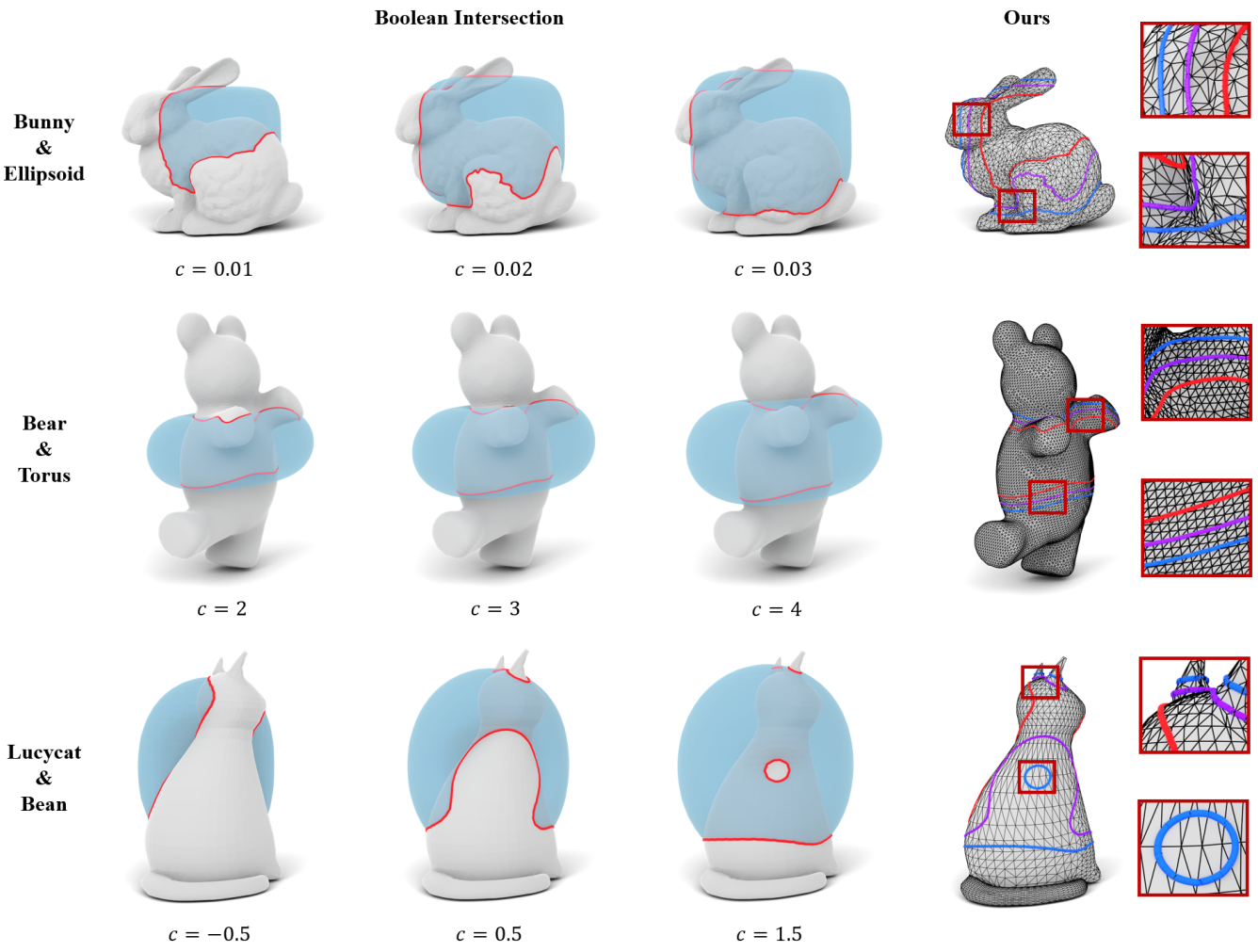


Figure 6: *Our algorithm effectively traces true isolines without contouring the mesh of the implicit function, unlike boolean intersection methods. The highlighted differences demonstrate that isolines extracted by our method curve naturally within triangles. Additional examples can be found in the supplementary material.*

For this series of experiments, the number of sampling points was fixed at 20. The findings, illustrated in Figure 7 and Table 3, underscore our algorithm's superiority in terms of both precision and time efficiency. Therefore, when contrasted with the boolean intersection approach, our method offers numerous advantages. It not only obviates the need for converting the scalar function into a mesh but also effectively handles a broad spectrum of complex meshes and challenging functions.

Specifically, in the initial PLM computation, we placed 5 and 10 sampling points on each edge via ε-sampling strategy. The subsequent computation involved several smaller triangular units generated by Delaunay triangulation. For our method, we sampled only 5 points per mesh edge. We then applied our refinement framework to both the PLM and our initial results.

4.4. Validation of Feature Preservation

Both the power diagram and the refinement process aim to characterize the non-differentiable properties of the function. Notably, the latter can serve as a post-processing tool integrated with PLM. To evaluate its feature-preserving effectiveness, we compared our method with PLM.

As shown in Figure 8, our method present a faithful initial curve with feature points that are missing in PLM. Although our fine-tuning process successfully recovers these missing points, PLM is computationally expensive due to its complex linear approximation. In contrast, our method benefits from a reliable initial result, leading to faster convergence and reduced computational time.

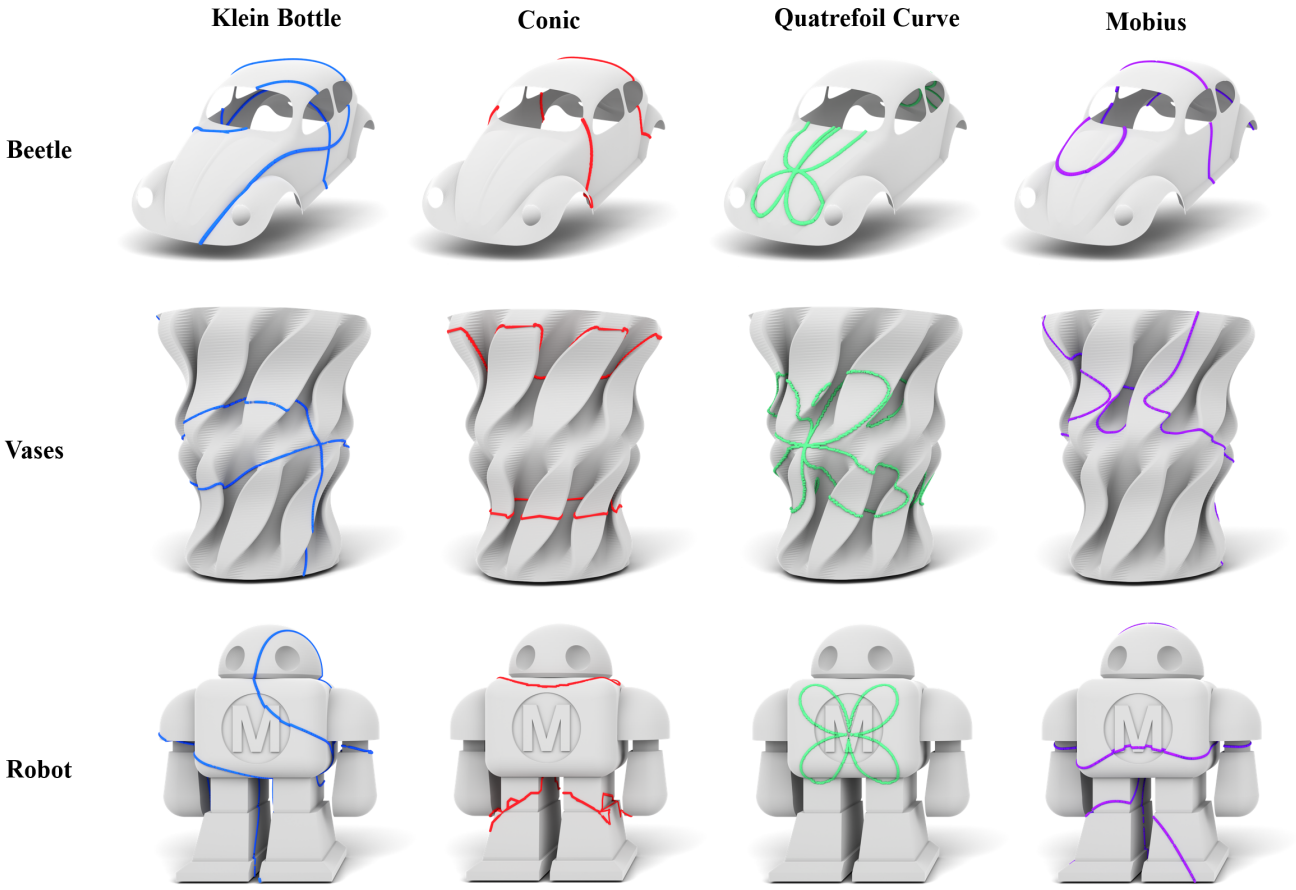


Figure 7: *Our algorithm is not limited by the type of input mesh and can accurately extract isolines, even with non-smooth functions. Additional examples can be found in the supplementary material.*

4.5. Ablation Study

The efficiency of our algorithm is influenced by two critical factors: the model's complexity, as reflected by the number of faces, and the number of sampling points employed. To determine the effects of these elements, we conducted tests to measure both the extraction time and the isolines' accuracy across models with varied resolutions and sampling point counts, as depicted in Figure 9 and Figure 10. The results indicate that, while the accuracy of the extracted isolines does depend on these factors, the isolines consistently appear as relatively smooth and high-fidelity curves. This underscores the algorithm's resilience to variations in model complexity and sampling density.

5. Extension

Visualizing isolines is essential in many geometry processing tasks, especially for analyzing special scalar fields like discrete signed distance fields or geodesic distance fields. However, since these fields cannot be represented in closed form, accurately computing isolines using boolean intersection methods is impossible. By leveraging gradient information, our method can reliably contour isolines on surface, making it a valuable tool in these scenarios. In this section, we present several applications to demonstrate how our method contours some common scalar fields on a surface.

The first set of experiments involves contouring the discrete distance field on a surface. We began by delineating the isolines of one mesh onto another mesh's signed distance field, as depicted in Figure 11. Then we extended this intriguing task to the unsigned distance field of the point cloud, as shown in Figure 12.

The second set of experiments focuses on contouring the geodesic field on a surface. Using the algorithm introduced by [XW09], we computed exact geodesic distances from source points on a surface mesh. Figure 13 illustrates the isolines of geodesic distance field generated with one and three source points. The zoomed-in results reveal smooth curves across the triangles, highlighting the reliable approximation achieved by our method.

6. Conclusion and Future Works

Isoline extraction from surface-based functions plays a crucial role in digital geometry processing applications. This paper presents the power diagram as an effective technique for swiftly isolating these isolines. Our research reveals that each site can define a circle that

Table 2: *The error and running time of various implicit functions and isovalues (c) on different models compared to the boolean intersection method [ZGZJ16], where the bold values indicate the best results. E_{avg} and E_{max} are scaled by* $\mathbf{10^3}$.

Model	Metrics	Test function	Ellipsoid			Torus			Bean		
		Constant c	0.01	0.02	0.03	2	3	4	-0.5	0.5	1.5
Bunny (7K faces)	E_{avg}	MC+Boolean	0.064	0.113	0.133	**5.557**	6.564	9.989	14.262	17.650	18.890
		Ours	**0.049**	**0.093**	**0.123**	5.692	**4.902**	**5.474**	**13.967**	**15.893**	**15.956**
	E_{max}	MC+Boolean	1.677	2.840	**0.661**	**52.512**	89.541	271.799	**168.382**	278.013	293.171
		Ours	**0.610**	**0.758**	1.250	65.239	**79.194**	**77.771**	170.530	**200.465**	**225.476**
	Time/s	MC+Boolean	2.50/0.72	2.56/1.04	2.54/1.10	2.66/1.20	2.72/1.34	2.73/1.47	2.12/0.90	2.16/0.66	2.17/0.77
		Ours	**0.56**	**0.57**	**0.57**	**0.61**	**0.60**	**0.61**	**0.29**	**0.30**	**0.30**
Bear (20K faces)	E_{avg}	MC+Boolean	0.081	0.116	0.663	14.519	6.282	9.581	18.579	20.924	22.640
		Ours	**0.046**	**0.063**	**0.095**	**5.008**	**5.728**	**8.273**	**10.963**	**16.716**	**17.179**
	E_{max}	MC+Boolean	1.360	1.412	1.120	**27.104**	27.527	**26.690**	78.968	**84.534**	129.041
		Ours	**0.316**	**0.501**	**0.761**	40.500	**27.440**	34.651	**67.876**	122.883	**125.117**
	Time/s	MC+Boolean	2.54/0.80	2.53/1.07	2.59/1.33	2.67/1.37	2.72/1.58	2.76/1.62	2.13/0.62	2.17/0.88	2.13/1.07
		Ours	**1.51**	**1.52**	**1.54**	**1.64**	**1.65**	**1.65**	**0.78**	**0.79**	**0.78**
Lucy-Cat (8K faces)	E_{avg}	MC+Boolean	0.074	0.118	0.124	10.514	13.740	36.102	17.342	18.843	31.514
		Ours	**0.063**	**0.083**	**0.069**	**7.790**	**10.317**	**10.123**	**12.276**	**14.918**	**23.679**
	E_{max}	MC+Boolean	0.236	**0.312**	**0.398**	**24.381**	**48.807**	112.767	125.805	**153.064**	**176.214**
		Ours	**0.232**	0.595	0.608	69.578	88.012	**109.586**	**119.838**	161.864	212.887
	Time/s	MC+Boolean	2.49/1.07	2.58/1.11	2.60/1.23	2.65/1.46	2.75/1.89	2.76/1.44	2.13/0.70	2.17/0.90	2.17/1.11
		Ours	**0.65**	**0.66**	**0.64**	**0.68**	**0.68**	**0.67**	**0.33**	**0.34**	**0.32**

Table 3: *The errors and execution time statistics for particular implicit functions on various models are reported. The values of E_{avg} and E_{max} are presented after being scaled by* $\mathbf{10^3}$. *It is noteworthy that the boolean intersection approach fails in this case.*

Type	Model	Test Function (isovalues are all zero)											
		Klein Bottle			Conic			Quatrefoil Curve			Mobius		
		Time/s	E_{avg}	E_{max}	Time/s	E_{avg}	E_{max}	Time/s	E_{avg}	E_{max}	Time/s	E_{avg}	E_{max}
Non-closed	Holes (26K faces)	1.47	19.660	78.231	2.11	0.981	5.034	1.56	0.004	0.099	0.99	2.306	59.512
	Beetle (38K faces)	9.68	20.860	260.943	9.20	0.725	2.428	3.86	0.006	0.191	3.60	1.438	14.251
Non-PWN	Robot (30K faces)	4.37	17.453	2349.340	7.23	7.637	59.643	5.92	0.016	0.809	2.51	7.017	376.687
	Vases (32K faces)	6.10	59.090	1304.366	8.10	2.820	11.609	6.71	0.013	0.406	1.43	5.224	159.578
Self-intersecting	Decorated-box (5K faces)	0.47	99.118	2646.850	0.21	5.232	23.248	0.11	0.011	0.266	0.09	8.385	65.592
	Carp-Muscle (40K faces)	4.22	14.872	1267.097	4.62	1.201	8.185	3.40	0.034	0.320	1.83	1.754	155.460

touches the actual isoline, and the bisector of these circles forms a polygonal curve. This curve serves as a valuable indicator for tracing the true isoline. We introduce several optimization strategies aimed at improving both the speed and numerical precision of the process. Additionally, our algorithm is designed to be fully parallelizable at the triangle level, significantly enhancing its efficiency. Considering the fundamental nature of isoline extraction in computer graphics, we anticipate that our algorithm will be of substantial utility.

This study primarily focuses on the extraction of 2D isolines. Looking ahead, it would be intriguing to adapt this methodology for 3D applications, enabling the rapid extraction of polygonal surfaces from implicit representations.

References

[ACK01] AMENTA N., CHOI S., KOLLURI R. K.: The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications* (2001), pp. 249–266. 3

[AK00] AURENHAMMER F., KLEIN R.: Voronoi diagrams. *Handbook of computational geometry 5*, 10 (2000), 201–290. 3

[Att14] ATTENE M.: Direct repair of self-intersecting meshes. *Graphical Models 76*, 6 (2014), 658–668. 2

[Aur87] AURENHAMMER F.: Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing 16*, 1 (1987), 78–96. 3

[BGF15] BARKI H., GUENNEBAUD G., FOUFOU S.: Exact, robust, and efficient regularized booleans on general 3d meshes. *Computers & Mathematics with Applications 70*, 6 (2015), 1235–1254. 2

[BJB*11] BHATIA H., JADHAV S., BREMER P.-T., CHEN G., LEVINE J. A., NONATO L. G., PASCUCCI V.: Flow visualization with quantified spatial and temporal errors using edge maps. *IEEE Transactions on Visualization and Computer Graphics 18*, 9 (2011), 1383–1396. 1

[CLSA20] CHERCHI G., LIVESU M., SCATENI R., ATTENE M.: Fast and robust mesh arrangements using floating-point arithmetic. *ACM Transactions on Graphics (TOG) 39*, 6 (2020), 1–16. 2

[CPAL22] CHERCHI G., PELLACINI F., ATTENE M., LIVESU M.: In-

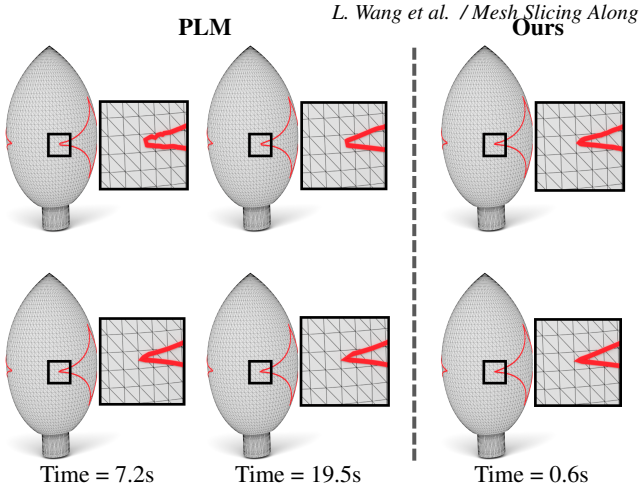


Figure 8: *Visual comparison feature recovery between PLM and our method applied to the star function on a model with 8K faces. The first row shows the initial results, while the second row displays the results after applying our refinement process. Despite an increased number of sampling points, PLM struggles to recover feature points. In contrast, our initial results using the power diagram effectively addresses this issue. By employ our fine-tuning framework, the missing features in PLM are successfully reconstructed.*

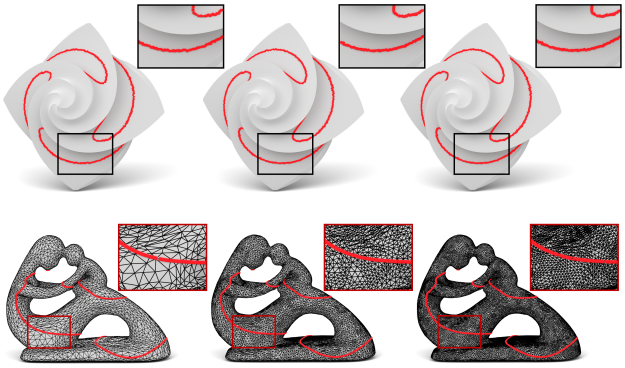


Figure 9: *The first row displays the result (red curve) of our algorithm applied to the cross torus function on the octa-flower model (20K faces). As the number of sampling points increases from left to right, the highlighted differences illustrate that the curve becomes smoother. The second row showcases the result (red curve) of our algorithm applied to the cross torus function on the fertility model (with a fixed number of sampling points set to 5). As the number of faces increases from left to right through mesh subdivision, the curve evolves accordingly.*

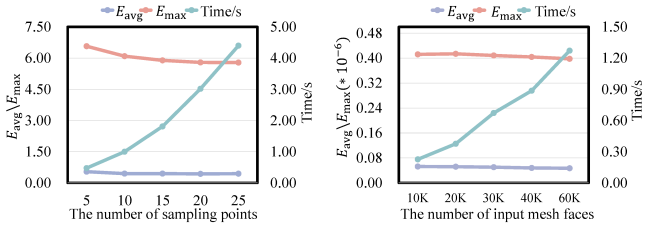


Figure 10: *The statistics result about accuracy and runtime performance corresponding with Figure 9.*

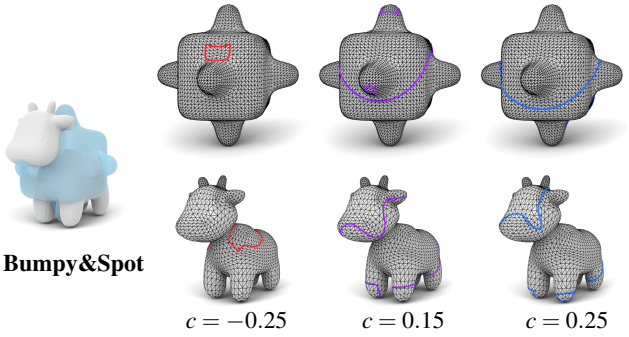


Figure 11: *Visualized isoline results extracted from one mesh based on the signed distance field (SDF) of another mesh. The first row displays isolines extracted from the SDF of the spot model (6K faces) on the bumpy model (2.5K faces), while the second row showcases the reverse scenario.*

teractive and robust mesh booleans. *arXiv preprint arXiv:2205.14151* (2022). 2

[CTFZ22] CHEN Z., TAGLIASACCHI A., FUNKHOUSER T., ZHANG H.: Neural dual contouring. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 1–13. 2

[CZ21] CHEN Z., ZHANG H.: Neural marching cubes. *ACM Transactions on Graphics (TOG) 40*, 6 (2021), 1–15. 2

[JKSH13] JACOBSON A., KAVAN L., SORKINE-HORNUNG O.: Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–12. 2

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 339–346. 2

[JP*18] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 5

[KBSS01] KOBBELT L. P., BOTSCH M., SCHWANECKE U., SEIDEL H.-P.: Feature sensitive surface extraction from volume data. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 57–66. 2

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics 21*, 4 (1987), 163–169. 2, 5

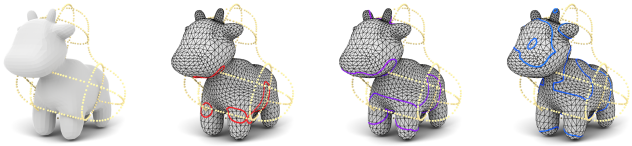
[LCT11] LIU Y.-J., CHEN Z., TANG K.: Construction of iso-contours, bisectors, and voronoi diagrams on triangulated surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 8 (2011), 1502–1517. 2

[MLP21] MANCINELLI C., LIVESU M., PUPPO E.: Practical computation of the cut locus on discrete surfaces. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 261–273. 2

[MMP87] MITCHELL J. S., MOUNT D. M., PAPADIMITRIOU C. H.: The discrete geodesic problem. *SIAM Journal on Computing 16*, 4 (1987), 647–668. 2

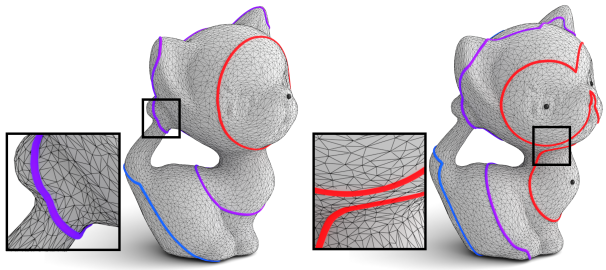
[MP22] MANCINELLI C., PUPPO E.: Vector graphics on surfaces using straightedge and compass constructions. *Computers & Graphics 105* (2022), 46–56. 2

[MWS*21] MUZAHID A. A. M., WAN W., SOHEL F., WU L., HOU L.: Curvenet: Curvature-based multitask learning deep networks for 3d



$c = 0.1$ $c = 0.2$ $c = 0.3$

Figure 12: *Visualized results of extracted isolines based on the unsigned distance field (UDF) of a given point cloud (0.5K, rendered in yellow).*



one source three sources

Figure 13: *Visualized isolines of three isovalues extracted from the geodesic distance field on the kitten model (13K).*

object recognition. *IEEE/CAA Journal of Automatica Sinica 8*, 6 (2021), 1177–1187. doi:10.1109/JAS.2020.1003324. 1

[MXT*21] MENG W., XIN S., TU C., CHEN S., HE Y., WANG W.: Geodesic tracks: Computing discrete geodesics with track-based steiner point propagation. *IEEE Transactions on Visualization and Computer Graphics 28*, 12 (2021), 4887–4901. 2

[NAT90] NAYLOR B., AMANATIDES J., THIBAULT W.: Merging bsp trees yields polyhedral set operations. *ACM Siggraph Computer Graphics 24*, 4 (1990), 115–124. 2

[SJK10] SCHIFKO M., JÜTTLER B., KORNBERGER B.: Industrial application of exact boolean operations for meshes. In *Proceedings of the 26th Spring Conference on Computer Graphics* (2010), pp. 165–172. 2

[SSK*05] SURAZHSKY V., SURAZHSKY T., KIRSANOV D., GORTLER S. J., HOPPE H.:. *ACM Trans. Graph. 24*, 3 (2005), 553–560. 2

[The24] THE CGAL PROJECT: *CGAL User and Reference Manual*, 5.6.1 ed. CGAL Editorial Board, 2024. URL: https://doc.cgal.org/5.6.1/Manual/packages.html. 4, 5

[TNWK22] TRETTNER P., NEHRING-WIRXEL J., KOBBELT L.: Ember: exact mesh booleans via efficient & robust local arrangements. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 1–15. 2

[WS19] WANG Y., SOLOMON J.: Intrinsic and extrinsic operators for shape analysis. In *Handbook of numerical analysis*, vol. 20. Elsevier, 2019, pp. 41–115. 1

[WWWG22] WANG N., WANG B., WANG W., GUO X.: Computing medial axis transform with feature preservation via restricted power diagram. *ACM Transactions on Graphics (TOG) 41*, 6 (2022), 1–18. 3

[XW09] XIN S.-Q., WANG G.-J.: Improving chen and han's algorithm on the discrete geodesic problem. *ACM Transactions on Graphics (TOG) 28*, 4 (2009), 1–8. 7

[YLL*09] YAN D.-M., LÉVY B., LIU Y., SUN F., WANG W.: Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In *Computer graphics forum* (2009), vol. 28, Wiley Online Library, pp. 1445–1454. 3

[YMS*21] YAN L., MASOOD T. B., SRIDHARAMURTHY R., RASHEED F., NATARAJAN V., HOTZ I., WANG B.: Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 599–633. 1

[ZGZJ16] ZHOU Q., GRINSPUN E., ZORIN D., JACOBSON A.: Mesh arrangements for solid geometry. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–15. 2, 5, 8