Constructing Intrinsic Delaunay Triangulations from the Dual of Geodesic Voronoi Diagrams

YONG-JIN LIU, DIAN FAN, and CHUN-XU XU
Tsinghua University
and
YING HE
Nanyang Technological University

Intrinsic Delaunay triangulation (IDT) naturally generalizes Delaunay triangulation from $\mathbb{R}^2$ to curved surfaces. Due to many favorable properties, the IDT whose vertex set includes all mesh vertices is of particular interest in polygonal mesh processing. To date, the only way for constructing such IDT is the edge-flipping algorithm, which iteratively flips non-Delaunay edges to become locally Delaunay. Although this algorithm is conceptually simple and guarantees to terminate in finite steps, it has no known time complexity and may also produce triangulations containing faces with only two edges. This article develops a new method to obtain proper IDTs on manifold triangle meshes. We first compute a geodesic Voronoi diagram (GVD) by taking all mesh vertices as generators and then find its dual graph. The sufficient condition for the dual graph to be a proper triangulation is that all Voronoi cells satisfy the so-called closed ball property. To guarantee the closed ball property everywhere, a certain sampling criterion is required. For Voronoi cells that violate the closed ball property, we fix them by computing topologically safe regions, in which auxiliary sites can be added without changing the topology of the Voronoi diagram beyond them. Given a mesh with n vertices, we prove that by adding at most $O(n)$ auxiliary sites, the computed GVD satisfies the closed ball property, and hence its dual graph is a proper IDT. Our method has a theoretical worst-case time complexity $O(n^2 + tn \log n)$, where t is the number of obtuse angles in the mesh. Computational results show that it empirically runs in linear time on real-world models.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*

General Terms: Algorithms

Additional Key Words and Phrases: Intrinsic Delaunay triangulation, geodesic Voronoi diagram, duality, the closed ball property

ACM Reference Format:

Yong-Jin Liu, Dian Fan, Chun-Xu Xu, and Ying He. 2017. Constructing intrinsic Delaunay triangulations from the dual of geodesic Voronoi diagrams. ACM Trans. Graph. 36, 2, Article 15 (April 2017), 15 pages.
DOI: http://dx.doi.org/10.1145/2999532

This work was supported by the National Key Research and Development Plan (2016YFB1001202), the Natural Science Foundation of China (61432003, 61521002, 61661130156), and a Royal Society-Newton Advanced Fellowship.

Authors' addresses: Y.-J. Liu, D. Fan, and C.-X. Xu, TNList, Department of Computer Science and Technology, Tsinghua University, Beijing, China; emails: liuyongjin@tsinghua.edu.cn, fand14@mails.tsinghua.edu.cn, xucx12@mails.tsinghua.edu.cn; Y. He, School of Computer Engineering, Nanyang Technological University, Singapore; email: yhe@ntu.edu.sg.

DOI: http://dx.doi.org/10.1145/2999532

1. INTRODUCTION

A Delaunay triangulation for a set $\mathbf{P}$ of points in $\mathbb{R}^2$ is a triangulation such that no point of $\mathbf{P}$ is inside the circumcircle of any triangle in the triangulation. It is well known that Delaunay triangulations tend to avoid skinny triangles, since they maximize the minimum angle of all angles of the triangles in the triangulation. Although Delaunay triangulations in Euclidean spaces are well understood [Okabe et al. 2000], intrinsic Delaunay triangulations (IDTs) on manifold domains have received less attention. Using the closed ball property, Edelsbrunner and Shah [1997] and Dyer et al. [2008] proposed adaptive sampling criteria for constructing an intrinsic Voronoi diagram and its dual Delaunay triangulation on smooth 2-manifolds. Recently, Boissonnat et al. [2013] proposed an algorithm for constructing IDT on smooth closed submanifolds of Euclidean spaces. Both methods are based on *convex neighborhood*, which, in general, is an extremely small region around a point on smooth manifolds. As a result,despite their important theoretical values, they are not practical for piecewise linear surfaces, which are dominant in digital geometry processing.

In this article, we focus on a special type of IDT defined on manifold triangle meshes, where the IDT's vertex set includes all mesh vertices. Such an IDT was first studied by Rivin [1994], who defined IDT edges using geodesic paths and replaced circumcircles with geodesic circumcircles. As pointed out by Bobenko and Springborn [2007], this kind of IDT is highly desired over digital geometry processing, as the classic cotangent Laplace-Beltrami operator (LBO) has nonnegative weights w_{ij} if and only if the underlying triangulation is Delaunay.

To date, the only practical algorithm for computing the IDTs mentioned previously is the edge-flipping algorithm [Bobenko and Springborn 2007; Fisher et al. 2007; Indermitte et al. 2001], which iteratively flips the non-Delaunay edge to become locally Delaunay. Bobenko and Springborn [2007] proved that the edge-flipping algorithm terminates in a finite number of steps, and therefore the intrinsic Delaunay tessellation exists. They also proved the uniqueness

of the intrinsic Delaunay tessellation. The edge-flipping algorithm is conceptually simple and easy to implement. However, it has no known time complexity and may also produce self-loops, leading to faces with only two edges.

This article presents a new method with bounded time complexity that can guarantee the computed IDTs are free of self-loops. Our idea is to construct IDT via the dual graph of geodesic Voronoi diagrams (GVDs). Dyer et al. [2007b] established the duality between GVD and IDT on 2-manifold meshes and showed that if a GVD satisfies the closed ball property [Edelsbrunner and Shah 1997], then the dual graph is a *proper*[1] IDT.

Our algorithm is the first of its kind to compute GVDs on meshes that are guaranteed to satisfy the closed ball property. Given a mesh with n vertices, our method first computes the GVD by taking all vertices as sites. For each Voronoi cell that violates the closed ball property, it computes a topologically safe region, in which auxiliary sites can be added without changing the topology beyond that cell. We prove that by adding at most $O(n)$ auxiliary sites, the computed GVD satisfies the closed ball property, and hereby it has a proper dual IDT. Moreover, thanks to the bounded time complexity of computing GVD [Liu et al. 2011; Liu and Tang 2013], our method has a theoretical worst-case time complexity $O(n^2+tn\log n)$, where t is the number of obtuse angles in the mesh. We observe that many real-world models are far from their Delaunay triangulations, and thus it takes the edge-flipping algorithm many iterations to converge. In contrast, the performance of our method is not sensitive to the number of non-Delaunay edges and empirically runs in linear time $O(n)$ on these models.

In addition to our theoretical contributions, the proper IDTs produced by our method are favorable to practical applications. Take the conformal parameterization [Fisher et al. 2007] as an example. The IDTs produced by the edge-flipping algorithm may contain degree-1 and/or -2 vertices. As a result, the injectivity of a discrete conformal mapping is lost, leading to rendering artifacts in texture mapping and also large local distortion. In contrast, our IDTs are proper and can be represented by any simplicial complex-based data structure, making them an ideal input to the existing digital geometry processing pipeline.

The rest of the article is organized as follows. Section 2 reviews the mathematical background and highlights the fundamental differences between Delaunay triangulations in $\mathbb{R}^2$ and IDTs. Section 3 introduces the key ideas of our method. Section 4 presents our algorithm in detail, followed by the correctness proof and complexity analysis in Section 5. Section 6 reports the experimental results and compares IDTs to other Delaunay structures. Finally, Section 7 concludes the article. To ease reading, we list the main notations in Table I and present the long proofs in the supplementary material.

Table I. Main Notations

M	Manifold triangle mesh
V, E, F	Sets of vertices, edges, and faces of M
$n(=\|V\|)$	Number of vertices
$v, v_i, v_j, \ldots$	Mesh vertices
$e, e_{ij} = \{v_i, v_j\}$	Mesh edges
$f, f_{ijk} = \{v_i, v_j, v_k\}$	Triangular faces
$p, q, p_i, q_i, \ldots$	Points on M
$\gamma(p, q)$	Geodesic path between p and q
$d(p, q)$	Geodesic distance between p and q
$D(p, r)$	Geodesic disk of radius r centered at p
$GVD(P)$	Geodesic Voronoi diagram of a set P of sites
$\Gamma(B_P, C_P)$	Topological representation of $GVD(P)$
B_P	Set of symbolic Voronoi edges
C_P	Set of symbolic Voronoi cells
$\beta(p, q)$	Bisector of sites p and q
$b(p, q)$	Symbolic representation of a Voronoi edge
$\widetilde{b}(p, q)$	Geometric realization of $b(p, q)$
$C(p_i)$	Symbolic representation of a Voronoi cell
$\widetilde{C}(p_i)$	Geometric realization of $C(p_i)$
$pb(p)$	Symbolic representation of a pseudobisector
$\widetilde{pb}(p)$	Geometric realization of $pb(p)$
$\delta_{\max}(\widetilde{C}(p))$	Maximal value of δ so that the δ-offset $\Theta(\widetilde{C}(p), \delta)$ does not contain any site in P and any Voronoi vertex that is not in $\widetilde{C}(p)$
$\delta_{\max}(GVD(P))$	Minimal $\delta_{\max}$ for all Voronoi cells
$\varepsilon(\delta)$	Radius of the ε-neighborhood $D(p, \varepsilon)$ for any $0 < \delta \leq \delta_{\max}(GVD(P))$
$\hat{\delta}_{\max}(GVD(P))$	Value for a strong δ-offset defined in Equation (S6) in the supplementary material
Single-order neighbor	Two sites whose Voronoi cells share exactly one Voronoi edge
Multiple-order neighbor	Two sites whose Voronoi cells share at least two Voronoi edges
$IDT(M)$	Intrinsic Delaunay triangulation of M
Ξ	Set of g-edges
Γ	Set of geodesic triangles
$\xi \in \Xi$	g-Edges
$\tau, \tau_i, \tau_j \in \Gamma$	Geodesic triangles

2. PRELIMINARIES

2.1 Geodesic Paths, Geodesic Triangles, and Geodesic Circles

Let M be a manifold triangle mesh, and let V, E, F be the set of vertices, edges, and faces of M, respectively. Every interior point of M has a neighborhood that is isometric to the neighborhood of either a point on a Euclidean plane or the apex of a Euclidean cone.

Consider two points $p, q \in M$. A geodesic path between p and q, denoted by $\gamma(p, q)$, is the locally shortest path between them. Mitchell et al. [1987] showed that the general form of a geodesic path γ is an alternating sequence of vertices (possibly empty) and edge sequences such that the unfolded image of the path γ along any edge sequence is a straight line segment and the angle of the path at a vertex is greater than or equal to π. The vertices with cone angles more than 2π are called *saddle vertices*, which play a critical role in geodesic computation [Ying et al. 2013; Xu et al. 2015]. We denote by $d(p, q)$ the geodesic distance between p and q, and $\beta(p, q)$ the bisector of p and q.

In a smooth, simply connected surface S with negative Gaussian curvature everywhere, the geodesic path $\gamma(p, q)$ is unique for any pair of points $p, q \in S$. However, in general, geodesics are not unique for regions with positive Gaussian curvature and/or nontrivial topology (Figure 1(a) through (c)). It is worth noting that discrete geodesic paths have similar properties, as the Gauss-Bonnet theorem also holds on piecewise linear surfaces. A geodesic γ joining two points p and q is minimal if its length is smaller than or equal to the length of any curve joining p and q.

Definition 1 (*Geodesic Triangle*). A geodesic triangle $\tau \subset M$ is a simply connected domain whose boundary $\partial\tau$ has three geodesic paths. Each geodesic path is called a *g-edge* and the endpoints of a g-edge are called *g-vertices*.

[1] A triangulation is *proper* if it is a realization of a simplicial complex. See Dyer et al. [2007b]. All Delaunay triangulations in $\mathbb{R}^2$ are proper.

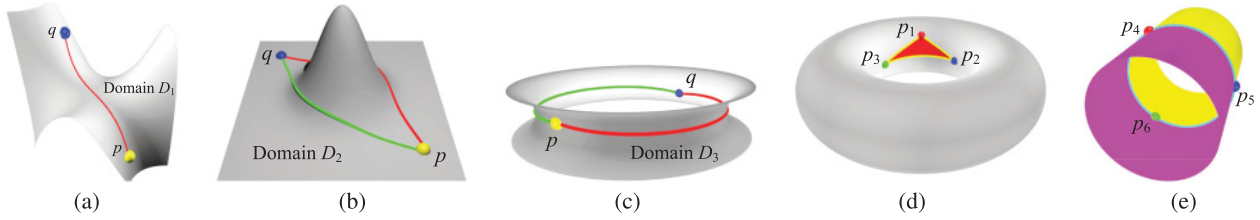
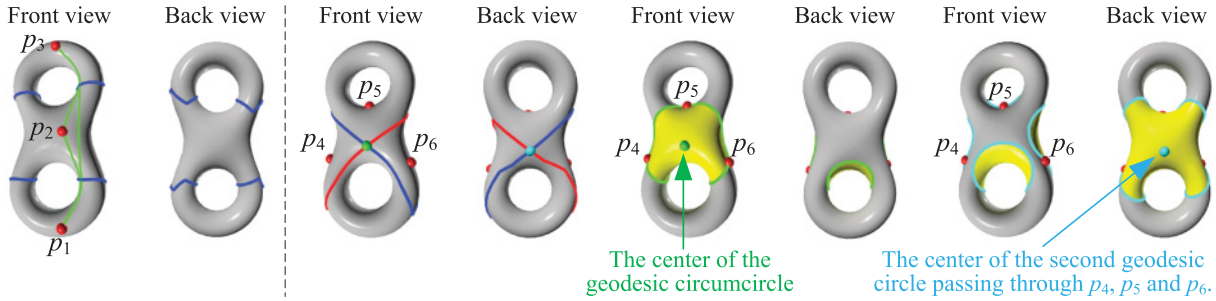


Fig. 1. Geodesic paths and geodesic triangles. (a) Domain D_1 is simply connected and has nonpositive Gaussian curvature everywhere. There is a unique geodesic path between any pair of distinct points in D_1. (b) Although domain D_2 is also simply connected, it has a part with positive Gaussian curvatures. As a result, the geodesic paths in D_2 are not unique. (c) Domain D_3 has negative Gaussian curvature everywhere. However, as its fundamental group is nontrivial, the geodesic paths in D_3 are not unique either. (d) The red region $\tau = (p_1, p_2, p_3)$ is a geodesic triangle on the torus M. However, the complement $M \setminus \tau$, although having three geodesic sides, is not a geodesic triangle, since it is not simply connected. (e) Three geodesic paths $\gamma(p_4, p_5)$, $\gamma(p_5, p_6)$ and $\gamma(p_6, p_4)$ wrap the cylinder. Although each colored region has three geodesic sides, they are not geodesic triangles, as it does not bound a simply connected region.



(a) $\triangle p_1 p_2 p_3$ has no circumcircle (b) Two distinct geodesic circles passing through p_4, p_5 and p_6

Fig. 2. Geodesic circumcircles. (a) Geodesic triangle $\triangle p_1 p_2 p_3$ has no geodesic circumcircle, since the bisector $\beta(p_1, p_2)$ (shown in blue) does not meet the other bisector $\beta(p_2, p_3)$. (b) Since $\beta(p_4, p_5)$ (blue) and $\beta(p_5, p_6)$ (red) intersect twice, there are two geodesic circles passing through three points p_4, p_5, and p_6; only one of them is the geodesic circumcircle of the geodesic triangle $\triangle p_4 p_5 p_6$.

On $\mathbb{R}^2$, any three intersecting lines form a triangle. However, not all three intersecting geodesics on a mesh form a geodesic triangle (see Figure 1(d) through (e)).

Definition 2 (*Geodesic Disk and Geodesic Circle*). A geodesic disk of radius r centered at a point $p \in M$, denoted by $D(p, r)$, consists of all points whose geodesic distances to p does not exceed r (i.e., $D(p, r) = \{q \in M : d(p, q) \leq r\}$). If a geodesic disk $D(p, r)$ is simply connected, its boundary $\partial D(p, r)$ is called a *geodesic circle*. A geodesic circle ∂D that circumscribes a geodesic triangle τ, $\tau \subset D$, is called a *geodesic circumcircle*.

In $\mathbb{R}^2$, three distinct points not lying on a line define a nondegenerate triangle that has a unique circumcircle. However, on a curved surface, not every geodesic triangle has a geodesic circumcircle. For the case that the geodesic circumcircle exists, there may be multiple geodesic circles passing through the tree vertices of the geodesic triangle (Figure 2).

2.2 IDTs on Meshes

The definition of IDT on 2-manifold meshes is due to Rivin [1994], who generalized the planar Delaunay condition (i.e., a circle circumscribing any Delaunay triangle does not contain any input points in its interior), by requiring the empty geodesic circumcircle property.

Definition 3 (*Intrinsic Delaunay Triangulation*). The intrinsic Delaunay triangulation (IDT) on M, denoted by $IDT(M) = (V, \Xi, \Gamma)$, is a tessellation of M such that

—the vertex set of $IDT(M)$ equals V;
—every edge ξ in Ξ is a geodesic path on M (i.e., a g-edge); and
—each face $\tau \in \Gamma$ is a geodesic triangle, which has a geodesic circumcircle containing no mesh vertices in its interior.

Then Dyer et al. [2007b] defined *proper* IDT as follows.

Definition 4 (*Proper IDT*). An IDT on M is proper if it is the realization of a simplicial complex.

Bobenko and Springborn [2007] showed that the edge-flipping algorithm terminates in finite steps, implying the existence of IDTs. They also proved the uniqueness of Delaunay tessellation (whose faces are general but not always triangular). The Delaunay triangulation can be obtained by triangulating the nontriangular faces.

2.3 Geodesic Voronoi Diagrams

Definition 5 (*Geodesic Voronoi Diagram*). Let $P = \{p_1, p_2, \ldots, p_m\}$ be a set of points on M. For a site p_i, the Voronoi cell $\widetilde{C}(p_i)$ consists of all points whose geodesic distances to p_i are less than or equal to their distances to any other site (i.e., $\widetilde{C}(p_i) = \{q \in M : d(p_i, q) \leq d(p_j, q), \forall i \neq j\}$). The geodesic Voronoi diagram (GVD) of P is the set $\{\widetilde{C}(p_1), \widetilde{C}(p_2), \ldots, \widetilde{C}(p_m)\}$.

In Sections 3 through 5, we assume that the points of P are in *general* positions—that is, no four or more points are on the same geodesic circumcircle, and hereby the intersection of four or more

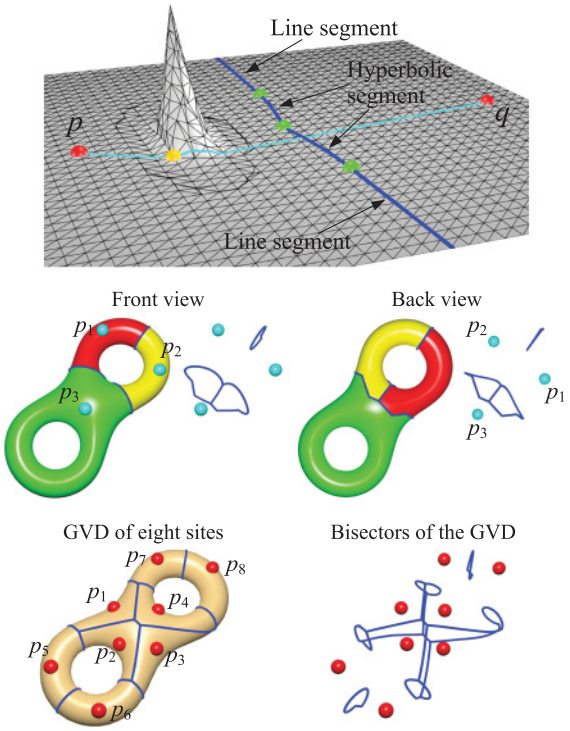


Fig. 3. Bisectors and Voronoi cells on a 2-manifold mesh are significantly different from their Euclidean counterparts. Row 1: There is a saddle vertex (yellow) on the geodesic path $\gamma(p, q)$ (cyan). As a result, the bisector $\beta(p, q)$ (blue) consists of both line segments and hyperbolic segments. Row 2: $C(p_3)$ (green) is of genus-1, and $C(p_1)$ (red) and $C(p_2)$ (yellow) are of genus-0 but with multiple boundaries. Row 3: A GVD with eight sites p_i, $i = 1, \ldots, 8$ on the two-hole torus model. Note that each of $C(p_5)$, $C(p_6)$, $C(p_7)$ and $C(p_8)$ has multiple boundaries. $C(p_1)$ and $C(p_2)$ share two common Voronoi edges, and so do the other pairs of Voronoi cells $(C(p_2), C(p_3))$, $(C(p_3), C(p_4))$, $(C(p_1), C(p_4))$ and $(C(p_2), C(p_4))$.

Voronoi cells is an empty set. In the supplementary material, we extend the proof of main results in this article to include degenerate cases. In Section 6.3, the implementation issue of handling degenerate cases is discussed.

It is well known that Voronoi cells in $\mathbb{R}^2$ are convex and simply connected. However, this property does not hold for GVDs [Liu et al. 2011; Xu et al. 2014; Liu 2015]. Although a geodesic Voronoi cell is still connected, it may contain multiple boundaries and/or handles (Figure 3 (rows 2 and 3)). Moreover, bisectors on triangle meshes generally consist of line segments and hyperbolic segments [Liu et al. 2011] (see Figure 3 (row 1)). For a special case that P includes all mesh vertices, all Voronoi edges of $GVD(P)$ are line segments [Dyer et al. 2007b; Liu et al. 2011].

2.4 The Closed Ball Property and Dual Proper IDT

In Euclidean domains, the dual graph of the Voronoi diagram of a point set P is a proper Delaunay triangulation; therefore, one can adopt the algorithms for constructing a Voronoi diagram to obtain Delaunay triangulation and vice versa. However, in general manifold domains, the dual proper IDT exists only if the GVD satisfies the closed ball property [Edelsbrunner and Shah 1997].

The closed ball property consists of three conditions:

(1) *Disk condition*: Each Voronoi cell is homeomorphic to a planar disk.
(2) *Two-cell intersection condition*: The intersection of any two Voronoi cells is either empty or a single Voronoi edge.
(3) *Three-cell intersection condition*: The intersection of any three Voronoi cells is either empty or a single Voronoi vertex.

Throughout this article, we assume that the point set P includes all mesh vertices in M and the points of P satisfy the general position condition. Dyer et al. [2007b] showed that if there are at least four distinct sites in $GVD(P)$ and both the disk condition and 2-cell intersection conditions are satisfied, then the 3-cell intersection condition is redundant.

THEOREM 6 (DYER ET AL. [2007B]). *The IDT is proper if and only if the dual GVD satisfies the closed ball property.*

By Theorem 6, the key to obtain a proper IDT is to construct a GVD satisfying the closed ball property. However, Dyer et al. [2007b] did not provide any algorithm. Our work fills the gap by developing a practical algorithm to compute such GVDs.

Note that Edelsbrunner and Shah [1997] introduced the closed ball property for triangulating abstract topological spaces. Recently, Dyer et al. [2015] studied a natural intrinsic definition of geometric simplices in Riemannian manifolds of arbitrary finite dimension and exploited these simplices to obtain criteria for triangulating compact Riemannian manifolds.

3. KEY IDEAS

3.1 Motivation

Flipping edges and computing the dual graph of Voronoi diagrams are two commonly used techniques for constructing Delaunay triangulations in $\mathbb{R}^2$. For IDTs, the edge-flipping algorithm [Indermitte et al. 2001; Fisher et al. 2007] is the only known algorithm so far. It is conceptually simple and easy to implement; however, it does not have a known time complexity and the computed IDT may not be proper, due to faces with only two edges. Motivated by Theorem 6, we take the other direction to construct IDT—that is, computing the dual of GVDs satisfying the closed ball property.

We consider only the disk condition and the 2-cell intersection condition in our algorithm, as all models we are dealing with have more than four vertices. Observe that the closed ball property often fails in regions with low sampling density. Thus, our idea is to add auxiliary sites to those Voronoi cells that violate the disk condition and 2-cell intersection condition. However, one has to be careful when adding an auxiliary site to a Voronoi diagram. Let $V(q)$ be a Voronoi cell violating the closed ball property. Adding a misplaced auxiliary site s to $V(q)$, although fixing the original Voronoi cell $V(q)$, may cause problems to the new Voronoi cell $V(s)$. Two such examples are presented in Figures 4 and 5.

To avoid the endless loop for adding auxiliary sites and also ease analysis, we introduce a novel concept, called a *topologically safe site*, where adding such a site to a GVD does not change the topology beyond the Voronoi cell containing the site. In the following, we introduce the topological representation of GVD and then prove the existence of topologically safe sites.

3.2 Topological Representation

Before defining a topologically safe site, we introduce the *topological* representation of Voronoi cells. Note that a Voronoi cell in

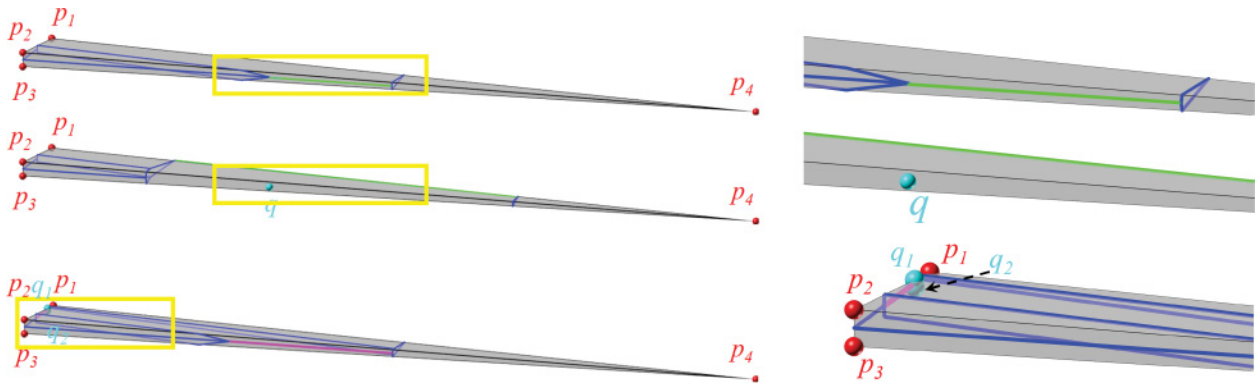


Fig. 4. Illustration of our strategy to ensure the disk condition. Row 1: Voronoi cell $C(p_1)$ violates the closed ball property due to two boundaries and one pseudobisector (green, to be defined in Section 4.3). Row 2: Although adding an auxiliary site q on the pseudobisector makes $C(p_1)$ a topological disk, the new cell $C(q)$ is not simply connected, and it also contains one pseudobisector. Row 3: Adding two topologically safe sites q_1 and q_2 (see Section 4.3 for details), our method fixes $C(p_1)$ and guarantees that the disk condition holds for $C(q_1)$ and $C(q_2)$. Moreover, the topologies of $C(p_2)$, $C(p_3)$, and $C(p_4)$ remain unchanged.

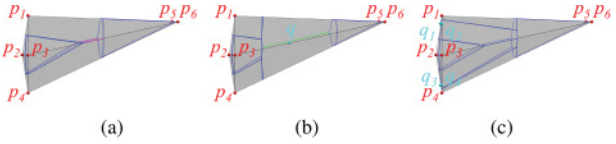


Fig. 5. Illustration of our strategy to ensure the 2-cell intersection condition. (a) All cells are topologically equivalent to a disk. However, $C(p_1)$ and $C(p_4)$ share two common Voronoi edges (pink), violating the 2-cell intersection condition. (b) Although adding an auxiliary site q (cyan) at one of the shared edges can separate $C(p_1)$ and $C(p_4)$, the new Voronoi cell $C(q)$ is not simply connected and contains a pseudobisector (green), violating the disk condition. (c) For each common Voronoi edge, our method adds two auxiliary sites in the topologically safe region (see Section 4.4 for details). As a result, it not only separates the problematic cells $C(p_1)$ and $C(p_4)$ but also guarantees that all new cells $C(q_i)$, $1 \le i \le 4$, are topological disks and satisfy the 2-cell intersection condition.

the GVD can be topologically nontrivial due to multiple boundaries. As Figure 8(b) later shows, there are three boundaries in the Voronoi cell of the "crotch" vertex in the Pant model. For a site s, we represent the Voronoi cell $C(s)$ using its disjoined boundaries (i.e., $C(s) = \{l_1, l_2, \ldots, l_h\}$). Each boundary is a loop $l_i = (b_{i_1}, b_{i_2}, \ldots, b_{i_l})$, which is an *ordered* list of Voronoi edges so that the interior of $C(s)$ is always on the left side when walking from b_{i_j} to $b_{i_{j+1}}$ ($b_{i_{l+1}} = b_{i_1}$). Figure 6 provides an illustration. Since each Voronoi edge is a trimmed bisector of two sites, we represent the Voronoi edge $b(p, q)$ by an ordered pair[2] of the indices of sites p and q. A Voronoi cell is called *simple* if it has only one boundary (i.e., its cardinality $|C(s)| = 1$); otherwise, it is called *nonsimple*.

Given a set of sites $P = \{p_i | p_i \in M\}_{i=1}^m$, we represent the GVD of P by a 2-tuple, $GVD(P) = (P, \Gamma(B_P, C_P))$, where the topological representation $\Gamma(B_P, C_P)$ consists of the sets of symbolic Voronoi edges B_P and Voronoi cells C_P. With the topological representation of a GVD, its geometry is readily available from the positions of sites P, using the algorithm `genus_r_Voronoi_diagram`(M, P) in Liu et al. [2011].

[2]Since there may be more than one shared Voronoi edge between two sites, we allow multiple occurrences of an ordered pair.

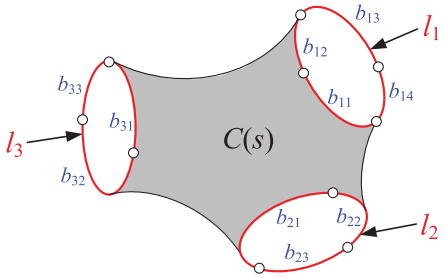


Fig. 6. Topological representation of a Voronoi cell $C(s)$ with three disjoined boundaries, $C(s) = \{l_1, l_2, l_3\}$. Each boundary is an ordered list of Voronoi edges, $l_1 = (b_{11}, b_{12}, b_{13}, b_{14})$, $l_2 = (b_{21}, b_{22}, b_{23})$, and $l_3 = (b_{31}, b_{32}, b_{33})$.

Throughout the article, we use the $\tilde{}$ symbol to distinguish an entity in its *topological* representation and *geometric* realization. For example, the topological representation $b(p, q)$ is an ordered pair of the indices of p and q, whereas the geometric representation $\widetilde{b}(p, q)$ is a polyline on the mesh. Similarly, $\widetilde{l}_i$ and $\widetilde{C}(s)$ are geometric realizations of boundary l_i and Voronoi cell $C(s)$, respectively.

3.3 Topologically Safe Sites

Definition 7 (*Topologically Safe Site*). Let P be a set of sites on M including all mesh vertices, and let $GVD(P)$ the corresponding geodesic Voronoi diagram. For a subset $Q \subseteq P$, let $B_Q(P) \triangleq \{b(r, s) : b(r, s) \in B_P \text{ and } r, s \in Q\}$ be the set of Voronoi edges that are elements of B_P and are generated by sites in Q. Consider a new site $q \notin P$, which is inside a Voronoi cell, say $q \in C(p)$. We say that q is topologically safe to $GVD(P)$ if $B_{P\setminus\{p\}}(P \cup \{q\}) = B_{P\setminus\{p\}}(P)$.

Intuitively speaking, the new site q is topologically safe to $GVD(P)$ if adding q to the Voronoi diagram does not destroy any existing Voronoi edges (in terms of topological representation) other than the edges of the Voronoi cell $C(p)$ that contains q (i.e., $q \in C(p)$). Figure 7 presents examples of topologically safe and unsafe sites.

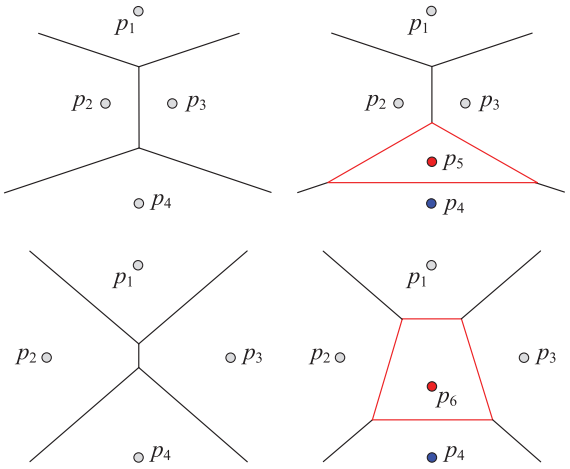


Fig. 7. Topologically safe and unsafe sites. Given four sites $P = \{p_1, p_2, p_3, p_4\}$ in two different positions, the corresponding GVDs have the same topological representation (i.e., $B_P = \{(1,2),(1,3),(2,3),(2,4),(3,4)\}$). Row 1: The site p_5 is topologically safe to $GVD(P)$, since $B_{P\setminus\{p_4\}}(P \cup \{p_5\}) = B_{P\setminus\{p_4\}}(P) = \{(1,2),(1,3),(2,3)\}$; in other words, adding p_5 does not destroy any existing Voronoi edge $b(p_i, p_j)$ for $i \neq 4$, $j \neq 4$. Row 2: The site p_6 is not topologically safe to $GVD(P)$, since $B_{P\setminus\{p_4\}}(P\cup\{p_6\}) = \{(1,2),(1,3)\}$ and $B_{P\setminus\{p_4\}}(P) = \{(1,2),(1,3),(2,3)\}$. Visually, adding p_6 to the Voronoi diagram destroys an existing Voronoi edge $b(p_2, p_3)$, which does not belong to $C(p_4)$. The new Voronoi edges are colored in red, and the existing Voronoi edges are colored in black.

To quantitatively characterize the topologically safe region and find topologically safe sites, we define the δ-offset for a Voronoi cell.

Definition 8 (δ-*Offset*). Consider a geometric realization of Voronoi cell $\widetilde{C}(p)$ with boundaries $\widetilde{C}(p) = (\widetilde{l}_1, \widetilde{l}_2, \ldots, \widetilde{l}_h)$. Let $\Theta(\widetilde{C}(p), \delta) = \{x : d(x,y) < \delta, y \in \widetilde{b}_i, \widetilde{b}_i \subset \widetilde{l}_1 \cup \widetilde{l}_2 \cup \cdots \cup \widetilde{l}_h\}$ be the offset of the boundary of $\widetilde{C}(p)$. $\delta_{\max}(\widetilde{C}(p))$ is the maximal value of δ so that $\Theta(\widetilde{C}(p), \delta)$ does not contain any site in P and any Voronoi vertex that is not in $\widetilde{C}(p)$. For a $GVD(P)$, define

$$\delta_{\max}(GVD(P)) \triangleq \min_{p\in P} \delta_{\max}(\widetilde{C}(p)), \tag{1}$$

the minimal $\delta_{\max}$ for all Voronoi cells.

The following proposition shows that each Voronoi cell has an ε-neighborhood, in which any point is topologically safe to the GVD. This allows us to add auxiliary sites to a Voronoi cell without changing the topology of its neighbors.

Proposition 1 (Existence of ε-Neighborhood). *For any $0 < \delta \leq \delta_{\max}(GVD(P))$, there always exists an $\varepsilon > 0$ such that in each Voronoi cell $C(p)$, $D(p,\varepsilon) = \{x : x \in M$ and $d(p,x) \leq \varepsilon\}$ is simply connected and any point x in $D(p,\varepsilon)$ is topologically safe to $GVD(P)$. $D(p,\varepsilon)$ is referred to as the ε-neighborhood of p.*

The value of ε depends on δ. See Equation (S1) in the supplementary material. Figure 8(c) shows an example of the δ-offset and ε-neighborhood on the Pant model.

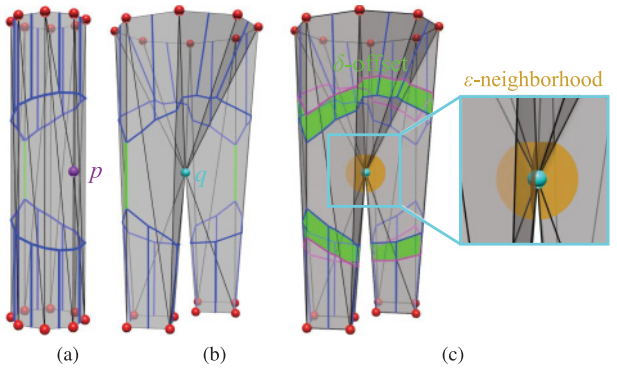


Fig. 8. Pseudobisectors occur in cylinder-shaped regions. There is one pseudobisector in $C(p)$ (a) two in $C(q)$ (b). The bisectors and pseudobisectors are colored in blue and green, respectively. (c) Outward δ-offset (green) of a Voronoi cell and its corresponding ε-neighborhood (orange).

ALGORITHM 1: Constructing Proper IDT from the Dual of GVD

Input: $M = (V, E, F)$, a 2-manifold triangle mesh
Output: $IDT(M)$, a proper IDT defined on M, whose vertex set contains all vertices in V

1: $GVD(V) =$ compute_gvd(M) (Procedure 2 in Section 4.2)
2: $GVD(P) =$ ensure_disk_condition($GVD(V)$), where $V \subseteq P$ (Procedure 3 in Section 4.3)
3: $GVD(P') =$ ensure_2-cell_intersection_condition($GVD(P)$), where $P \subseteq P'$ (Procedure 4 in Section 4.4)
4: $IDT(M) =$ compute_dual_graph($GVD(P')$) (Procedure 5 in Section 4.5)

4. ALGORITHM

4.1 Overview

To facilitate the presentation, we assume that the input 2-manifold mesh M is closed. In the supplementary material, we extend our algorithm to handle meshes with boundaries. Our algorithm (Algorithm 1) consists of four steps. First, taking all mesh vertices as sites, it computes the geodesic Voronoi diagram $GVD(V)$. Then, it checks the disk condition for all Voronoi cells. If a Voronoi cell, say $C(v_i)$, is not homeomorphic to a disk, the algorithm adds auxiliary sites in the ε-neighborhood of v_i to fix $C(v_i)$. Next, it checks the 2-cell intersection condition for all pairs of adjacent Voronoi cells. If two cells, say $C(v_i)$ and $C(v_j)$, share two or more Voronoi edges, the algorithm again adds auxiliary sites to reduce the number of common edges to one. Finally, since the updated GVD satisfies the closed ball property, it computes the dual graph, which is a proper IDT. Intuitively speaking, our method adaptively increases the sampling density for the regions where the closed ball property fails. Figure 9 illustrates the algorithmic pipeline using a toy cylinder model.

4.2 Computing GVD

Liu et al. [2011] presented a generic algorithm to construct GVD on manifold triangle meshes. For $m(\leq n)$ arbitrary sites placed on an n-vertex mesh M, the algorithm runs in $O(n^2 \log n)$ time. One, of course, can apply Liu et al.'s algorithm directly to our application. However, our scenario is slightly different in that we

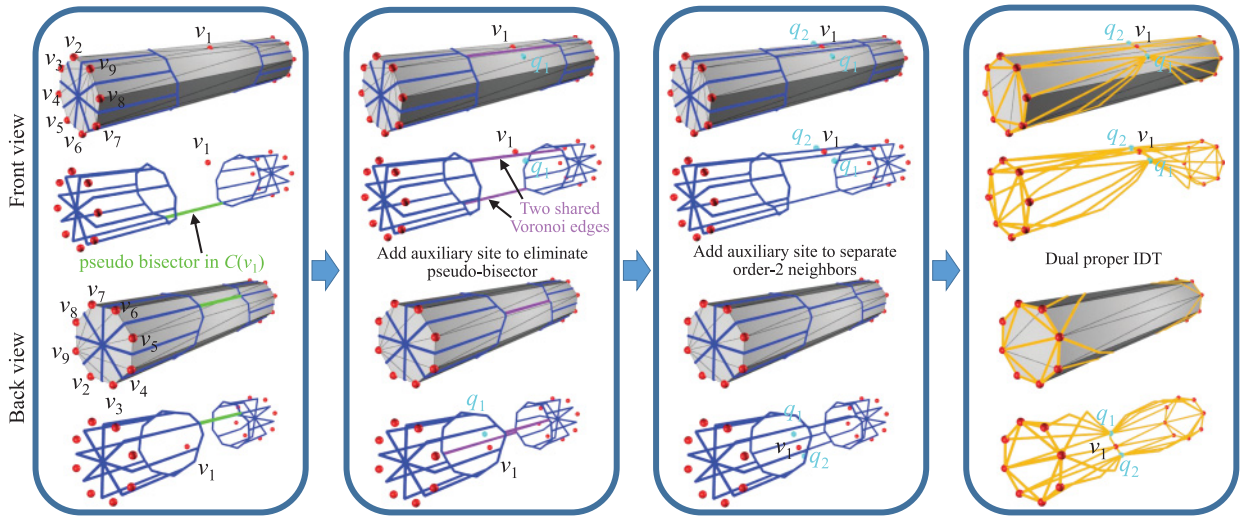


Fig. 9. Algorithmic pipeline. (a) The cylinder mesh has 17 vertices (red). Taking all mesh vertices as sites, we compute the geodesic Voronoi diagram $GVD(V)$. However, $GVD(V)$ does not have a dual proper Delaunay triangulation, as Voronoi cell $C(v_1)$ is not simply connected and it also contains a pseudobisector (green). (b) After adding an auxiliary site q_1 (cyan) in the ε-neighborhood of site v_1, both $C(v_1)$ and $C(q_1)$ become topological disks. However, they share two common Voronoi edges (purple). (c) To ensure the 2-cell intersection condition, we add another auxiliary site q_2 in the ε-neighborhood of v_1. (d) Now the geodesic Voronoi diagram $GVD(P')$ satisfies the closed ball property, and thus its dual graph is a proper IDT.

take all mesh vertices as sites. For better performance, we adapt Liu et al's algorithm as follows.

We assign all mesh vertices a distance value 0, as they are the source points. Then we apply the MMP algorithm to compute the geodesic distances for the points on mesh edges. After termination, the MMP algorithm partitions each mesh edge into disjoint intervals, called *windows*, where each window encodes the shortest paths coming from the same face sequence. Note that the original MMP algorithm must solve a quadratic equation to trim overlapped windows due to the existence of hyperbolic bisectors. Fortunately, all mesh vertices are the sources in our case, and hence we solve only a linear equation for window trimming.

We are interested in the mesh edges that contain windows corresponding to different sources, as they are crossed by some Voronoi edges. Observe that a Voronoi vertex corresponds to three Voronoi edges. Therefore, we can locate Voronoi vertices by finding those triangles where three or more Voronoi edges meet. In lines 6 through 16 of Procedure 2, we compute the Voronoi edges with distinct endpoints. Then in lines 17 through 22, we compute the Voronoi edges that are self-loops.

For simplicity, the pseudocode in Procedure 2 does not handle the case that a triangular face contains two or more Voronoi vertices. We refer readers to Algorithm 1[3] in Xu et al. [2014] for details.

4.3 Ensuring the Disk Condition

In $\mathbb{R}^2$, the term *bisector* refers to the set of points that are equidistant to two distinct sites. In contrast to the 2D counterpart, there are two types of bisectors on polyhedral surfaces: one is the conventional bisector of two distinct sites, and the other corresponds to a single site, hereby called the *pseudo*bisector.

[3]Its implementation details are available at http://cg.cs.tsinghua.edu.cn/people/~Yongjin/PGVD-pg2014-supplement.pdf.

Definition 9 (*Pseudobisector*). The pseudobisector of a site p consists of points $q \in \widetilde{C}(p)$ such that there are two geodesic paths realizing the minimum distance between p and q.

Pseudobisectors occur in cylinder-shaped regions (see Figure 8(a) and (b)). Although a Voronoi cell may have multiple pseudobisectors, the total number of pseudobisectors in $GVD(V)$ is bounded by the number of vertices in a mesh.

Proposition 2 (Complexity of Pseudobisectors). *The number of pseudobisectors in $GVD(V)$ is $O(n)$, where $n = |V|$ is the number of vertices in M.*

Observe that pseudobisectors are due to low sampling density. For a Voronoi cell $C(v_i)$, $v_i \in V$ with pseudobisectors, we add auxiliary sites in the ε-neighborhood of v_i to destroy all all pseudobisectors in $C(v_i)$. Since those auxiliary sites are topologically safe to $GVD(V)$, the topologies of the Voronoi cells adjacent to $C(v_i)$ remain unchanged. As a result, we only need to ensure that the Voronoi cells of these auxiliary sites are topological disks.

We first compute the $\delta_{\max}$-offset for $GVD(P)$ using Equation (1) such that the corresponding $\varepsilon(\delta_{\max})$-neighborhood contains topologically safe sites. Then we compute a *strong* δ-offset $\hat{\delta}(GVD(P)) \leq \delta_{\max}(GVD(P))$ using Equation (S6) in the supplementary material such that pseudobisectors cannot appear in the Voronoi cells of newly added topologically safe sites in the $\varepsilon(\hat{\delta})$-neighborhood, which are sampled in the following way.

Figure 10 illustrates our strategy for eliminating pseudobisectors. The Voronoi cell $C(p)$ has multiple boundaries and has two pseudobisectors. For each pseudobisector $\widetilde{pb}_i(p)$, let $x_i \in \widetilde{pb}_i$ be the point that minimizes the geodesic distance from p to $\widetilde{pb}_i$. Since $x_i \in \widetilde{pb}_i$, there are two minimal geodesics $\gamma(p, x_i)$ and $\gamma'(p, x_i)$ from p to x_i in $\widetilde{C}(p)$. To eliminate a pseudobisector $pb_i(p)$, we add two auxiliary sites $a_i \in \gamma(p, x_i)$ and $b_i \in \gamma'(p, x_i)$ such that $d(p, a_i) = d(p, b_i) = \varepsilon(\hat{\delta})$.

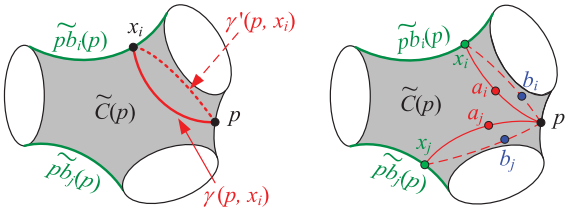


Fig. 10. Voronoi cell $\widetilde{C}(p)$ has two pseudobisectors. To destroy the pseudobisector $pb_i(p)$, we add two auxiliary sites a_i and b_i in the ε-neighborhood of p. See the main text for details.

Procedure 2: Compute_gvd(M)

Taking all mesh vertices V as sources, apply the MMP algorithm to compute geodesic distances.
Put the triangles containing windows from exactly two different sources into a list BT.
Put the triangles containing windows from three or more different sources into a list TT.
Create a Voronoi vertex list Ω from TT, in which each Voronoi vertex ω_i with sources $(v_{i_1}, v_{i_2}, v_{i_3})$ corresponds to a triangle f_i in TT.
Initialize a Voronoi edge list $B_V = \emptyset$.
// Compute Voronoi edges ended at distinct Voronoi vertices
for every $w_i \in \Omega$ **do**
  Get the corresponding f_i from Ω
  for $m = 1$ to 3 **do**
    if the Voronoi edge $b(v_{i_m}, v_{i_{(m+1)\%3}}) \notin B_V$ **then**
      Starting from face f_i, compute the Voronoi edge $\widetilde{b}(v_{i_m}, v_{i_{(m+1)\%3}})$ until reaching another face f_j containing a Voronoi vertex.
      Remove those triangles crossed by $\widetilde{b}(v_{i_m}, v_{i_{(m+1)\%3}})$ from BT.
      Add $b(v_{i_m}, v_{i_{(m+1)\%3}})$ into B_V.
    end if
  end for
end for
// Compute Voronoi edges that are self-loops
while BT is not empty **do**
  Take the first face f from BT. // f is crossed by only one Voronoi edge.
  Starting from f, compute the Voronoi edge $\widetilde{b}'$ until getting back to f.
  Remove those triangles crossed by $\widetilde{b}'$ from BT.
end while
Extract the Voronoi cells C_V from the set of Voronoi edges B_V.
return (B_V, C_V), the topological representation of $GVD(V)$.

Observe that for any point $y \in \widetilde{pb}_i(p)$, the geodesic distance $d(y, p) > \max\{d(y, a_i), d(y, b_i)\}$. Therefore, adding a_i and b_i to the Voronoi diagram destroys the pseudobisector $\widetilde{pb}_i(p)$. Moreover, both a_i and b_i are topologically safe sites, as they are within the ε-neighborhood of site p.

In fact, adding one auxiliary site a_i is sufficient to destroy the pseudobisector $\widetilde{pb}_i$. We need the other auxiliary site b_i to ensure that there are no new pseudobisectors in $\widetilde{C}(a_i)$ and $\widetilde{C}(b_i)$. In general, for a Voronoi cell $C(p)$ with k pseudobisectors, the following proposition shows that adding $2k$ auxiliary sites are sufficient to eliminate all existing pseudobisectors. At the same time, no new pseudobisectors will be introduced.

Procedure 3: Ensure_Disk_Condition($GVD(V)$)

$P = V$
Compute $\varepsilon(\hat{\delta}(GVD(V)))$ using Equations (S1), (S6), and (S7) in the supplementary material.
for every $v \in V$ **do**
  if $C(v)$ has two or more boundaries **then**
    Initialize sets of auxiliary sites $A = \emptyset$ and $B = \emptyset$.
    for each pseudobisector $\widetilde{pb}_i \subset \widetilde{C}(v)$ **do**
      Find $x_i \in \widetilde{pb}_i$ that minimizes the distance from v to $\widetilde{pb}_i$.
      Compute two minimal geodesics $\gamma(v, x_i)$ and $\gamma'(v, x_i)$.
      Find $a_i \in \gamma$ and $b_i \in \gamma'$ such that $d(v, a_i) = d(v, b_i) = \varepsilon(\hat{\delta})$.
      $A = A \cup \{a_i\}$ and $B = B \cup \{b_i\}$.
    end for
    $P = V \bigcup A$.
    Locally update $GVD(P)$, $\hat{\delta}(GVD(P))$ and $\varepsilon(\hat{\delta})$.
    for each auxiliary site $a_i \in A$ **do**
      if $C(a_i)$ has two or more disjoint boundaries **then**
        $P = P \cup \{b_i\}$.
        Locally update $GVD(P)$, $\hat{\delta}(GVD(P))$ and $\varepsilon(\hat{\delta})$.
      end if
    end for
  end if
end for
return $GVD(P)$, in which all Voronoi cells are topological disks.

Proposition 3 (Eliminating Pseudobisectors). *Let $\widetilde{C}(p)$ be a Voronoi cell with k pseudobisectors in $GVD(P)$, $V \subseteq P$. After adding $2k$ auxiliary sites $\{a_i, b_i\}_{i=1}^k$ to $\widetilde{C}(p)$, the updated cells $\widetilde{C}(p)$ and $\{\widetilde{C}(a_i), \widetilde{C}(b_i)\}_{i=1}^k$ are all topological disks. The topology of all other cells remains unchanged.*

Theoretically, $2k$ auxiliary sites are required to eliminate all pseudobisectors in $C(p)$. In practice, we adopt an incremental strategy in Procedure 3. We add the auxiliary site a_i first and then update the topological representations of $C(p)$ and $C(a_i)$. We add the other site b_i only when $C(a_i)$ has two or more boundaries.

Since there are $O(n)$ pseudobisectors in a mesh, Procedure 3 adds $O(n)$ auxiliary sites so that all Voronoi cells in $GVD(P)$, $V \subseteq P$, are topological disks.

4.4 Ensuring the 2-Cell Intersection Condition

After eliminating pseudobisectors, all Voronoi cells in $GVD(P)$ are topological disks. Now we check the 2-cell intersection condition for all pairs of adjacent Voronoi cells. Let B_P be the set of symbolic Voronoi edges in $GVD(P)$. Recall that in topological representation, the Voronoi edge $b(p_i, p_j)$ is an ordered pair $(\min\{I(p_i), I(p_j)\}, \max\{I(p_i), I(p_j)\})$, where $I(p)$ is the index of site p. We sort the Voronoi edges in B_P in ascending order by their first elements. If two ordered pairs have the same first element, the second element is used to break a tie. If a Voronoi edge $b(p_i, p_j)$ appears more than once in B_P, the corresponding Voronoi cells $C(p_i)$ and $C(p_j)$ violate the 2-cell intersection condition. Figure 11 provides an example.

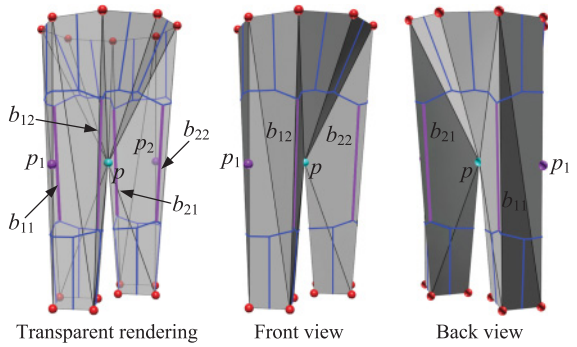


Fig. 11. Since each Voronoi cell $C(p_i)$, $i = 1, 2$, has two shared Voronoi edges b_{i1} and b_{i2} with $C(p)$, p_i is a multiple-order neighbor of p.

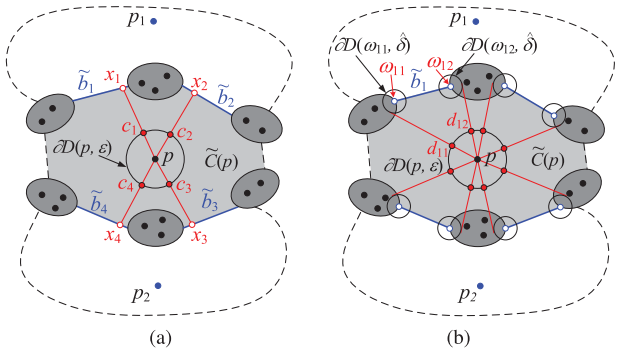


Fig. 12. Both p_1 and p_2 are multiple-order neighbors of p. For each Voronoi edge $\widetilde{b}_i$ shared by p_1 (or p_2) and p, $i = 1, 2, 3, 4$, we add three auxiliary sites $\{c_i, d_{i1}, d_{i2}\}_{i=1}^{k}$ in the $\varepsilon(\hat{\delta})$-neighborhood of p. Then the updated Voronoi cells $\widetilde{C}(p)$, $\{\widetilde{C}(c_i), \widetilde{C}(d_{i1}), \widetilde{C}(d_{i2})\}_{i=1}^{2}$ are all topological disks and satisfy the 2-cell intersection condition. See the main text for details.

Definition 10 (*Neighboring Site*). p and q are neighboring sites if $C(p)$ and $C(q)$ share at least one Voronoi edge. If there is exactly one shared Voronoi edge, we say q (respectively p) is a single-order neighbor to p (respectively q). Otherwise, q (respectively p) is referred to as a multiple-order neighbor of p (respectively q).

Similar to pseudobisectors, multiple-order neighbors are also due to low sampling density. To separate two multiple-order neighbors, we add auxiliary sites in between them. We adopt the following two-step strategy to ensure that the Voronoi cell of each newly added auxiliary site satisfies the closed ball property, where referring to Figure 12, both p_1 and p_2 are multiple-order neighbors to p:

—Step 1. For each Voronoi edge $\widetilde{b}_i$ shared by p_1 (or p_2) and p, $i = 1, 2, 3, 4$, we find a point $x_i \in \widetilde{b}_i$ that minimizes the distance from p to $\widetilde{b}_i$. Then we compute $c_i \in \gamma(p, x_i)$ such that $d(p, c_i) = \varepsilon(\hat{\delta})$.

—Step 2. Each Voronoi edge $\widetilde{b}_i$ has two Voronoi vertices ω_{i1} and ω_{i2}. At each ω_{ij}, $j = 1, 2$, we place a geodesic circle $\partial D(\omega_{ij}, \hat{\delta})$. There is a geodesic to p that is tangential to $\partial D(\omega_{ij}, \hat{\delta})$ and not intersecting $\widetilde{b}_i$. Denote this geodesic as $\gamma(\omega_{ij})$. Let d_{ij} be the point on $\gamma(\omega_{ij})$ satisfying $d(p, d_{ij}) = \varepsilon(\hat{\delta})$.

In summary, in the Voronoi cell $\widetilde{C}(p)$, for each Voronoi edge shared by p and a multiple-order neighbor p_1 (or p_2), we add three auxiliary sites c_i, d_{i1}, and d_{i2} to separate p_1 (or p_2) and p. In general, if Voronoi cell $C(p)$ shares k Voronoi edges with multiple-order neighbors, we add $3k$ auxiliary sites to separate them.

Proposition 4 (Separating Multiple-Order Neighbors). *Let $GVD(P)$, $V \subseteq P$, be a GVD in which each Voronoi cell is a topological disk. Let $\widetilde{C}(p)$ be a Voronoi cell that shares k Voronoi edges with multiple-order neighbors. Adding $3k$ auxiliary sites $\{c_i, d_{i1}, d_{i2}\}_{i=1}^{k}$ in the $\varepsilon(\hat{\delta})$-neighborhood of p is topologically safe to $GVD(P)$, and is sufficient to make the updated Voronoi cells $\widetilde{C}(p)$, $\{\widetilde{C}(c_i), \widetilde{C}(d_{i1}), \widetilde{C}(d_{i2})\}_{i=1}^{k}$ all topological disks and satisfying the 2-cell intersection condition.*

Procedure 4: Ensure_2-Cell_Intersection_Condition($GVD(P)$)

Let $P' = P$.
Sort all Voronoi edges in ascending order.
for any two Voronoi edges $b_k = b_l = \{p_i, p_j\}$ with the same ordered pair **do**
  Place p_j into the multiple-order neighbor list of p_i
  Place p_i into the multiple-order neighbor list of p_j.
end for
for each Voronoi cell $\widetilde{C}(p)$ with multiple-order neighbor(s) **do**
  for each shared Voronoi edge $\widetilde{b}_i$ with a multiple-order neighbor **do**
    Find $x_i \in \widetilde{b}_i$ that minimizes the distance from p to $\widetilde{b}_i$.
    Compute the geodesic $\gamma(p, x_i)$.
    Find $c_i \in \gamma(p, x_i)$ such that $d(p, c_i) = \varepsilon(\hat{\delta})$.
    $P' = P' \cup \{c_i\}$.
  end for
  Locally update $GVD(P')$, $\hat{\delta}$ and $\varepsilon(\hat{\delta})$.
  Remove p from the multiple-order neighbor list of each site in p's multiple-order neighbor list.
  while there is a new Voronoi cell $\widetilde{C}(c_i)$ whose multiple-order neighbor list is not empty **do**
    Find two Voronoi vertices ω_{i_1} and ω_{i_2} of the Voronoi edge $\widetilde{b}_i$.
    Compute the geodesics γ_{i_1} and γ_{i_2} to p tangent to $\partial D(\omega_{i_1}, \hat{\delta})$ and $\partial D(\omega_{i_2}, \hat{\delta})$, respectively, that do not intersect $\widetilde{b}_i$.
    Find $d_{i_1} \in \gamma_{i_1}$ and $d_{i_2} \in \gamma_{i_2}$ such that $d(p, d_{i_1}) = d(p, d_{i_2}) = \varepsilon(\hat{\delta})$.
    $P' = P' \cup \{d_{i_1}, d_{i_2}\}$.
    Locally update $GVD(P')$, $\hat{\delta}$ and $\varepsilon(\hat{\delta})$.
  end while
end for
return $GVD(P')$, which satisfies the closed ball property.

As shown in the supplementary material, adding k auxiliary sites $\{c_i\}_{i=1}^{k}$ is sufficient to remove all k multiple-order neighbors from $\widetilde{C}(p)$. Moreover, we need an additional $2k$ auxiliary sites $\{d_{i1}, d_{i2}\}_{i=1}^{k}$ to ensure that the Voronoi cells $\{\widetilde{C}(c_i), \widetilde{C}(d_{i1}), \widetilde{C}(d_{i2})\}_{i=1}^{k}$ are free of new pseudobisectors and multiple-order neighbors. In Procedure 4, we use an incremental sampling scheme that adds k auxiliary sites $\{c_i\}_{i=1}^{k}$ first. We add the additional sites d_{i1} and d_{i2} only if a Voronoi cell $\widetilde{C}(c_i)$ has new multiple-order neighbors.

4.5 Computing the Dual Graph

Since the closed ball property holds everywhere for $GVD(P')$, its dual Delaunay triangulation is proper. Furthermore, each Voronoi

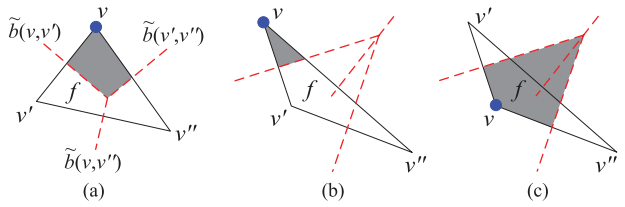


Fig. 13. Consider the window w associated to the edge opposite to vertex v. (a, b) When the angle of f at v is *acute*, the window can only stay inside f—that is, it cannot cross f. (c) When the angle of f at v is *obtuse*, the window propagates across f. The red dashed lines are the bisectors of two vertices. The region swept by window w is shaded.

edge has a unique dual Delaunay edge by the empty geodesic circumcircle property [Dyer et al. 2007b]. Given a Voronoi edge $\widetilde{b}(p,q)$ in $GVD(P')$, we denote by $\mathcal{F}_p$ (respectively $\mathcal{F}_q$) the set of faces containing the geodesic paths from p (respectively q) to any point on $b(p,q)$. Then we compute the unique Delaunay edge $\gamma(p,q)$ by unfolding the faces $\mathcal{F}_p \cup \mathcal{F}_q$ to $\mathbb{R}^2$ (Procedure 5).

Procedure 5: Compute_Dual_Graph($GVD(P')$)

1: **for** each Voronoi edge $\widetilde{b}(p,q)$ in $GVD(P')$ **do**
2: Using the windows stored at the two sides of $\widetilde{b}(p,q)$, compute the face sets $\mathcal{F}_p$ and $\mathcal{F}_q$, respectively.
3: Compute $\gamma(p,q)$ by unfolding the triangles in $\mathcal{F}_p \bigcup \mathcal{F}_q$.
4: **end for**
5: **for** any three Voronoi cells that meet at a Voronoi vertex **do**
6: Create a geodesic triangle with the three corresponding g-edges.
7: **end for**
8: **return** $IDT(M)$, a proper IDT.

5. CORRECTNESS AND COMPLEXITY ANALYSIS

THEOREM 11. *Let M be an arbitrary 2-manifold triangle mesh with n vertices. Algorithm 1 converts M into a proper IDT with $O(n)$ auxiliary sites. The algorithm has a worst-case $O(n^2+tn\log n)$ time complexity, where t is the number of obtuse angles in the mesh.*

PROOF. First, we show that the Procedure 2 compute_gvd takes $O(n^2+tn\log n)$ time. The MMP algorithm [Mitchell et al. 1987] partitions each mesh edge into a set of intervals, called *windows*, which locally encode the geodesic information. For each vertex v, the MMP algorithm initializes a window w for every edge opposite to v. Such a window can be characterized by the angle associated with v that is opposite to the edge containing w. Then the MMP algorithm propagates windows across mesh faces to simulate the wavefront propagation. A window stops propagating if it hits the boundary or another window.

Our analysis is based on two key observations (Figure 13):

—If the angle of f at v is *acute*, the window w stays inside f—that is, it does not propagate beyond f.
—If the angle of f at v is *obtuse*, the window is propagated across f. In the worst case, a window can cross $O(n)$ faces (Figure 14).

Since there are at most $O(n)$ obtuse angles in M, the MMP algorithm produces $O(tn)$ windows and hereby runs in $O(tn\log n)$ time. Figure 14 shows such a worst case with $t = O(n)$, implying that

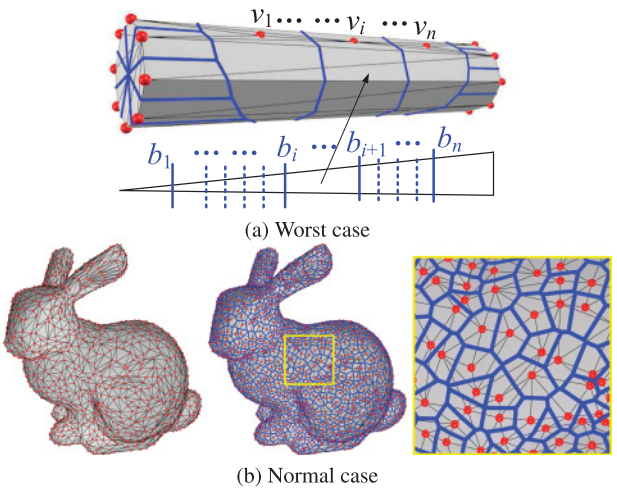


Fig. 14. Complexity of Voronoi edges. (a) Each mesh vertex v_i, $i = 1, 2, \ldots, n$, has two obtuse angles, and each obtuse angle initializes a window that can cross $O(n)$ triangles. Therefore, the triangles on the side of the cylinder are *global*, as each of these triangles contains $O(n)$ Voronoi edges. As a result, computing $GVD(V)$ takes $O(n^2\log n)$ time. (b) In general, if the majority of mesh faces are *local* (i.e., each edge intersects only a few Voronoi edges (see the close-up view)), the GVD can be computed in $O(n)$ time.

the bound $O(tn\log n)$ is tight. For this case, the time complexity becomes $O(n^2\log n)$, which is a well-known result of the original MMP algorithm [Mitchell et al. 1987].

Then we show that Procedures 3, 4, and 5 all take $O(n^2)$ time. Computing $\varepsilon(\hat{\delta})$ takes $O(n^2)$ time. For each pseudobisector $\widetilde{pb}_i$ contained in a Voronoi cell $\widetilde{C}(v)$, finding $x_i \in \widetilde{pb}_i$ that minimizes the distance from v to $\widetilde{pb}_i$ and computing the geodesic paths $\gamma(v,x_i)$ and $\gamma'(v,x_i)$ take $O(n)$ time. Finding a_i on γ and b_i on γ' satisfying $d(v,a_i) = d(v,b_i) = \varepsilon(\hat{\delta})$ takes $O(1)$ time. Locally updating $GVD(P)$ takes $O(n)$ time. By Proposition 2, there are totally $O(n)$ pseudobisectors in $GVD(V)$. Thus, Procedure 3 takes $O(n^2)$ time.

Similarly, Procedure 4 adds at most $O(n)$ auxiliary sites. Since updating Voronoi cell for each new site takes $O(n)$ time, it takes $O(n^2)$ time.

In Procedure 5, computing a Delaunay edge $\gamma(p,q)$ takes $O(n)$ time, as there are at most $O(n)$ faces in the set $\mathcal{F}_p \bigcup \mathcal{F}_q$. Also note that there are $O(n)$ Voronoi vertices and $O(n)$ Voronoi edges in $GVD(P')$, and hereby Procedure 5 takes $O(n^2)$ time.

Putting it all together, our algorithm has a worst-case time complexity $O(n^2+tn\log n)$. □

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Performance

We implemented our algorithm in C++ and tested it on 20 synthetic and real-world models with diverse geometric and topological features. To quantitatively investigate the relation between mesh quality and runtime performance, we adopted the popular *anisotropy* measure [Zhong et al. 2013] $\sigma(t) = \frac{ph}{2\sqrt{3}S}$ for quality of a triangle t, where p is the half-perimeter, h is the length of its longest edge, and S is the triangle area. It is easy to see that $\sigma(t) \geq 1$, and the equality holds when t is equilateral. We measure the triangulation quality by the mean σ_{mean}, maximum σ_{max}, and standard deviation σ_{std} of σ for all triangles of M. Usually, a larger σ means the higher

Table II. Mesh Complexity and Running Time Statistics

Model	Original Mesh		Edge Flipping		IDT			Delaunay Mesh		
	$\|E\|$	Anisotropy σ	Time (sec)	# ef	Time (sec)	# as	Anisotropy σ	Time (sec)	# sp	Anisotropy σ
Pant	38	(3.04, 4.48 0.19)	0.000	14	0.000	4	(2.33, 3.34, 0.16)	0.003	12	(2.69, 4.47, 0.15)
Horse	300	(1.82, 5.30, 0.05)	0.000	58	0.000	2	(1.47, 4.15, 0.03)	0.003	107	(1.97, 18.25, 0.07)
Bunny	300	(1.71, 8.56, 0.07)	0.000	35	0.000	2	(1.47, 2.91, 0.03)	0.002	73	(1.83, 15.48, 0.07)
Kitten	300	(1.66, 16.21, 0.09)	0.000	35	0.000	2	(1.48, 5.26, 0.04)	0.002	58	(2.02, 19.61, 0.10)
CSG	357	(4.74,33.02,2.13)	0.006	120	0.004	2	(1.94, 4.93, 0.06)	0.009	407	(5.41, 64.13, 0.84)
Bear	400	(2.47, 40.57, 2.21)	0.003	88	0.002	2	(1.50, 3.30, 0.25)	0.004	217	(4.00, 64.80, 1.02)
Fandisk	1,081	(44.22, 177.54, 9.49)	0.015	382	0.011	0	(3.21, 0.34, 0.11)	0.125	9,530	(5.82, 92.49, 7.17)
Headst	1,714	(90.00, 330.64, 47.52)	0.060	1,693	0.001	0	(3.50, 21.03, 2.32)	0.905	93,730	(2.96, 379.12, 0.45)
Eight	3,033	(1.67, 11.80, 0.88)	0.031	717	0.016	0	(1.45, 4.81, 0.01)	0.102	8,688	(2.02, 30.59, 0.02)
Sphere	5,994	(1.67, 59.60, 2.14)	0.019	5	0.006	0	(1.67, 59.60, 2.13)	0.036	138	(1.70, 59.60, 2.12)
Lamp	12,780	(78.02, 754.60, 14.70)	0.095	3,223	0.003	0	(19.73, 5.33, 0.61)	16.679	814,943	(15.88, 102.15, 6.33)
Teapot	17,493	(1.69, 20.74, 1.66)	0.171	4,136	0.125	0	(1.03, 7.32, 0.40)	0.564	75,828	(1.95, 57.42, 2.48)
Decocube	24,000	(1.56, 13.43, 0.65)	0.188	4,227	0.140	0	(1.42, 4.02, 0.22)	0.311	45,967	(1.94, 12.23, 1.25)
Sword	29,568	(11.12, 1353.23, 3.42)	0.298	11,345	0.060	0	(9.50, 1309.48, 3.13)	1.923	224,240	(6.48, 7632.53, 2.60)
Fertility	36,920	(2.34, 17.20, 1.11)	0.609	17,127	0.561	0	(1.82, 2.76, 0.10)	1.044	112,437	(3.50, 22.37, 0.13)
Crank	60,000	(17.52,279.54,14.16)	1.539	66,235	1.350	0	(3.60, 3.02, 0.04)	2.210	413,768	(7.63, 12.51, 4.33)
Bunny	216,054	(1.23, 11.83, 0.19)	0.780	3,025	0.156	0	(1.15, 4.39, 0.01)	1.041	73,874	(1.29, 43.48, 0.01)
Armadillo	518,916	(1.31, 95.29, 2.39)	2.324	23,831	0.983	0	(1.30, 83.86, 2.13)	2.520	265,651	(1.52, 76.29, 1.90)
Lucy	788,721	(1.45, 34.24, 1.38)	5.179	97,418	3.960	0	(1.41, 1.22, 0.36)	7.026	924,703	(1.86, 11.17, 0.38)
Buddha	1,195,719	(1.47, 29.16, 5.22)	8.502	149,728	5.379	0	(1.26, 0.79, 1.28)	9.430	964,417	(2.24, 18.13, 1.99)

Note: $|E|$ represents the number of edges in M, # ef represents the number of edge flips by the edge-flipping algorithm, # as represents the number of auxiliary sites added by our algorithm, and # sp represents the number of splitting points added by the Delaunay mesh algorithm. The 3-tuple (σ_{mean}, σ_{max}, σ_{std}) shows the mean, max, and standard deviation of the anisotropy measure σ. The running time was measured in seconds on an Intel Core i7-2600 CPU (3.40GHz).

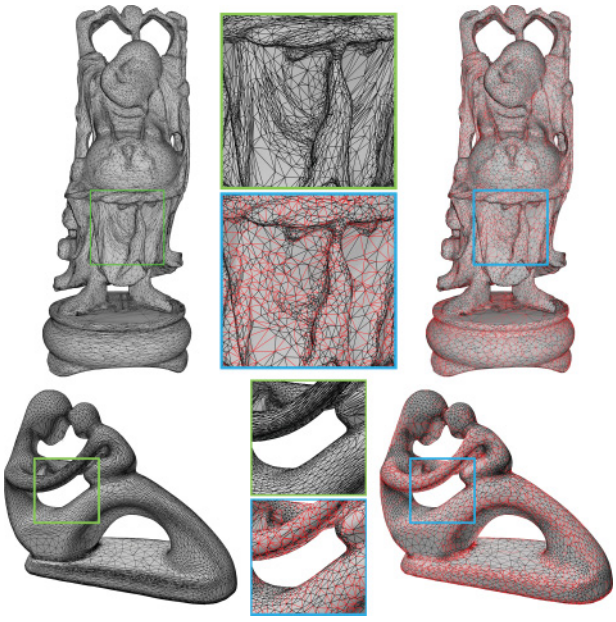


Fig. 15. Results. The original mesh edges and the Delaunay edges are colored in black and red, respectively. The figures are of high resolution, allowing close-up examination.

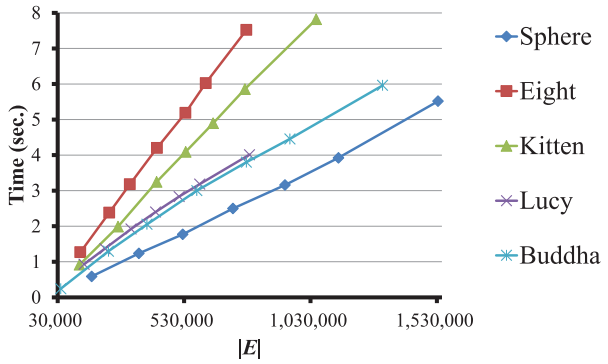


Fig. 16. Our algorithm empirically runs in linear time on real-world meshes. The horizontal axis shows the mesh complexity, and the vertical axis is the execution time (in seconds).

degree of anisotropy and the farther away the mesh is from its IDT. As Table II shows, most meshes are far from their Delaunay triangulations. Take the fertility model as an example (Figure 15). Since 46.3% edges are non-Delaunay edges, it takes the edge-flipping algorithm 17,127 iterations to fix them. Our algorithm, in contrast, computes the IDT in a noniterative manner, and its performance is not sensitive to the number of non-Delaunay edges.

We also observe that the theoretical worst-case time complexity $O(n^2 \log n)$ is very pessimistic, as it happens only when each triangle contains $O(n)$ bisectors. Figure 14(a) presents a model with the worst-case time complexity. We call a triangle with $O(n)$ Voronoi edges *global*, as it indeed spans a global region on the model. We observe that on many real-world models, the majority of the triangles are not global, even though the mesh triangulation is poor (i.e., far from its Delaunay triangulation). Computational results show that on average, a mesh edge on a real-world model has only $O(1)$ Voronoi edges (see Figure 14(b)). As a result, all of the four steps in our algorithm run in $O(n)$ time. Computational results confirm that our algorithm has an empirical linear time complexity (Figure 16),

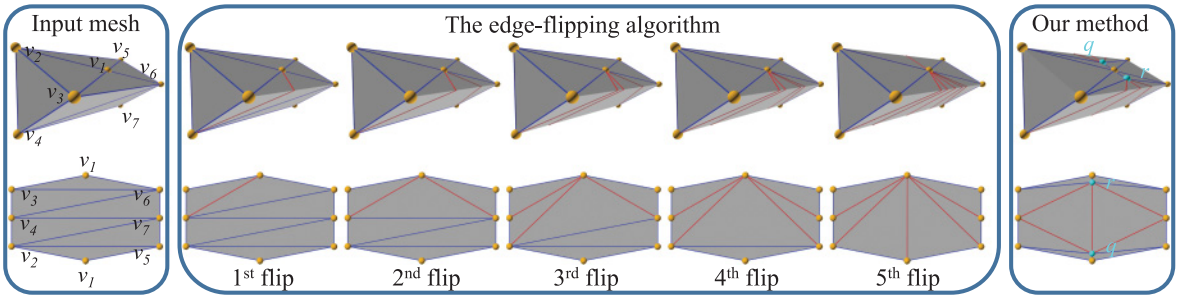


Fig. 17. Illustration of the edge-flipping algorithm. Consider a prism with seven vertices. Among the four edges opposite to v_1, v_2v_3, and v_5v_6 are locally Delaunay and the other two edges v_2v_5 and v_3v_6 are not. The edge-flipping algorithm iteratively flips the non-Delaunay edges until all edges become locally Delaunay. However, the resulting Delaunay triangulation is not proper due to the self-loop (red) connecting v_1 and itself. Our method adds two auxiliary sites q and r in the topologically safe region of v_1 and produce a proper IDT. The bottom row shows the 2D flattening of the region.

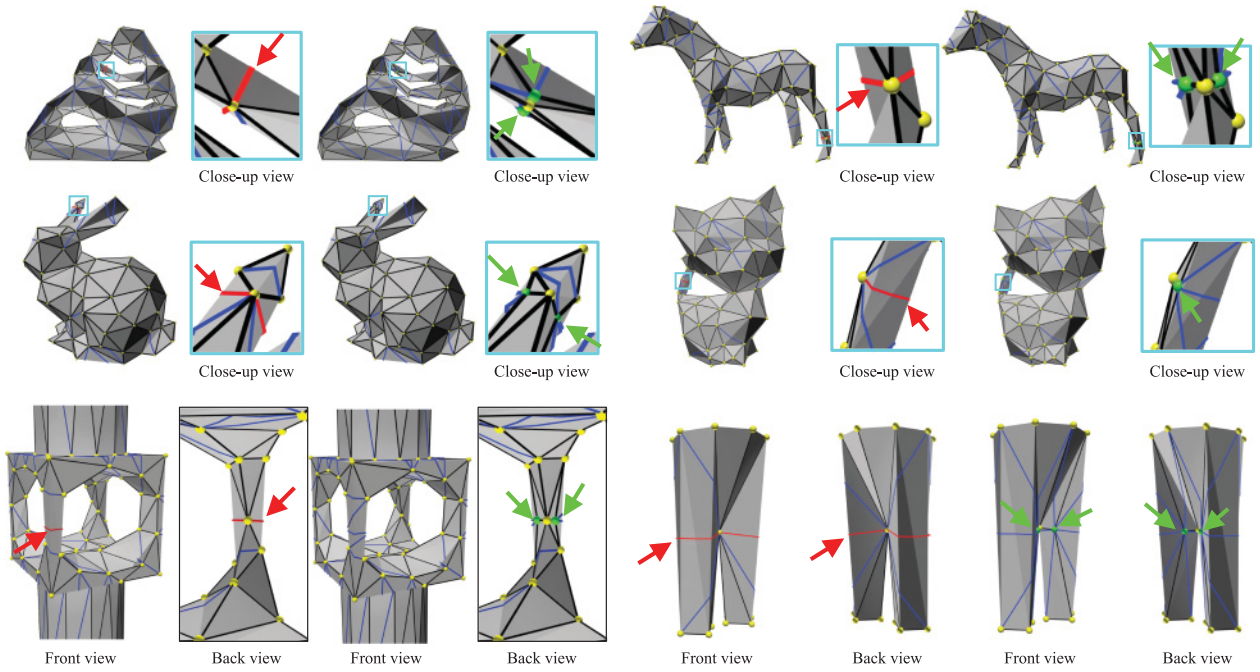


Fig. 18. Comparison with the edge-flipping algorithm. Due to low sampling density on the thin handles, the IDTs produced by the edge-flipping algorithm contain self-loops (red arrows). Our method generates proper IDTs by adding auxiliary sites on the handles (green arrows).

and it consistently outperforms the edge-flipping algorithm in terms of execution time.

6.2 Results

As mentioned previously, the edge-flipping algorithm may produce IDT with self-loops. In contrast, our method guarantees the proper IDT by adding auxiliary sites at the poorly sampled region (Figures 17 and 18). Theoretically, our algorithm adds $O(n)$ auxiliary sites to obtain a proper IDT. In practice, we observe that only a very small number of auxiliary sites are required for real-world models. This is not a surprise, since the auxiliary sites are only added on the sparsely sampled regions that are not homeomorphic to a disk (e.g., the cylinder-like geometry in Figures 9 and 18). As Table II shows, although most test models are far from their Delaunay triangulations, they have a fairly good sampling density. Consequently,

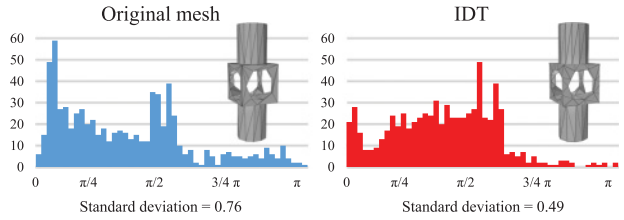


Fig. 19. Angle histograms show that IDT improves triangulation quality.

the resultant $GVD(V)$ automatically satisfies the closed ball density, and our algorithm does not add any new site at all. For these models, both our method and the edge-flipping algorithm produce exactly the same results. Table II and Figure 19 also show that IDTs significantly improve triangulation quality.

ALGORITHM 6: Constructing Proper IDT with Uniform Samples

Input: $M = (V, E, F)$, a 2-manifold triangle mesh

Output: $IDT(M)$, a proper IDT defined on M, whose vertex set contains all vertices in V

1: $GVD(V) =$ compute_gvd(M) (Procedure 2)
2: $GVD(P) =$ ensure_disk_condition_2($GVD(V)$), where $V \subseteq P$ (Procedure 7)
3: $GVD(P') =$ ensure_2-cell_intersection_condition_2($GVD(P)$), where $P \subseteq P'$ (Procedure 8)
4: $IDT(M) =$ compute_dual_graph($GVD(P')$) (Procedure 5)

Procedure 7: ensure_disk_condition_2($GVD(V)$)

1: $P = V$
2: Compute $\varepsilon(\hat{\delta}(GVD(V)))$ using Equations (S1), (S6), and (S7) in the supplementary material.
3: **for** every $v \in V$ **do**
4: **if** $C(v)$ has two or more boundaries **then**
5: **for** each pseudobisector $\widetilde{pb}_i \subset \widetilde{C}(v)$ **do**
6: Find $x_i \in \widetilde{pb}_i$ that minimizes the distance from v to $\widetilde{pb}_i$.
7: Compute the Voronoi cell $\widetilde{C}(x_i)$ in $GVD(V \cup \{x_i\})$.
8: **if** $\widetilde{C}(x_i)$ has only one connected boundary **then**
9: $P = V \bigcup \{x_i\}$.
10: Locally update $GVD(P)$, $\hat{\delta}(GVD(P))$ and $\varepsilon(\hat{\delta})$.
11: **else**
12: Compute two minimal geodesics $\gamma(v, x_i)$ and $\gamma'(v, x_i)$.
13: Find $a_i \in \gamma$ and $b_i \in \gamma'$ such that $d(v, a_i) = d(v, b_i) = \varepsilon(\hat{\delta})$.
14: $P = V \bigcup \{a_i\}$.
15: Locally update $GVD(P)$ and $\varepsilon(\hat{\delta}(GVD(P)))$.
16: **if** $C(a_i)$ has two or more disjoint boundaries **then**
17: $P = P \cup \{b_i\}$.
18: Locally update $GVD(P)$, $\hat{\delta}(GVD(P))$ and $\varepsilon(\hat{\delta})$.
19: **end if**
20: **end if**
21: **end for**
22: **end if**
23: **end for**
24: **return** $GVD(P)$, in which all Voronoi cells are topological disks.

6.3 An Improved Algorithm

Algorithm 1 adds new auxiliary sites near the existing sites to eliminate pseudobisectors and separate multiple-order neighbors. However, in practice, to obtain a fairly regular Voronoi diagram, it is often desired to add new sites away from the existing sites. In this section, we propose two effective strategies for eliminating pseudobisectors and separate multiple-order neighbors. Combining them with Procedures 3 and 4, the improved algorithm (Algorithm 6) can output more uniform samples than those of Algorithm 1 and can also guarantee that the resulting IDTs are proper. The time complexity remains unchanged as stated in Theorem 11.

The first strategy is to add an auxiliary site on each pseudobisector. Let $\widetilde{pb}_i$ be a pseudobisector in a Voronoi cell $\widetilde{C}(v)$, and let $x_i \in \widetilde{pb}_i$ be the position that minimizes the distance from v to $\widetilde{pb}_i$. It is readily seen that adding an auxiliary site at x_i will remove pseudobisector pb_i from $\widetilde{C}(v)$ in the updated $GVD(V \cup \{x_i\})$. However, the new Voronoi cell $\widetilde{C}(x_i)$ cannot be guaranteed to be free of pseudobisectors; see a counterexample in Figure 4. In practice, we find that this strategy works well and that the extremely pathological cases like the one in Figure 4 are rare. Procedure 7 combines this strategy (lines 7 through 10) with Procedure 3. Since locally computing $\widetilde{C}(x_i)$ and examining the existence of any pseudobisector in $\widetilde{C}(x_i)$ take $O(n)$ time, Procedure 7 has the same time complexity as Procedure 3.

Procedure 8: ensure_2-cell_intersection_condition_2($GVD(P)$)

1: Let $P' = P$.
2: Sort all Voronoi edges in ascending order.
3: **for** any two Voronoi edges $b_k = b_l = \{p_i, p_j\}$ with the same ordered pair **do**
4: Place p_j into the multiple-order neighbor list of p_i
5: Place p_i into the multiple-order neighbor list of p_j.
6: **end for**
7: **for** each Voronoi cell $\widetilde{C}(p)$ with a nonempty list of multiple-order neighbors **do**
8: **for** each shared Voronoi edge $\widetilde{b}_i$ with a multiple-order neighbor **do**
9: Find $x_i \in \widetilde{b}_i$ that minimizes the distance from p to $\widetilde{b}_i$.
10: Compute the Voronoi cell $\widetilde{C}(x_i)$ in $GVD(V \cup \{x_i\})$.
11: **if** $\widetilde{C}(x_i)$ has only one connected boundary and has no multiple-order neighbor **then**
12: $P' = P' \bigcup \{x_i\}$.
13: Locally update $GVD(P')$, $\hat{\delta}$ and $\varepsilon(\hat{\delta})$.
14: **else**
15: Compute the geodesic $\gamma(p, x_i)$.
16: Find $c_i \in \gamma(p, x_i)$ such that $d(p, c_i) = \varepsilon(\hat{\delta})$.
17: $P' = P' \cup \{c_i\}$.
18: Locally update $GVD(P')$, $\hat{\delta}$ and $\varepsilon(\hat{\delta})$.
19: **if** $\widetilde{C}(c_i)$ have multiple-order neighbor sites **then**
20: Find two Voronoi vertices ω_{i_1} and ω_{i_2} of the Voronoi edge $\widetilde{b}_i$.
21: Compute the geodesics γ_{i_1} and γ_{i_2} to the p tangent to $\partial D(\omega_{i_1}, \hat{\delta})$ and $\partial D(\omega_{i_2}, \hat{\delta})$, respectively, that do not intersect $\widetilde{b}_i$.
22: Find $d_{i_1} \in \gamma_{i_1}$ and $d_{i_2} \in \gamma_{i_2}$ such that $d(p, d_{i_1}) = d(p, d_{i_2}) = \varepsilon(\hat{\delta})$.
23: $P' = P' \cup \{d_{i_1}, d_{i_2}\}$.
24: Locally update $GVD(P')$, $\hat{\delta}$ and $\varepsilon(\hat{\delta})$.
25: **end if**
26: **end if**
27: **end for**
28: **end for**
29: **return** $GVD(P')$, which satisfies the closed ball property.

Let $\widetilde{C}(p)$ be a Voronoi cell that shares k Voronoi edges with multiple-order neighbors. The second strategy is to add an auxiliary site on each of these k Voronoi edges $\{\widetilde{b}_i\}_{i=1}^{k}$. Let $x_i \in \widetilde{b}_i$ be the position that minimizes the distance from p to $\widetilde{b}_i$. It is readily seen that adding an auxiliary site at x_i will remove the bisector $\widetilde{b}_i$ from $\widetilde{C}(p)$ in the updated $GVD(P \cup \{x_i\})$. However, the new Voronoi cell $\widetilde{C}(x_i)$ cannot be guaranteed to be free of pseudobisectors and multiple-order neighbors; see a counterexample in Figure 5. Akin to the first strategy, in practice we find that the second strategy also works well and that the extremely pathological cases like the one in

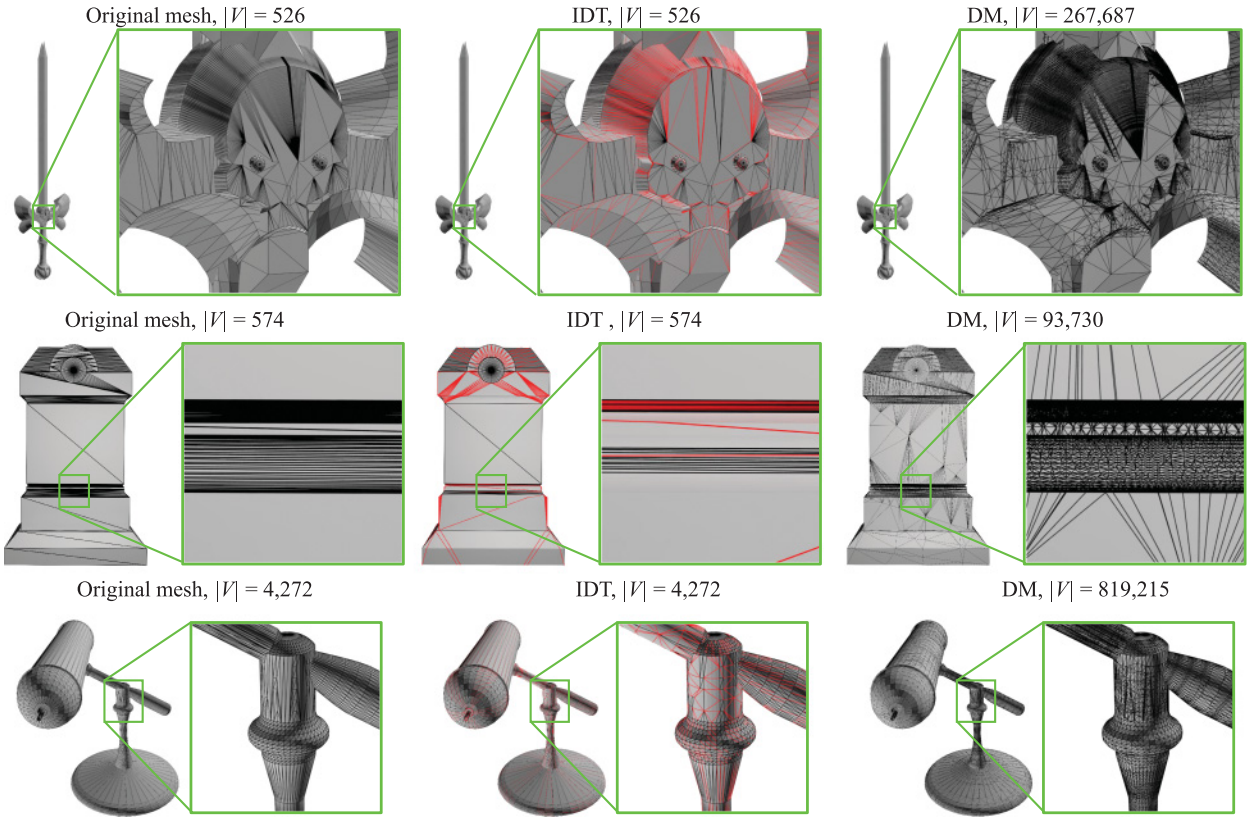


Fig. 20. Comparison with Delaunay meshes. Since the original meshes are highly anisotropic, Delaunay meshes require a large number of splitting points to ensure the local Delaunay condition. In contrast, IDT does not require any auxiliary site at all, as the sampling density on the cylinder-shaped regions is high enough.

Figure 5 are rare. Procedure 8 combines the second strategy (lines 10 through 13) with Procedure 4. Again, since locally computing $\widetilde{C}(x_i)$ and examining the existence of any pseudobisector and multiple-order neighbors in $\widetilde{C}(x_i)$ takes $O(n)$ time, Procedure 8 has the same time complexity as Procedure 4.

See the supplementary material for the results of Algorithm 6.

Handling degenerate cases. For constructing GVD and its dual IDT, a critical degenerate case is that more than three sites are on the same geodesic circumcircle, resulting in a nonempty intersection of more than three Voronoi cells. A common technique [Edelsbrunner and Mücke 1990] is to make a conceptual perturbation on these sites to eliminate all degeneracies. Consider four sites lying on a circumcircle: any arbitrarily small perturbation will result in two distinct circumcircles, each of which passes through three sites. However, the centers of these two circumcircles are arbitrarily close, leading to an arbitrarily small $\delta_{\max}(GVD(P))$ in Equation (1). We handle this case by storing a local adaptive $\delta_{\max}(p)$ for each Voronoi cell $\widetilde{C}(p)$ instead of using a global $\delta_{\max}(GVD(P))$. We found that together with two improved strategies in Procedures 7 and 8, this adaption works quite well in our practice. Elaborate techniques for handling all kinds of degenerate cases deserve further study in their own right and will be reported in future work.

6.4 Comparison with Delaunay Meshes

Delaunay meshes M [Dyer et al. 2007a] are manifold triangle meshes that form an IDT of its vertices with respect to the piecewise flat metric of its polyhedral surface. In other words, the proper IDT associated to a Delaunay mesh is just the mesh itself. Liu et al. [2015] showed that for an arbitrary manifold triangle mesh M with n vertices, there exists a Delaunay mesh with $O(Kn)$ vertices, where K is a model-dependent constant. They also developed an efficient algorithm to construct Delaunay mesh in $O(nK \log K)$ time.

Both proper IDT and Delaunay mesh have exactly the same geometry of M and satisfy the Delaunay condition everywhere. However, they differ in representation, space complexity, and time complexity.

Representation. An IDT edge is a geodesic path that may go through many triangles. Hence, it is tedious to explicitly represent IDT edges. Usually, one often takes the IDT as an abstract surface representation—that is, ignoring the geometry of geodesic paths and storing only their lengths. To apply IDT to the subsequent geometry processing algorithms, one has to adapt their interfaces for the abstract representation. In contrast, Delaunay meshes are just normal triangle meshes that can be represented by any mesh data structures, such as half edges and triangle soup. Consequently, the existing geometry processing algorithms can benefit the favorable properties of Delaunay meshes without changing any code.

Space complexity. The price to pay for the flexible representation of Delaunay meshes is its high space complexity. As shown in Liu et al. [2015], a Delaunay mesh has $O(Kn)$ vertices, where the constant K depends on the triangulation quality. We observe that

for anisotropic meshes (especially the ones made by artists), K can be very large (even comparable with n). In contrast, we prove that a proper IDT has $O(n)$ vertices—that is, its space complexity is independent of the mesh quality. Take the Lamp model (Figure 20) as an example. The Delaunay mesh has almost 200 times more vertices than the original mesh, whereas the IDT does not require any auxiliary sites.

Time complexity. The algorithms for constructing IDTs and Delaunay meshes are also different. To construct a proper IDT, we need to compute the GVD, which takes $O(n^2 + tn \log n)$ time. The Delaunay mesh construction algorithm simply flips the flippable NLD edges and splits the nonflippable NLD edges until all edges are local. The Delaunay mesh construction algorithm is fairly efficient for meshes with good triangulation quality due to its $O(nK \log K)$ time complexity. However, for anisotropic meshes such as Sword and Lamp (see Figure 20), where a large number of splitting points were added, constructing Delaunay mesh is highly expensive. In contrast, the performance of our method is not sensitive to the mesh quality.

In summary, both proper IDT and Delaunay mesh have their merits and limitations. Delaunay mesh is effective for triangle meshes with fairly good quality, whereas proper IDT is advantageous for anisotropic meshes.

7. CONCLUSIONS

This article presents a new method for constructing proper IDT on 2-manifold triangle meshes. Unlike the existing edge-flipping algorithm, our method is based on the dual graph of GVD. The key idea is to add auxiliary sites to the poorly sampled regions so that the closed ball property holds everywhere, and hereby the dual graph is a proper triangulation. We prove that our method produces a proper IDT by adding at most $O(n)$ sites for an n-vertex mesh. Moreover, thanks to the bounded time complexity of computing GVDs, our method has a theoretical worst-case time complexity $O(n^2 + tn \log n)$, where t is the number of obtuse angles in the mesh. Computational results show that it empirically runs in linear time $O(n)$ on real-world models.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their detailed and valuable comments, which helped us improve the quality of the article significantly. Special thanks go to Reviewer #1, who pointed out the case of Voronoi cells with multiple pseudobisectors.

REFERENCES

Alexander I. Bobenko and Boris A. Springborn. 2007. A discrete Laplace-Beltrami operator for simplicial surfaces. *Discrete and Computational Geometry* 38, 4, 740–756.

Jean-Daniel Boissonnat, Ramsay Dyer, and Arijit Ghosh. 2013. Constructing intrinsic Delaunay triangulations of submanifolds. arXiv:1303.6493.

Ramsay Dyer, Gert Vegter, and Mathijs Wintraecken. 2015. Riemannian simplices and triangulations. *Geometriae Dedicata* 179, 1, 91–138.

Ramsay Dyer, Hao Zhang, and Torsten Möller. 2007a. Delaunay mesh construction. In *Proceedings of the Symposium on Geometry Processing (SGP'07)*. 273–282.

Ramsay Dyer, Hao Zhang, and Torsten Möller. 2007b. Voronoi-Delaunay duality and Delaunay meshes. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*. 415–420.

Ramsay Dyer, Hao Zhang, and Torsten Möller. 2008. Surface sampling and the intrinsic Voronoi diagram. In *Proceedings of the Symposium on Geometry Processing (SGP'08)*. 1393–1402.

Herbert Edelsbrunner and Ernst Peter Mücke. 1990. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics* 9, 1, 66–104.

Herbert Edelsbrunner and Nimish R. Shah. 1997. Triangulating topological spaces. *International Journal of Computational Geometry and Applications* 7, 4, 365–378.

Matthew Fisher, Boris Springborn, Alexander I. Bobenko, and Peter Schröder. 2007. An algorithm for the construction of intrinsic Delaunay triangulations with applications to digital geometry processing. *Computing* 82, 2–3, 199–213.

C. Indermitte, Th. Liebling, M. Troyanov, and H. Clémençon. 2001. Voronoi diagrams on piecewise flat surfaces and an application to biological growth. *Theoretical Computer Science* 263, 1–2, 263–274.

Yong-Jin Liu. 2015. Semi-continuity of skeletons in 2-manifold and discrete Voronoi approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 9, 1938–1944.

Yong-Jin Liu, Zhanqing Chen, and Kai Tang. 2011. Construction of iso-contours, bisectors, and Voronoi diagrams on triangulated surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 8, 1502–1517.

Yong-Jin Liu and Kai Tang. 2013. The complexity of geodesic Voronoi diagrams on triangulated 2-manifold surfaces. *Information Processing Letters* 113, 4, 132–136.

Yong-Jin Liu, Chunxu Xu, Dian Fan, and Ying He. 2015. Efficient construction and simplification of Delaunay meshes. *ACM Transactions on Graphics* 34, 6, 174. DOI:http://dx.doi.org/10.1145/2816795.2818076

Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. 1987. The discrete geodesic problem. *SIAM Journal on Computing* 16, 4, 647–668.

Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung-Nok Chiu. 2000. *Spatial Tessellations: Concept and Applications of Voronoi Diagrams*. Wiley.

Igor Rivin. 1994. Euclidean structures on simplicial surfaces and hyperbolic volume. *Annals of Mathematics* 139, 3, 553–580.

Chunxu Xu, Yong-Jin Liu, Qian Sun, Jinyan Li, and Ying He. 2014. Polyline-sourced geodesic Voronoi diagrams on triangle meshes. *Computer Graphics Forum* 33, 7, 161–170.

Chunxu Xu, Tuanfeng Y. Wang, Yong-Jin Liu, Ligang Liu, and Ying He. 2015. Fast wavefront propagation (FWP) for computing exact geodesic distances on meshes. *IEEE Transactions on Visualization and Computer Graphics* 21, 7, 822–834.

X. Ying, X. Wang, and Y. He. 2013. Saddle vertex graph (SVG): A novel solution to the discrete geodesic problem. *ACM Transactions on Graphics* 32, 6, Article No. 170.

Zichun Zhong, Xiaohu Guo, Wenping Wang, Bruno Lévy, Feng Sun, Yang Liu, and Weihua Mao. 2013. Particle-based anisotropic surface meshing. *ACM Transactions on Graphics* 32, 4, Article No. 99.

Received January 2015; revised November 2016; accepted January 2017