Efficient Construction and Simplification of Delaunay Meshes

Yong-Jin Liu[*] Tsinghua University
Chun-Xu Xu Tsinghua University
Dian Fan Tsinghua University
Ying He[*] Nanyang Technological University

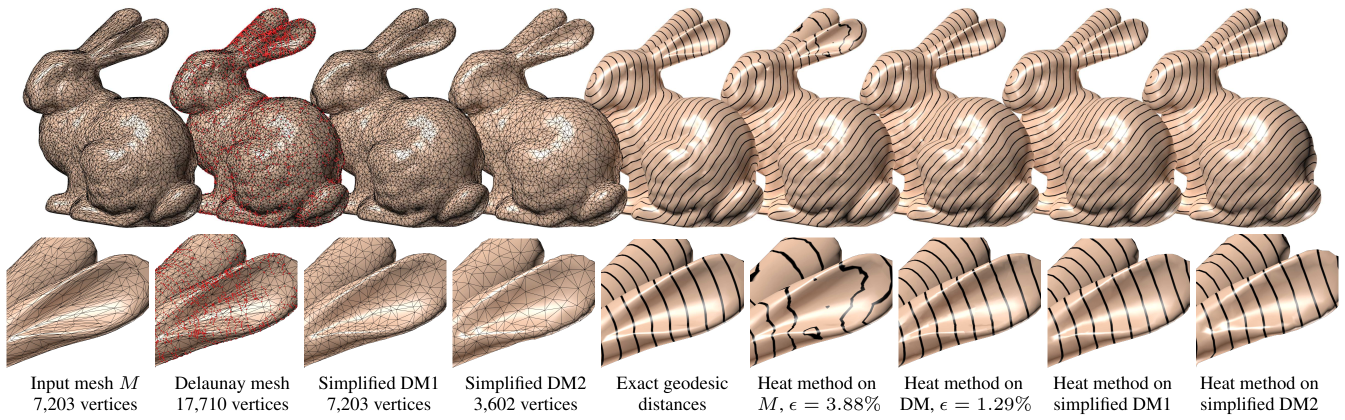


Figure 1: *We present an efficient algorithm to convert an arbitrary manifold triangle mesh M to a Delaunay mesh (DM), which has the same geometry of M. Our algorithm can also produce progressive Delaunay meshes, allowing a smooth choice of detail levels. Since DMs are represented using conventional mesh data structures, the existing digital geometry processing algorithms can benefit the numerical stability of DM without changing any codes. For example, DMs significantly improve the accuracy of the heat method for computing geodesic distances.*

Abstract

Delaunay meshes (DM) are a special type of triangle mesh where the local Delaunay condition holds everywhere. We present an efficient algorithm to convert an arbitrary manifold triangle mesh M into a Delaunay mesh. We show that the constructed DM has $O(Kn)$ vertices, where n is the number of vertices in M and K is a model-dependent constant. We also develop a novel algorithm to simplify Delaunay meshes, allowing a smooth choice of detail levels. Our methods are conceptually simple, theoretically sound and easy to implement. The DM construction algorithm also scales well due to its $O(nK \log K)$ time complexity.

Delaunay meshes have many favorable geometric and numerical properties. For example, a DM has exactly the same geometry as the input mesh, and it can be encoded by any mesh data structure. Moreover, the empty geodesic circumcircle property implies that the commonly used cotangent Laplace-Beltrami operator has non-negative weights. Therefore, the existing digital geometry processing algorithms can benefit the numerical stability of DM without changing any codes. We observe that DMs can improve the accuracy of the heat method for computing geodesic distances. Also, popular parameterization techniques, such as discrete harmonic mapping, produce more stable results on the DMs than on the input meshes.

[*]Corresponding authors: Y.-J. Liu (liuyongjin@tsinghua.edu.cn) and Y. He (yhe@ntu.edu.sg)

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

Keywords: Delaunay triangulation, Delaunay mesh, geodesic Voronoi diagram, Delaunay mesh simplification

1 Introduction

Delaunay triangulations are widely used in scientific computing in many diverse applications. While Delaunay triangulations in Euclidean space have been extensively studied and well understood, little progress has been made in Delaunay triangulations on curved manifolds. In sharp contrast to the Euclidean counterpart, Delaunay triangulations do not exist for an arbitrary set of points on a Riemannian manifold. If a Delaunay triangulation exists for a sufficiently dense set of points on a manifold, it has geodesic curves as edges and hereby is called *intrinsic Delaunay triangulation* (IDT). Using the strong convexity radius and the injectivity radius, Dyer et al. [2008] proposed adaptive sampling criteria for constructing IDTs on smooth 2-manifolds. In the discrete setting, Rivin [1994] and Indermitte et al. [2001] defined IDT on polyhedral surfaces, where the IDT edges are geodesic paths (i.e., polylines). Similar to planar Delaunay triangulations, triangles in an IDT satisfy the Delaunay criterion, i.e., the geodesic circumcircle of an IDT triangle does not contain any vertices in its interior. IDT is desired to many graphics applications. For example, among all possible triangulations on a 2-manifold mesh M, the Delaunay triangulation minimizes the discrete Dirichlet energy of a piecewise linear function on M [Rippa 1990]. The commonly used cotangent formula of Laplace-Beltrami operator has non-negative weights if and only if the underlying triangulation is Delaunay [Bobenko and Springborn 2007].

Despite of the aforementioned nice features, IDT did not have found widespread acceptance in computer graphics, mainly because it is difficult to represent its geodesic-path-based edges using the existing mesh data structures, such as triangle soup, winged-edge, half-edge, quad-edge, etc. This paper focuses on Delaunay meshes, a special triangle mesh whose IDT is the mesh itself. As a simple, convenient, and versatile representation of surfaces, Delaunay meshes are the ideal input to the existing graphics pipeline. Delaunay mesh was first studied by Dyer et al. [2007], who proposed an algorithm to convert an arbitrary manifold triangle mesh to a Delaunay mesh. Their idea is to recursively split non-Delaunay edges and refine the local triangulations until all edges become locally Delaunay. Dyer et al.'s algorithm is conceptually simple and easy to implement, and has guarantees of correctness and termination in finite steps. However, their local refinement strategy is purely combinatorial, and it does not consider the local geometry. As a result, it often adds too many splitting points to the input mesh, significantly increasing its space complexity and hereby compromising the runtime performance of the follow-up geometric processing and/or rendering. See Figure 14 for an illustrative example and Section 6.1 for detailed discussion.

In this paper, we present a novel method to construct Delaunay meshes. Given an arbitrary manifold triangle mesh M with n vertices, our method converts M into a Delaunay mesh with $O(Kn)$ vertices and runs in $O(nK \log K)$ time, where K is a model-dependent constant. Unlike Dyer et al's algorithm, our method takes the local geometry into consideration when refining the non-Delaunay edges. As Figure 1 shows, our method adds on average only 2 splitting points to each non-Delaunay edge of the Bunny model. As a result, our method produces Delaunay meshes with significantly fewer vertices than Dyer et al's method. Following the powerful QEM framework [Garland and Heckbert 1997], we also develop an efficient algorithm to simplify Delaunay meshes, allowing a smooth choice of detail levels. Our methods are conceptually simple, theoretically sound and easy to implement. The DM construction algorithm also scales well due to its $O(nK \log K)$ time complexity.

DMs have many favorable geometric and numerical properties. A DM has exactly the same geometry as the input mesh, and it can be encoded by any mesh data structure. Due to the empty circumcircle property, the sum of the two angles facing an internal edge is no more than π, implying that the commonly used cotangent Laplace-Beltrami operator has guaranteed non-negative weights. As a result, many existing algorithms can benefit numerical stability of DMs without changing any codes. For example, we observe that DMs can improve the accuracy of the heat method for computing geodesic distances. See Figure 1. Popular surface parameterization methods, such as discrete harmonic mapping, also produce better results on the DMs than on the input meshes.

2 Related Work

There is a huge body of literature of Delaunay triangulation and its broad applications [Okabe et al. 2000; Cheng et al. 2012]. Due to limited space, we review only the most relevant works on intrinsic Delaunay structures. Similar to the planar case, edge flipping and the dual of Voronoi diagram are two commonly used methods for studying Delaunay triangulations on polyhedral surfaces.

Rivin [1994] defined intrinsic Delaunay triangulation (IDT) on polyhedral surfaces by the local Delaunay criterion, and claimed (but did not prove) the existence theorem via the edge flipping algorithm. Bobenko and Springborn [2007] proved the termination of the edge flipping algorithm. They also defined intrinsic Delaunay tessellation (whose faces are generically but not always triangular) via a global empty circle criterion and proved its existence and uniqueness. An IDT is then obtained from the Delaunay tessellation by triangulating the non-triangular faces. They pointed out that a Delaunay triangulation, while in general not unique, differs from another Delaunay triangulation only by edges with vanishing cotweights. Unlike the planar case where edge flipping takes $\Theta(n^2)$ time, it is difficult to obtain the time complexity for polyhedral surfaces. Fisher et al. [2007] observed the edge flipping algorithm runs very fast on real-world meshes and its runtime is easily dominated by subsequent numerical computing tasks.

Voronoi diagram, as the sibling of Delaunay triangulation, can be naturally extended from Euclidean planes to curved surfaces. However, it is tricky to study intrinsic Delaunay triangulations via the dual graph of Voronoi diagram, since IDTs do not exist for arbitrary set of points and there are certain density requirements to ensure that the triangulations can accurately represent both the topology and geometry of the manifold. Dyer et al. [2008] showed that if a geodesic Voronoi diagram satisfies the closed ball property [Edelsbrunner and Shah 1997] (see Section 3), its dual is an IDT. Using a local feature size function [Amenta and Bern 1999], they also presented an adaptive sampling criterion to ensure the closed ball property.

Centroidal Voronoi tessellation (CVT) is a special type of Voronoi diagram, where each Voronoi cell's generator coincides with its center of mass. The commonly used method for computing CVT is the Lloyd's method [Lloyd 1982], which iteratively moves the generators to the mass centers. Although conceptually simple and easy to implement, the Lloyd's method is a first-order optimization method and hereby inefficient in practice. Liu et al. [2009] proved that the CVT energy is C^2 continuous for convex domains with smooth density, as well as in most situations encountered in real-world applications. As a result, it is possible to minimize the CVT functional using Newton or quasi-Newton methods with fast convergence. CVT is a powerful computational tool to generate highly regular triangulations for arbitrary meshes [Yan et al. 2009]. Moreover, it can be easily extended to anisotropic metrics [Du and Wang 2005] and L_p distance metric [Lévy and Liu 2010]. However, the CVT energy is highly non-linear, thus, computing the global optimal is technically challenging.

Glickenstein [2005] introduced weighted triangulations, which associate each vertex with a scalar weight and generalize weighted-Delaunay/power diagrams on arbitrary polyhedral meshes. As an alternative representation, weighted triangulations have great flexibility in the location of dual vertices while maintaining primal-dual orthogonality. de Goes et al. [2014] defined an admissible set of discrete metrics for weighted triangulations and showed this augmented metric can be directly used to derive discrete differential operators, such as the Laplace-Beltrami operator, that retain important properties of their smooth counterparts. Weighted triangulations are closely related to well-centered meshes, in which each simplex contains its circumcenter in its interior. In the case of planar triangulations, each triangle is acute angled. VanderZee et al. [2007] presented an iterative algorithm to transform a given planar triangle mesh into a well-centered one by moving the interior vertices while keeping the connectivity fixed. Mullen et al. [2011] constructed well-centered planar triangulations by minimizing a functional of weighted circumcenters. To construct well-centered triangulations on 3D models, de Goes et al. [2014] alternatively optimized the vertex positions and their weights using the L-BFGS method, which converged quickly (in only 10 to 20 iterations).

A manifold mesh with only non-obtuse triangles is a *special* Delaunay mesh. Burago and Zalagaller [1960] proved the existence of acute triangulations of general 2-dimensional polyhedral surfaces. Saraf [2009] provided an elementary proof of the same result using completely different methods. Based on Saraf-type triangulations, Maehara [2011] proved that a polyhedral surface can be triangulated into $O(Cn)$ acute triangles, where C is a model-dependent constant. Maehara's complexity is similar to ours, however, our method constructs Delaunay meshes, allowing both obtuse and non-obtuse triangles. Therefore, our results are more general than acute-angled triangulations. Refer to [Zamfirescu 2002] for a survey of acute triangulations.

3 Mathematical Background

This section provides the necessary background on Delaunay meshes and geodesic Voronoi diagrams. Recall that a Delaunay triangulation for a set P of points in a plane is a triangulation $DT(P)$ such that no point in P is inside the circumcircle of any triangle in $DT(P)$. Now let us consider a manifold triangle mesh $M = (V, E, F)$, where V, E, F are the vertex, edge and face sets, respectively. For adjacent vertices v_i and v_j, we denote by (v_i, v_j) the edge connecting them. Throughout this paper, we denote $n = |V|$ the number of vertices in M and K a model-dependant constant.

Bobenko and Springborn [2007] defined the intrinsic Delaunay triangulation associated to M as follows: the IDT's vertex set is the same as that of M; every IDT edge is a geodesic path in M; and for each interior edge the local Delaunay criterion is satisfied, i.e., the sum of the opposite angles in the adjacent triangles is no more than π.

Since a geodesic path (especially a long one) goes through many triangles, it is tedious to explicitly represent IDT edges. As a result, one often takes the IDT as an *abstract* surface representation by ignoring the geodesic paths and storing only their lengths. To ease the representation of an intrinsic Delaunay structure on polyhedral surfaces, Dyer et al. [2007] introduced Delaunay meshes.

Definition 1 *(Delaunay Mesh) A Delaunay mesh M is a manifold triangle mesh that forms an intrinsic Delaunay triangulation of its vertices with respect to the piecewise flat metric of its polyhedral surface. In other words, the IDT associated to a Delaunay mesh is just the mesh itself.*

Dyer et al. [2007] proposed a simple edge flipping and refinement algorithm to construct Delaunay meshes. An internal edge $e \in E$ is *locally Delaunay* if the sum of the two angles facing e does not exceed π, otherwise e is an non-locally Delaunay (NLD) edge. An NLD edge is flippable if it has a zero dihedral angle, otherwise it is unflippable. Their algorithm first flips all flippable NLD edges without changing the geometry of M. Rather than flipping the remaining unflippable NLD edges, which will reduce the surface area and introduce shape distortion, their algorithm adopts a geometry-preserving remeshing strategy: given a non-flippable NLD edge $e = (p, q)$, it determines a splitting vertex $s \in e$ such that the length $d(s, p) = 2^k\delta$ (if p is a mesh vertex, otherwise $d(s, q) = 2^k\delta$) for some (possibly negative) integer $k \in \mathbb{Z}$, where the factor δ is any positive number. Splitting e at s creates two planar hereby flippable edges and two non-flippable NLD edges incident to s. The algorithm recursively flips the flippable NLD edges and splits the non-flippable NLD edges until all edges are locally Delaunay. See Figure 2 for an example. Dyer et al. proved the correctness and the termination of the algorithm. We call their algorithm *combinatorial*, since it does not take geometry into consideration. Moreover, their method lacks a bound of the added splitting vertices.

Our framework is built upon geodesic Voronoi diagrams (GVD) [Liu et al. 2011], which are a natural generalization of Voronoi diagrams on polyhedral surfaces.

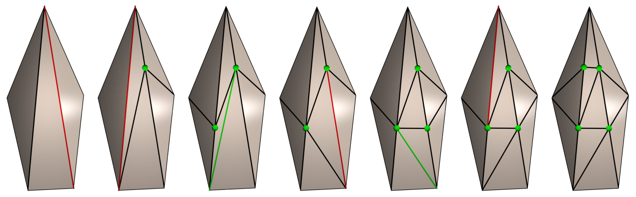


Figure 2: *Given a non-Delaunay mesh, Dyer et al.'s algorithm recursively splits the non-flippable NLD edges (red) until all edges are locally Delaunay. The green edges are flippable NLD edges and the green points are the splitting points added by the algorithm.*

Definition 2 *(Geodesic Voronoi Diagram). Let $P = \{p_i\}_{i=1}^m$ be a set of points in M. The Voronoi cell $C(p_i)$ corresponding to generator p_i is the set of all points in M whose distance to p_i is less than or equal to the distance to any other generators, i.e., $C(p_i) = \{q \in M | d(p_i, q) \leq d(p_j, q), \forall j \neq i\}$, where $d(x, y)$ is the geodesic distance between x and y. The geodesic Voronoi diagram of P, denoted by $GVD(P)$, is the union of all Voronoi cells.*

In contrast to the Euclidean case, not every GVD has a dual triangulation. Dyer et al. [2008] showed that a GVD has a dual triangulation if it satisfies the following three conditions, which are known as the closed ball property [Edelsbrunner and Shah 1997]:

1. **Disk condition**: each Voronoi cell is homeomorphic to a planar disk;
2. **2-cell intersection condition**: the intersection of any two Voronoi cells is either empty or a single Voronoi edge;
3. **3-cell intersection condition**: the intersection of any three Voronoi cells is either empty or a single Voronoi vertex.

Note that in the closed ball property, the generators are assumed to be in general position. In practice, a small perturbation of generators is helpful to deal with the degenerate cases, such as two Voronoi cells intersecting at a single point. Dyer et al. [2008] also showed that if the GVD has at least four distinct sites and both the disk condition and 2-cell intersection conditions are satisfied, then the 3-cell intersection condition is redundant.

4 Constructing Delaunay Meshes

In this section, we first present a sufficient condition of Delaunay mesh, and show the existence of DM via a constructive proof. We then present an efficient algorithm for constructing DM on arbitrary manifold triangle meshes. To ease the presentation, we assume that the input mesh M is *closed* in this section and then deal with meshes with *boundaries* in Supplementary Material.

4.1 Sufficient Condition & Existence

Definition 3 *Let $p \in M$ be an arbitrary point on a closed manifold triangle mesh M. The 1-ring neighborhood $N_1(p)$ of p is defined as follows:*

- *If p is a vertex $p \in V$, $N_1(p)$ consists of the triangles incident to p;*
- *If p lies on an edge e, $N_1(p)$ consists of the two triangles adjacent to e;*
- *If p is inside a triangle f, $N_1(p)$ is f.*

Theorem 1 *(Delaunay Mesh Condition for Closed Meshes) Let $P \supseteq V$ be a set of points on M including all vertices. If for every point $p \in P$, the Voronoi cell $C(p)$ is contained in $N_1(p) \setminus \partial N_1(p)$, then the dual of the geodesic Voronoi diagram $GVD(P)$ is a Delaunay mesh.*

Proof. Let $\mathring{N}_1(p) \triangleq N_1(p) \setminus \partial N_1(p)$ denote the interior of $N_1(p)$. Consider two adjacent Voronoi cells $C(p_i)$ and $C(p_j)$. We first show that their generators are in each other's 1-ring neighborhood, i.e., $p_i \in N_1(p_j)$ and $p_j \in N_1(p_i)$. Assume $p_i \notin N_1(p_j)$. Then $\mathring{N}_1(p_i) \cap \mathring{N}_1(p_j) = \emptyset$. Since $C(p_i) \subset \mathring{N}_1(p_i)$, the bisector of p_i and p_j is outside $\mathring{N}_1(p_j)$. As a result, $C(p_j) \not\subset \mathring{N}_1(p_j)$, which is a contradiction. See Figure 3(a) for an example of p_i being a mesh vertex.

Secondly, since for any two adjacent Voronoi cells $C(p_i)$ and $C(p_j)$, p_i and p_j are in each other's 1-ring neighborhood, it is read-

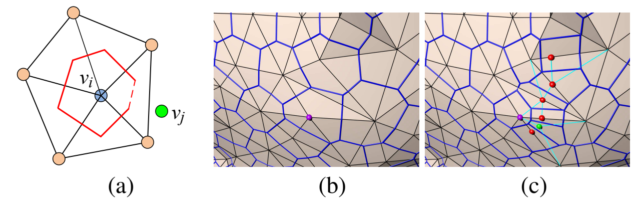


Figure 3: *Delaunay mesh condition. (a) Proof of Theorem 1. The Voronoi cells $C(p_i)$ and $C(p_j)$ are adjacent. If $p_i \notin N_1(p_j)$, then $C(p_j) \not\subset \mathring{N}_1(p_j)$. The dashed red line segment is the shared boundary of $C(p_i)$ and $C(p_j)$. (b) The mesh does not satisfy the Delaunay mesh condition. (c) Local refinement of Voronoi cells that violate the Delaunay mesh condition (cf. Algorithm 1). The red vertices are the auxiliary points added to the NLD edges and the cyan edges are the new Delaunay edges.*

ily seen that (1) the shortest geodesic connecting p_i and p_j is unique and is a straight line segment, and (2) the geodesic Voronoi diagram $GVD(P)$ satisfies the closed ball property. That completes the proof. □

Unfortunately, most real-world meshes are far from Delaunay (see Figure 3(b) for an example). To convert an arbitrary triangle mesh to a Delaunay mesh with the same geometry, a possible way is to add sufficient number of auxiliary points on mesh edges so that each Voronoi diagram is small enough to be in the 1-ring neighborhood of its generator. See Figure 4.

Denote $l_{\max}$ and $l_{\min}$ the maximal and minimal edge lengths in M, and $\theta_{\min}$ the minimum angle in M. Define

$$\rho_v \triangleq \min\left\{\frac{l_{\min}\sin\theta_{\min}}{0.5+\sin\theta_{\min}}, \frac{l_{\min}}{2}\right\}, \tag{1}$$

and

$$\rho_e \triangleq 2\rho_v \sin\theta_{\min}, \tag{2}$$

Definition 4 *(Delaunay Sampling Criterion) Let $S = \{s_1, s_2, \ldots\}$ be a set of points, where each point s_i is on some edge of M. We say S satisfies the Delaunay mesh sampling criterion if the following conditions hold:*

- *Each edge $e = (v_{i_1}, v_{i_2}) \in E$ has at least one point of S. Let $s_{k_1}, s_{k_2} \in e$ be the (possibly identical) points closest to v_{i_1} and v_{i_2}, respectively. Then $d(s_{k_1}, v_{i_1}) = d(s_{k_2}, v_{i_2}) = \rho_v$;*
- *Given any two adjacent samples s_i and s_j (if exists) on an edge $e \in E$, their Euclidean distance is less than or equal to ρ_e.*

Now we show that any closed manifold triangle mesh M has an associated Delaunay mesh $DM(M)$, which has exactly the same geometry of M.

Theorem 2 *(Existence of DM for Closed Meshes) If the point set S defined on a closed mesh $M = (V, E, F)$ satisfies the Delaunay mesh sampling criterion, then the geodesic Voronoi diagram $GVD(V \bigcup S)$ satisfies the Delaunay mesh condition, hereby its dual graph $IDT(V \bigcup S)$ is a Delaunay mesh, whose space complexity is $O(Kn)$, where $K = \frac{l_{\max}}{l_{\min}\sin^2\theta_{\min}}$.*

Proof. Note that the generator set $V \bigcup S$ consists of two sets V and S, we prove the Delaunay mesh condition holds for generators of each set separately.

First, we prove by contradiction that for any sample $s \in S$, the Voronoi cell $C(s)$ is contained in $\mathring{N}_1(s)$, i.e., $C(s) \subset \mathring{N}_1(s)$. Assume there is a Voronoi cell $C(s)$ and a point $q \in C(s)$ but $q \notin \mathring{N}_1(s)$. Let $e \in E$ be the edge containing s. The geodesic $\gamma(q, s)$ cannot pass through any vertices v incident to e, otherwise $q \in C(v)$. Then geodesic $\gamma(q, s)$ is a line segment $\overline{qs}$, which intersects an edge e' incident to e at a point $x \in C(s)$. There are three cases to consider:

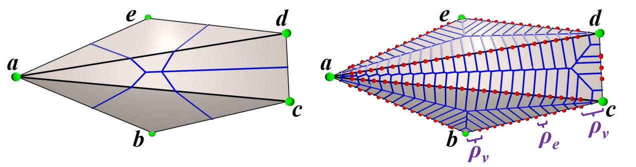


Figure 4: *The input mesh M (left) does not satisfy the Delaunay mesh condition, since the Voronoi cells $C(b)$ and $C(e)$ are not in their 1-ring neighborhoods. We can convert M into a Delaunay mesh (right) by adding sufficient number of auxiliary points (shown in red) on edges. The Delaunay mesh sampling criterion requires 1) any two adjacent auxiliary points are no more than ρ_e apart; and 2) each endpoint is ρ_v to the edge's endpoint.*

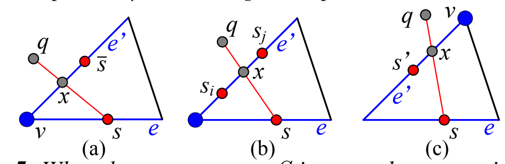
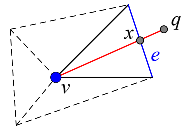


Figure 5: *When the generator $s \in S$ is a sample, we consider three cases of geodesic $\gamma(s, q)$. The red dots are the sample points placed on mesh edges.*

- Case 1 (see Figure 5(a)): on edge e', the generators closest to x from different sides are sample point $\bar{s}$ and vertex v which is an endpoint of e. In this case, the Euclidean distances $\|v\bar{s}\| = \rho_v$ and $\|vs\| \ge \rho_v$, implying that points q and $\bar{s}$ are on the same side of the bisector of s and $\bar{s}$. Therefore, point q is closer to $\bar{s}$ than to s, i.e., $q \in C(\bar{s})$, which is a contradiction.
- Case 2 (see Figure 5(b)): on edge e', the generators closest to x from different sides are sample points s_i and s_j. In this case, the shortest distance from x to edge e, denoted by $d(x, e)$, satisfies $d(x, e) \ge \rho_v \sin\theta_{\min} = \rho_e/2$. Then $\|xs\| \ge d(x, e) \ge \rho_e/2$. Since $\|s_i s_j\| \le \rho_e$, x is closer to one of s_i and s_j than to s. Thus, $x \notin C(s)$ hereby $q \notin C(s)$, which is a contradiction.
- Case 3 (see Figure 5(c)): on edge e', the generators closest to x from different sides are sample point s' and vertex v'. In this case, the shortest distance $d(x, e)$ from point x to edge e satisfies $d(x, e) \ge (l_{\min} - \rho_v)\sin\theta_{\min}$. If $\rho_v = \frac{l_{\min}\sin\theta_{\min}}{0.5+\sin\theta_{\min}}$, we have $(l_{\min} - \rho_v)\sin\theta_{\min} = \rho_v/2$. If $\rho_v = l_{\min}/2$, then $\frac{l_{\min}\sin\theta_{\min}}{0.5+\sin\theta_{\min}} \ge \frac{l_{\min}}{2} \Rightarrow \sin\theta_{\min} \ge 0.5$. Thus, $(l_{\min} - \rho_v)\sin\theta_{\min} = l_{\min}\sin\theta_{\min}/2 \ge l_{\min}/4 \ge \rho_v/2$. In either case, we have $d(x, e) \ge \rho_v/2$. Since x is closer to one of s' and v' than to s, $x \notin C(s)$ hereby $q \notin C(s)$, which is a contradiction.

Next, we show that for any vertex $v \in V$, the Voronoi cell $C(v)$ is contained in $\mathring{N}_1(v)$. Suppose it is not true. Then there exists a Voronoi cell $C(v)$, $v \in V$, which contains at least one point $q \notin \mathring{N}_1(v)$ (see the right inset). Then geodesic $\gamma(q, v)$ must intersect an edge e opposite to v. According to the aforementioned three cases, we can show that $x \notin C(v)$ and hereby $q \notin C(v)$, a contradiction. This completes the proof that the dual graph $IDT(V \bigcup S)$ is a Delaunay mesh.



Finally, we show the space complexity of $IDT(V \bigcup S)$. Note that for each mesh edge of length l_e, there are $\lceil \frac{l_e - 2\rho_v}{\rho_e} \rceil + 1 \le \lceil \frac{l_{\max}}{2\rho_v \sin\theta_{\min}} \rceil + 1$ samples. With straightforward calculation, we obtain the space complexity $O(Kn)$, where $K = \frac{l_{\max}}{l_{\min} \sin^2 \theta_{\min}}$. $\square$

The existence theorem leads to a naïve DM construction algorithm that adds $\lceil \frac{l_e - 2\rho_v}{\rho_e} \rceil + 1$ candidates on edge e. However, this simple strategy is very conservative for real-world models. Note that the Delaunay sampling criterion is sufficient but not necessary. Here we present a practical algorithm (Algorithm 1) to construct DMs by taking only a subset of S. The algorithm is conceptually simple and takes $O(K^2 n^2 \log(Kn))$ time. We then improve the time complexity to $O(nK \log K)$ in the next subsection.

Algorithm 1 Constructing Delaunay Mesh via Geodesic Voronoi Diagram

```
Input: A closed manifold triangle mesh $M = (V, E, F)$
Output: The $DM(M) = (V_D, E_D, F_D)$ such that $V \subseteq V_D$, $\cup F = \cup F_D$ and $|V_D| = O(K|V|)$ for a model-dependent constant $K$
 1: For each edge $e \in E$, construct a set $C_e$ of candidate samples satisfying the Delaunay mesh sampling criterion.
 2: Set the Voronoi generators $S = V$.
 3: Build the geodesic Voronoi diagram $GVD(S)$.
 4: Place the Voronoi cells that are outside the 1-ring neighborhood of their generators into a queue $\mathcal{Q}$.
 5: while $\mathcal{Q}$ is not empty do
 6:     Pop the top element $C(s)$ from $\mathcal{Q}$.
 7:     Find a mesh edge $e \in E$ on the the boundary of $N_1(s)$ where $C(s)$ crosses. Continue if such an edge does not exist, since $C(s)$ may have already been fixed by some previously added sample.
 8:     Find the candidate sample $c \in C_e$ but $c \notin S$ that is closest to $s$.
 9:     Add $c$ into $S$ and locally update the Voronoi diagram.
10:     If $C(s) \not\subset \mathring{N}_1(s)$, put $C(s)$ into $\mathcal{Q}$.
11:     If $C(c) \not\subset \mathring{N}_1(c)$, put $C(c)$ into $\mathcal{Q}$.
12: end while
13: Construct the DM via the dual graph of $GVD(S)$.
```

We first generate for each edge e a set $\mathcal{C}_e$ of candidate samples satisfying the Delaunay mesh sampling criterion. Taking the mesh vertices as the Voronoi generators, we compute the geodesic Voronoi diagram using [Liu et al. 2011]. Then we iteratively fix the Voronoi cells that violate the Delaunay mesh condition. Let $C(s)$ be a Voronoi cell that is outside the 1-ring neighborhood $N_1(s)$ of its generator s. Denote by $\partial N_1(s)$ the boundary of $N_1(s)$. Pick an edge $e \in \partial N_1(s)$ that crosses the Voronoi cell $C(s)$. Among the candidate samples on e, we find the one $s' \notin S$ which is closest to s. We then add s' into the generator set S and locally update the Voronoi diagram. If the updated cell $C(s)$ still crosses edge e, we need to further add sample points to e (see Figure 6). Note that the new Voronoi cell $C(s')$ may also violate the Delaunay mesh condition. If so, we need to further refine it by adding additional points, again taken from the pre-defined pools of candidate samples.

Intuitively speaking, Algorithm 1 aims at reducing the sizes of Voronoi cells that violate the Delaunay mesh condition. In each iteration, the algorithm adds a generator taken from a large pool of candidates and it terminates when all Voronoi cells are within their 1-ring neighborhood of their generators. Since there are $O(K)$ candidates on each edge, Algorithm 1 is guaranteed to terminate in $O(Kn)$ steps.

Constructing the geodesic Voronoi diagram $GVD(V)$ takes $O(n^2 \log n)$ time [Liu et al. 2011]. In each iteration, it takes $O(Kn)$ time to locally update the Voronoi diagram for the new generator. Finally, constructing the dual graph of $GVD(S)$ takes $O(Kn)$ time. Putting them all together, Algorithm 1 constructs a DM in $O(K^2 n^2 \log(Kn))$ time.

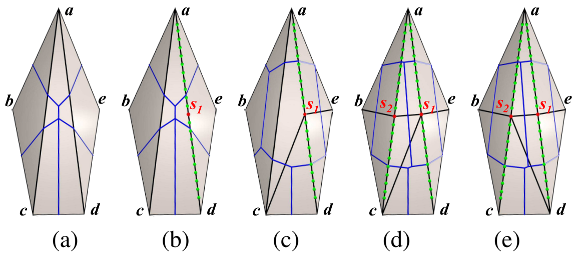


Figure 6: *Illustration of Algorithm 1. (a) Voronoi cells $C(b)$ and $C(e)$ are outside the 1-ring neighborhood of the corresponding generators. (b) Among the candidate samples (in green) on edge (a, d), pick sample s_1 which is closest to e. (c) Locally update the Voronoi diagram at the new generator s_1. Note that both $C(b) \subset \mathring{N}_1(b)$ and $C(e) \subset \mathring{N}_1(e)$, but the new Voronoi cell $C(s_1)$ violates the Delaunay mesh condition. (d) Find the new generator $s_2 \in (a, c)$ which is closest to s_1. Now $C(s_1) \subset \mathring{N}_1(s_1)$. Note that $s_2 \in (a, c)$ is a sample point whose 1-ring neighborhood $N_1(s_2) = \triangle abc \cup \triangle acd$. Thus, $C(s_2) \subset \mathring{N}_1(s_2)$. (e) The final Delaunay mesh is obtained by the dual graph of the GVD with 7 generators $\{a, b, c, d, e, s_1, s_2\}$. One can clearly see that each Voronoi diagram is within the 1-ring neighborhood of its generator.*

4.2 An $O(nK \log K)$-Time Construction Algorithm

As a *direct* approach, Algorithm 1 is conceptually simple and intuitive. However, constructing GVD is computationally expensive, hereby diminishes its application to large-scale models. Here we present an *indirect* algorithm, running in $O(nK \log K)$ time. This algorithm is easy to implement since it computes neither geodesic distances nor Voronoi diagrams. The key idea is to split the non-flippable NLD edges and re-tessellate their local triangulations to Delaunay triangles. Note that the splitting points, if chosen carelessly, may break the Delaunay condition for the opposite edges. We show that carefully selected splitting points can minimize ongoing headache.

We consider the butterfly-shaped local region of an NLD edge e, and define five $I_i \in e$, $i = 0, \cdots, 4$, as shown in Figure 7. Note that the interval I_0 is always non-empty, and the other intervals I_i, $1 \le i \le 4$, may be empty, depending on the geometry of triangle t_i and the position of edge e.

Observe that splitting edge e at a point $s \in I_0$ produces two edges (a, s) and (s, b), which are both locally Delaunay. Similarly, splitting e at a point $s \in I_i$, $1 \le i \le 4$ makes the edge e_i locally Delaunay. Our goal is to find a point $s \in e$ so that we can produce the maximal number of Delaunay edges by splitting e at s. Toward this goal, we find an interval $I_s \in I_0$, which maximizes the number of intersections between I_s and $\{I_1, \cdots, I_4\}$.

Note that after splitting, the Delaunay condition of edges (a, c), (c, b), (b, d) and (d, a) becomes even worse, since the sum of the two subtended angles for each edge strictly increases. If such a neighboring edge becomes non-flippable NLD, we need to further split it using the above maximal intersection strategy. As we will show later, the edge splitting process is guaranteed to terminate in $O(n)$ steps. We also observe that the complexity of the generated DM depends on the order of processing non-flippable NLD edges. We maintain these NLD edges using a stack, whose last-in-first-out order gives the most recent edges higher priority. Computational results show that the stack-based implementation produces 10-15% fewer splitting points than the queue-based implementation. We outline the geometry-aware NLD edge refinement in Algorithm 2 and illustrate the pipeline on a toy model in Figure 8.

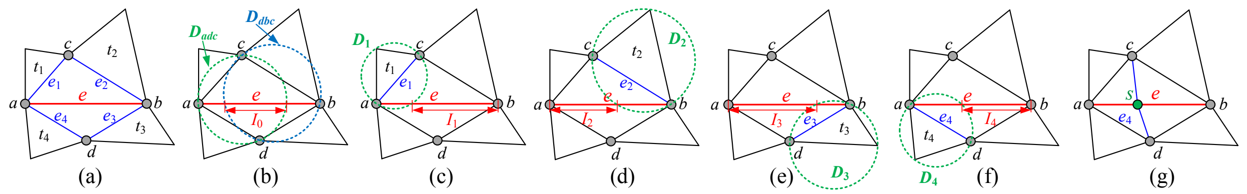
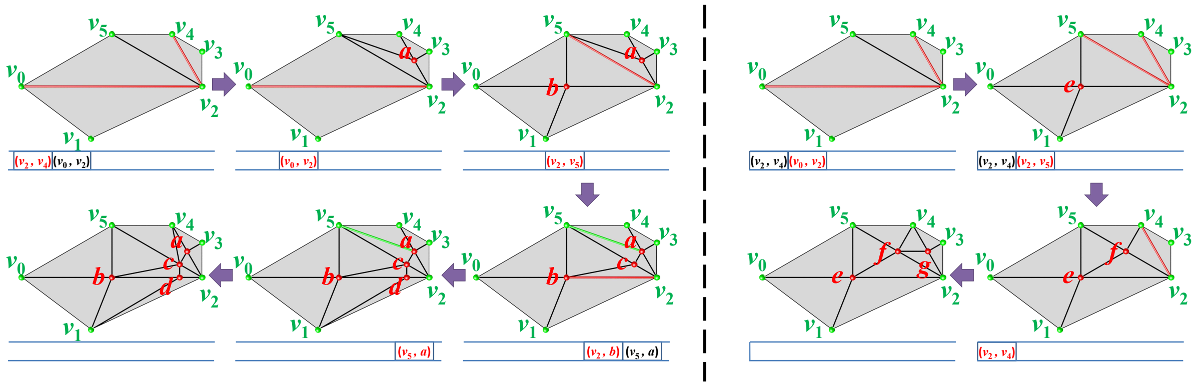


Figure 7: *The local geometry of an NLD edge e. (a) shows the butterfly shape* $W(e) = \triangle abc \cup \triangle adb \cup_{i=1}^{4} t_i$ *after making all triangles coplanar. (b)* D_{adc} *(resp.* D_{dbc}*) is the disk bounded by the circumcircle of* $\triangle adc$ *(resp.* $\triangle dbc$*). Define interval* $I_0 \triangleq D_{adc} \cap D_{dbc} \cap e$*. Splitting edge e at a point* $s \in I_0$ *produces two edges* $e = (a, s) \cup (s, b)$*, which are locally Delaunay. (c)-(f)* D_i*,* $i = 1, \cdots, 4$*, is the disk bounded by the circumcircle of triangle* t_i*. Define interval* $I_i \triangleq e \setminus (e \cap D_i)$*. Splitting e at a point* $s \in I_i$*,* $1 \leq i \leq 4$ *makes the edge* e_i *locally Delaunay. (g) Find a point* $s \in e$ *so that splitting e at s produces the maximum number of Delaunay edges.*



(a) Queue-based implementation
(b) Stack-based implementation

Figure 8: *Illustration of Algorithm 2 on a non-planar toy model with 2 non-flippable NLD edges* (v_0, v_2) *and* (v_2, v_4)*. Note that splitting an non-flippable NLD edge could turn some neighboring Delaunay edges into non-Delaunay. For example, after splitting* (v_0, v_2) *at b, the edge* (v_2, v_5) *becomes non-Delaunay (see Step 3 in (a)). We observe that the complexity of DM depends on the order of processing non-flippable NLD edges. The queue-based implementation adds 4 splitting points. In contrast, the stack-based implementation, which gives the recently generated NLD edges higher priority, produces only 3 points. The non-flippable and flippable NLD edges are colored in red and green, respectively. The diagrams below the toy model show snapshots of the data structure and the red tuple is the current non-flippable NLD edge.*

Algorithm 2 Constructing Delaunay Mesh via Geometry-aware NLD Edge Refinement

```
Input: A closed manifold triangle mesh $M = (V, E, F)$
Output: A DM $(V_D, E_D, F_D)$ such that $V \subseteq V_D$, $\cup F = \cup F_D$
    and $|V_D| = O(K|V|)$
 1: For each edge $e \in E$, construct a set $C_e$ of candidate samples
    satisfying the Delaunay mesh sampling criterion.
 2: Push all NLD edges into a stack $\mathcal{S}$.
 3: while $\mathcal{S}$ is not empty do
 4:     Pop a non-flippable NLD edge $\widetilde{e}$ from $\mathcal{S}$.
 5:     Find the mesh edge $e \in E$ containing $\widetilde{e}$, i.e., $\widetilde{e} \subseteq e$.
 6:     Find the interval $I_s \subset \widetilde{e}$ that maximizes the number of inter-
        sections with $I_i$.
 7:     if $C_e \cap I_s \neq \emptyset$ then
 8:         Take a sample $s \in C_e \cap I_s$.
 9:     else
10:         Find the sample in $C_e$ which is nearest to $I_s$.
11:     end if
12:     Split edge $\widetilde{e}$ at $s$ and then remove $s$ from $C_e$.
13:     Find the two triangles $t_i, t_j \in F$ such that $t_i \cap t_j = e$.
14:     Flip all the flippable NLD edges in $t_i$ and $t_j$.
15:     for each edge $l \subseteq \partial t_i \cup \partial t_j$ do
16:         if $l$ is an non-flippable NLD edge then
17:             Push $l$ into $\mathcal{S}$.
18:         end if
19:     end for
20: end while
```

Theorem 3 *Algorithm 2 constructs a DM in* $O(nK \log K)$ *time.*

Proof. There are $\lceil \frac{l_e - 2\rho_v}{\rho_e} \rceil + 1$ of candidate samples on each edge. Thus, steps 1 and 2 take $O(Kn)$ time.

In each iteration, the algorithm first splits a non-flippable NLD edge $\widetilde{e}$ at a sample taken from the candidate set C_e on mesh edge $e \supseteq \widetilde{e}$ in $O(\log K)$ time. Then it flips all the flippable NLD edges within the triangles t_i and t_j which have a common edge e in $O(1)$ expected time [Guibas et al. 1992]. Finally, it pushes the non-flippable NLD edges on the boundary of t_i and t_j into the stack $\mathcal{S}$ in $O(1)$ time. Therefore, each iteration takes $O(\log K)$ time.

Also note that the number of NLD edges entering the stack $\mathcal{S}$ is $O(Kn)$. Therefore, Algorithm 2 runs in $O(nK \log K)$ time. □

5 Delaunay Mesh Simplification

Algorithm 2 converts an arbitrary triangle mesh into a DM by splitting its NLD edges, which increases the mesh complexity. In many applications, it is often desirable to use approximations in place of excessively detailed models. A naïve approach is to first reduce the mesh resolution using existing simplification tools (e.g., QEM [Garland and Heckbert 1997]), and then convert the simplified mesh into a DM using Algorithm 2. This approach, although being readily available, cannot guarantee the generated DM has the user-desired resolution.

In this section, we present an algorithm for Delaunay mesh simplification, which iteratively removes vertices and maintains the Delaunay condition until the mesh resolution is reduced to the user-specified value. To remove a vertex v, the algorithm collapses one of v's incident edges and then flip the other incident edges if necessary. In the following, we first define two types of vertices that can be *safely* removed without violating the local Delaunay condition, and then detail our DM simplification algorithm.

Note that our algorithm adopts the QEM framework [Garland and Heckbert 1997] by adding extra constraints for edge contraction. The issue of topological type preservation in edge contraction has been addressed in [Dey et al. 1999], which proved that the complex obtained after an edge contraction is homeomorphic to the original one if the neighborhood of the contracted edge satisfies a link condition.

5.1 Type-I Removable Vertices

Let us first consider the planar case.

Proposition 1 *Let $DT(S)$ be the Delaunay triangulation of a set S of points in $\mathbb{R}^2$ and $v \in S$ an interior point with degree less than 6. There exists an edge $(v, v') \in DT(S)$ such that the simplified triangulation of set $S \setminus \{v\}$, produced by contracting edge $(v, v') \to v'$, is still Delaunay.*

Proof. We first show that $DT(S \setminus \{v\})$ differs from $DT(S)$ only by the triangles inside the region $N_1(v)$. Note that for each triangle $t \in DT(S)$ and $t \notin N_1(v)$, its circumcircle does not contain any points in S. After removing v, the empty circumcircle condition still holds on t. Thus, t is a triangle in $DT(S \setminus \{v\})$.

Since the degree of v is less than 6, removing v and its incident edges produces a hole with no more than 6 boundary edges. Triangulating such a hole can always be regarded as an edge contraction $(v, v') \to v'$. □

It is worth noting that the condition of vertex degree less than 6 in Proposition 1 is sufficient but not necessary. We observe that real-world triangle meshes contain many vertices with degree greater than or equal to 6, which can also be safely removed via edge contraction without edge flipping. See Section 6 for detailed discussion. Based on this observation, we define the first type of removable vertices (Figure 9) as follows.

Definition 5 *A vertex $v \in V$ is type-I removable, if it has an incident edge e, whose contraction does not violate the local Delaunay condition, i.e., the mesh is still Delaunay after contracting e.*

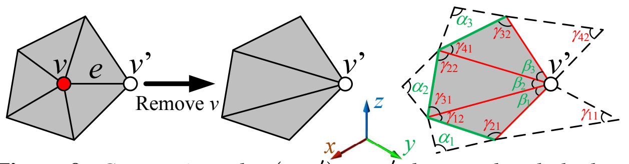


Figure 9: *Contracting edge $(v, v') \to v'$ does not break the local Delaunay condition, so vertex v is type-I removable. Specifically, for each edge opposite to v' (shown in green), the sum of the two subtended angles is less than π, $\alpha_i + \beta_i \leq \pi$; and for each edge incident to v', the sum of the two subtended angles is less than π, $\gamma_{j1} + \gamma_{j2} \leq \pi$.*

5.2 Type-II Removable Vertices

Still let us focus on planar Delaunay triangulation first. Type-I removable vertices allow us to *directly* simplify the Delaunay triangulation using only edge contraction. However, when type-I removable vertices are not available, we must seek other type of removable vertices by allowing edge flipping. Figure 10 shows such as an example, where the simplified Delaunay triangulation $DT(S \setminus \{v_0\})$ can only be obtained by both edge contraction and flipping. Note that by the first part of proof in Proposition 1, edge flipping does not apply for the edges opposite to v_0 and the edges outside of $N_1(v_0)$.

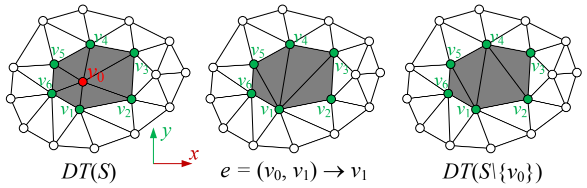


Figure 10: *Let $DT(S)$ be the Delaunay triangulation of a set S of planar points and $v_0 \in S$ an interior point. To obtain the Delaunay triangulation $DT(S \setminus \{v_0\})$, we contract an incident edge $e = (v_0, v_1) \to v_1$ and then flip edge (v_1, v_3) to (v_2, v_4). Note that the original edges opposite to v_0 remain unchanged, since they are still Delaunay edges. In other words, we can re-triangulate the hole without flipping any edges outside of $N_1(v_0)$.*

Generalizing this idea to 3D meshes, we define type-II removable vertices (Figure 11) as follows.

Definition 6 *A vertex $v \in V$ is type-II removable, if the resulting hole can be re-triangulated to Delaunay without flipping any edges opposite to v.*

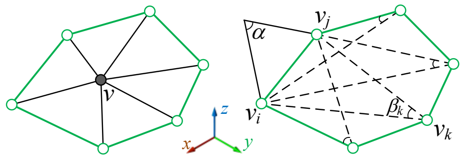


Figure 11: *Vertex v is type-II removable if for every opposite edge (v_i, v_j) (in green) and for every vertex $v_k \in N_1(v) \setminus \{v, v_i, v_j\}$, the sum of angles $\alpha + \beta_k \leq \pi$, where α is the subtended angle of (v_i, v_j) in the face outside $N_1(v)$ and β_k is the angle $\angle v_i v_k v_j$.*

5.3 Algorithm

Garland and Heckbert [1997] proposed an efficient algorithm for producing high quality approximations of polygonal models using quadric error metrics (QEM). Their algorithm associates a symmetric 4×4 matrix $\mathbf{Q}$ with each vertex $v = [v_x, v_y, v_z, 1]^T$, where the quadratic form $v^T\mathbf{Q}v$ measures the sum of squared distances from v to its neighboring triangles. For each edge (v_i, v_j), the optimal contraction target $\overline{v}$ that minimizes the error $\overline{v}^T(\mathbf{Q}_i + \mathbf{Q}_j)\overline{v}$ is computed. It places all the edges in a priority queue keyed on error with the edge of minimum error at the top. The QEM algorithm iteratively contracts the edge (v_i, v_j) of minimum error from the queue, and updates the errors of all edges incident to v_i.

To fit our vertex removal strategy into the QEM framework, we assign each vertex $v \in V$ a quadruple $(e_v, \mathbf{Q}_v, \varepsilon_v, \lambda_v)$, where e_v is v's incident edge to be contracted during the simplification, $\mathbf{Q}_v$ the QEM matrix associated with v, and ε_v the approximation error when edge e_v is contracted. It is worth noting that the quadratic errors of type-I and type-II vertices are of *different* units, since the error of a type-I vertex is the *accumulated* error during a sequence of edge contractions whereas the error of a type-II vertex measures only the *local* approximation error when that vertex is removed. We adopt a parameter λ_v to make the two types of error *unitless* so that they can be added together. Intuitively speaking, λ_v tracks the number of edge contractions applied in order to reach vertex v.

We initialize $e_v = NULL$, $\varepsilon_v = +\infty$ and λ_v the number of faces incident to vertex v. For each edge $e = (v_i, v_j)$ with at least one endpoint of type-I removable, say v_i, we compute the approximation error $\varepsilon_{v_i} = v_j^T(\mathbf{Q}_{v_i} + \mathbf{Q}_{v_j})v_j$, assuming that v_i is removed by contracting edge $(v_i, v_j) \to v_j$.

Next, we set the approximation error for the type-II removable vertices. Let v_i be a vertex of type-II but not type-I removable, and (v_i, v_j) the contracted edge. Note that after contracting edge $(v_i, v_j) \to v_j$, we must flip some edge(s) to ensure the Delaunay condition. Let LT_b (resp. LT_a) denote the local triangulation before (resp. after) edge flipping. We measure the difference between LT_b and LT_a by the squared Hausdorff distance:

$$d_H^2(B, LT_a) = \max_{b \in B}\{\min_{p \in LT_a} \|b - p\|_2^2\}, \tag{3}$$

where B is the set of mass centers for the triangles in LT_b. The approximation error ε_{v_i} for the type-II removable vertex v_i is given by $\varepsilon_{v_i} = v_j^T(\mathbf{Q}_{v_i} + \mathbf{Q}_{v_j})v_j + (\lambda_{v_i} + \lambda_{v_j})d_H^2(B, LT_a)$.

We place all vertices in a priority queue $\mathcal{Q}$ keyed on ε_v with the vertex of minimal error at the top. The algorithm then iteratively removes the vertex v of minimal error from the queue, contracts the edge $e_v = (v, v') \to v'$, and updates the quadruple $(e'_v, \mathbf{Q}_{v'}, \varepsilon_{v'}, \lambda_{v'})$ of v'. Since edge contraction may change the type of vertex v', we check its type and update it if necessary. We also need to update the vertex type, approximation error and associated edge for every vertex v'' in the 1-ring neighborhood of v'. The algorithm repeats this procedure until the number of vertices in the simplified mesh matches the user-specified value.

Algorithm 3 Delaunay Mesh Simplification

```
Require: A Delaunay mesh $M = (V, E, F)$ and the target vertex number $N_v (\leq |V|)$.
Ensure: A simplified Delaunay mesh with $N_v$ vertices.
 1: Initialize each vertex $v \in V$ a quadruple $(e_v, \mathbf{Q}_v, \varepsilon_v, \lambda_v)$, where $\mathbf{Q}_v$ is the QEM matrix, $e_v = NULL$, $\varepsilon_v = +\infty$ and $\lambda_v$ is the number of faces incident to $v$.
 2: for every edge $e = (v_i, v_j) \in E$ do
 3:     if $v_i$ is type-I removable then
 4:         $e_{v_i} = e$; $\varepsilon_{v_i} = v_j^T(\mathbf{Q}_{v_i} + \mathbf{Q}_{v_j})v_j$.
 5:     end if
 6:     if $v_j$ is type-I removable then
 7:         $e_{v_j} = e$; $\varepsilon_{v_j} = v_i^T(\mathbf{Q}_{v_i} + \mathbf{Q}_{v_j})v_i$.
 8:     end if
 9: end for
10: for every type-II removable vertex $v_i$ do
11:     Find $v_j \in N_1(v_i)$ that minimizes $v_j^T(\mathbf{Q}_{v_i} + \mathbf{Q}_{v_j})v_j$.
12:     $e_{v_i} = (v_i, v_j)$.
13:     Compute the squared Hausdorff distance $d_H^2$ using Equation (3).
14:     $\varepsilon_{v_i} = v_j^T(\mathbf{Q}_{v_i} + \mathbf{Q}_{v_j})v_j + (\lambda_{v_i} + \lambda_{v_j})d_H^2$.
15: end for
16: Place all vertices in a priority queue $\mathcal{Q}$ keyed on $\varepsilon_v$ with the vertex of minimal error at the top.
17: while $|V| > N_v$ do
18:     Extract the top vertex $v$ from $\mathcal{Q}$.
19:     Contract edge $e_v = (v, v') \to v'$.
20:     $\mathbf{Q}_{v'} = \mathbf{Q}_v + \mathbf{Q}_{v'}$ and $\lambda_{v'} = \lambda_v + \lambda_{v'}$.
21:     Update the type of $v'$.
22:     for every vertex $v'' \in N_1(v')$ do
23:         Update its type, approximation error $\varepsilon_{v''}$ and associated edge $e_{v''}$.
24:     end for
25: end while
```

6 Experimental Results & Applications

6.1 DM Construction

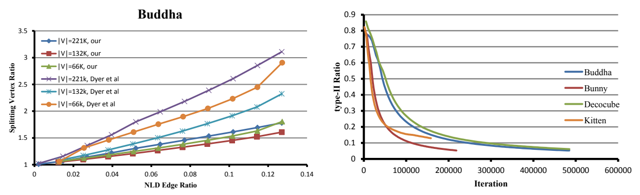
We implemented our DM construction and simplification algorithms in C++ and evaluated their performance on a wide-range of synthetic and real-world models. Running time was measured on a PC with an Intel Core i7-2600 CPU 3.40 GHz and 8GB memory.

Model	$\|V\|$	# non-flippable NLD edges	Dyer's method		Our method	
			N_s	T (s)	N_s	T (s)
Teapot	22,162	11,924 (17.9%)	88,017	0.73	27,571	0.34
Armadillo	66,544	35,368 (17.7%)	257,145	2.28	80,101	1.05
Kitten	70,442	37,411 (17.7%)	293,432	2.46	88,495	1.12
Bunny	98,996	51,665 (17.3%)	370,303	3.42	120,486	1.58
Gargoyle	111,137	57,091 (17.1%)	447,677	3.80	130,976	1.78
Lucy	155,814	82,110 (17.6%)	632,851	5.46	192,346	2.54
Decocube	219,032	116,647 (17.7%)	906,539	8.04	280,708	3.64
Buddha	221,400	116,818 (17.6%)	788,157	7.43	264,294	3.50
Blade	500,000	194,448 (12.9%)	1,180,363	9.06	431,625	4.24
Dragon	1,000,000	450,302 (15.1%)	3,249,611	33.97	1,373,154	15.35
Thai Statute	2,000,000	756,104 (12.6%)	4,983,205	55.74	1,920,739	22.17

Table 1: *Mesh complexity and runtime performance. The percentage is the NLD edge ratio, defined as the number of non-flippable NLD edges to the total number of edges. N_s is the number of splitting points required to ensure the Delaunay condition and the running time T was measured in seconds on an Intel Core i7-2600 CPU 3.40 GHz. Our method adds significantly fewer splitting points than Dyer et al's algorithm, and it also runs two times faster.*

The proposed DM construction algorithms (cf. Algorithm 1 and Algorithm 2) produce Delaunay meshes with very similar complexity. Since the edge refinement algorithm has better runtime performance, we report its results only. We define the NLD edge ratio R, defined as the number of non-flippable NLD edges to the total number of edges. Table 1 shows that real-world meshes have large NLD edge ratios (e.g., more than 10%), meaning that they are far from Delaunay triangulations. As proved in Section 4, the generated Delaunay mesh has $O(Kn)$ combinatorial complexity. In practice, we observe that the DM complexity is linearly proportional to the ratio of NLD edges. As Figure 12(a) and Figure 18 show, the complexity of DM depends on the quality of the input mesh. For common models with fair triangulations, our DMs have roughly 1.5 to 2.5 times the number of vertices of the input mesh. Due to its linear time complexity, our method is highly efficient and it takes only a few seconds on 500K-vertex models.

We also compare our method with Dyer et al.'s algorithm. As Figures 14 and 15 show, our geometry-aware NLD edge refinement is more effective to improve the Delaunay quality than their combinatorial refinement. In practice, our method generates only 20-30% splitting points as Dyer et al's method, hereby our method outruns theirs by a factor of 2-3.



(a) DM complexity (b) Statistics of removable vertices

Figure 12: *(a) We observe that the DM complexity is linearly proportional to the ratio of NLD edges. The horizontal axis is the NLD edge ratio (i.e., the number of non-flippable NLD edges to the total number of edges) and the vertical axis is the splitting vertex ratio (i.e., the number of splitting vertices added to the number of vertices in the input mesh). (b) Statistics of type-I and type-II removable vertices in a sequence of simplified DMs. At the beginning, most of the deleted vertices are of type II. When the DM resolution becomes low, type-I vertices are dominant.*

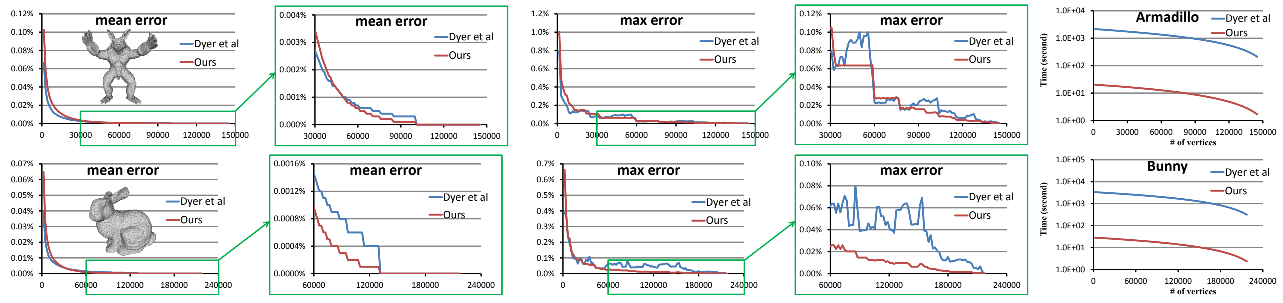


Figure 13: *Evaluating the runtime performance and accuracy of the DM simplification algorithms. The horizontal axis is mesh resolution and the vertical axes show the mean and maximal approximation errors as well as running time (in logarithm). The original Delaunay meshes have 146K and 219K vertices, respectively. We gradually reduce the mesh resolution while keeping the Delaunay condition. Our method runs consistently faster than Dyer et al's algorithm in all resolutions. Our results are slightly better in high resolution, whereas Dyer et al's results are more accurate when the resolution is low. See Supplementary Material for more results.*

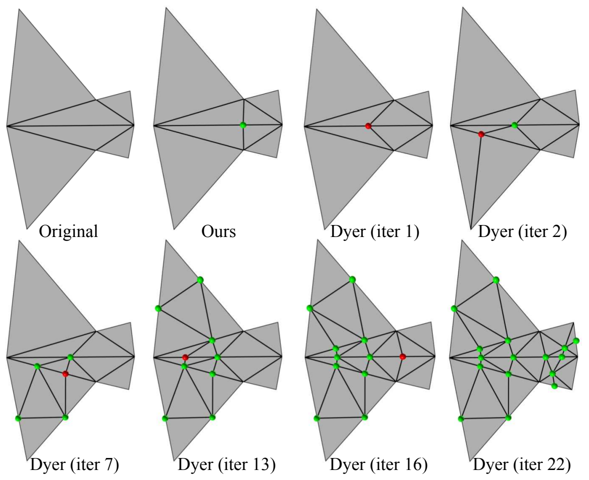


Figure 14: *Our geometry-aware edge refinement vs Dyer et al's combinatorial edge refinement. The bird-shaped non-planar patch is taken from a real-world model, where the horizontal edge is a non-flippable NLD edge. Our method splits the NLD edge at a single point so that all edges become Delaunay. As a combinatorial method, Dyer et al's edge refinement does not take the local geometry into consideration. Therefore, their method takes more iterations and produces a Delaunay mesh with 16 splitting points. The red point denotes the new splitting point at the current iteration of Dyer et al's algorithm.*

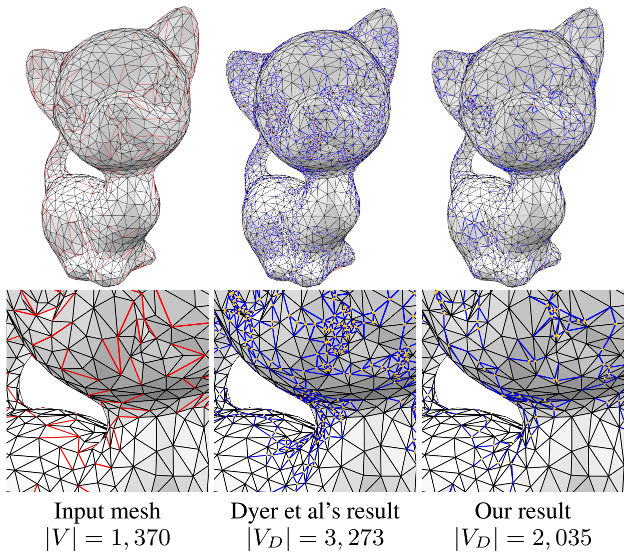


Figure 15: *Our method produces the DM with significantly less vertices than Dyer et al.'s method. The added auxiliary points are in yellow, the NLD edges in red and the added edges in blue.*

6.2 DM Simplification

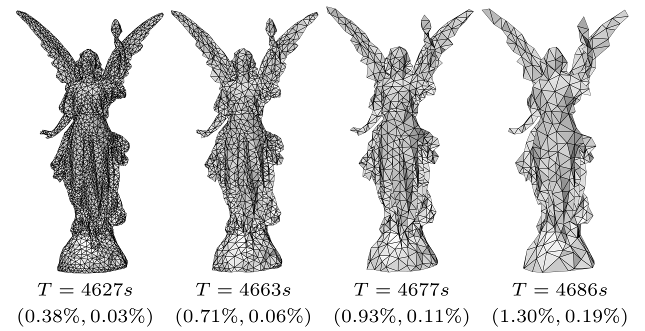
The proposed DM simplification algorithm is efficient, since the overhead per edge contraction is small. We observe that the speed of our method is comparable to the QEM method [Garland and Heckbert 1997]. Figure 12(b) shows the statistics of type-I and type-II vertices on common models. We observe that type-II removable vertices are dominant in the early stage of the simplification (i.e., when the resolution is still high), and type-I removable vertices become more important when the resolution is low.

Note that the DM simplification algorithm proposed in [Dyer et al. 2007] also adopts the QEM framework, which iteratively contracts mesh edges and ensures the Delaunay condition. Let $e = (v_i, v_j)$ be a to-be-contracted edge in a Delaunay mesh and $\overline{v}$ the target vertex after edge contraction. To ensure that the simplified mesh is still Delaunay, Dyer et al. showed that vertex $\overline{v}$ must be inside some allowed region $\Omega_{\overline{v}}$. However, computing the exact allowed region is challenging due to the lack of the analytical form. Dyer et al. [2007] suggested computing a subspace $\Omega'_{\overline{v}} \subset \Omega_{\overline{v}}$, represented by the intersection of half spaces. The optimal position of $\overline{v}$ is then obtained by minimizing the quadratic QEM objective function with linear inequality constraints. However, solving such an optimization problem is time consuming. For example, it takes their algorithm almost 80 minutes to simplify the 345K-vertex Lucy model to 500 vertices. In contrast, our method takes only 40 seconds. Computational results show that our method runs two orders of magnitudes faster than Dyer et al's method.

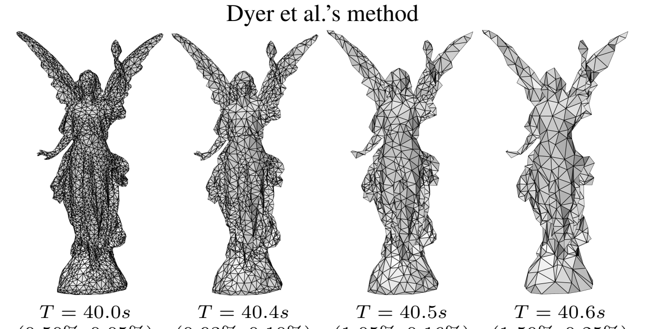
We adopt the Metro tool [Cignoni et al. 1998] to measure the difference between the original DMs and the simplified ones. Figure 13 plots the maximal error $\varepsilon_{\max}$ and the mean error $\bar{\varepsilon}$ in the simplification process. We can see that our simplification is slightly more accurate when the DM resolution is high, and Dyer et al's simplification is of better quality when the resolution is low. However, the visual difference between two methods is not significant at all regardless of DM resolution. See Figure 16.

6.3 Applications

Discrete Laplace-Beltrami operators (LBO) are ubiquitous in computer graphics, spanning from geometric modeling to simulation and imaging. Many applications require the discrete operators retain key structural properties inherent to the continuous setting. Un-


$T = 4627s$ $(0.38\%, 0.03\%)$ $T = 4663s$ $(0.71\%, 0.06\%)$ $T = 4677s$ $(0.93\%, 0.11\%)$ $T = 4686s$ $(1.30\%, 0.19\%)$

Dyer et al.'s method


$T = 40.0s$ $(0.50\%, 0.05\%)$ $T = 40.4s$ $(0.93\%, 0.10\%)$ $T = 40.5s$ $(1.05\%, 0.16\%)$ $T = 40.6s$ $(1.59\%, 0.25\%)$

Our method

Figure 16: *Comparison with Dyer et al's DM simplification algorithm. The original DM has $345K$ vertices and the simplified DMs have $5K$, $2K$, $1K$ and 500 vertices, respectively. T is running time and the tuple shows (the maximal error $\varepsilon_{\max}$, the mean error ε_{mean}). Our method maintains a comparable level of accuracy of Dyer's method, but runs two orders of magnitude faster than theirs.*

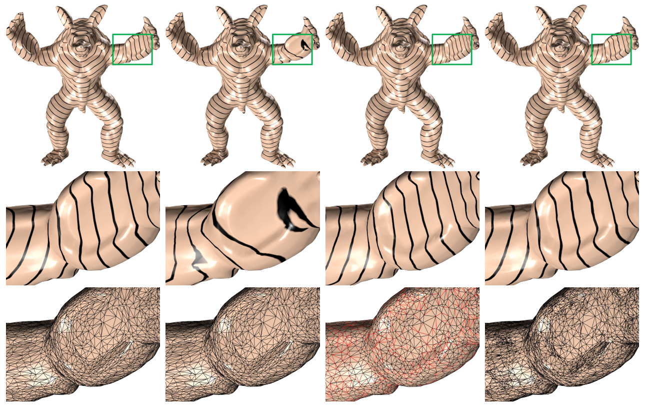

$|V| = 30,002$ 0.26s, $\epsilon = 0\%$ (a) Exact result
$|V| = 30,002$ 0.02s, $\epsilon = 2.76\%$ (b) HM on M
$|V| = 30,002$ 0.02s, $\epsilon = 0.81\%$ (c) HM on IDT
$|V_D| = 67,254$ 0.03s, $\epsilon = 0.69\%$ (d) HM on DM

Figure 17: *Solving the discrete geodesic problem. The MMP algorithm [Mitchell et al. 1987] computes exact geodesic distances in an empirical $O(n^{1.5}\log n)$ time, whereas the heat method (HM) [Crane et al. 2013] computes approximate distances in near-linear time. Since the HM solves a pair of elliptic linear system, its accuracy highly depends on the Delaunay quality of the input mesh. Both IDT and DM can improve the accuracy of the HM significantly. Note that the HM computes the first order approximation. The DM based result is slightly more accurate, since the average edge length of a DM is shorter than that of the IDT. Since the DM has more vertices than the IDT, it takes the heat method slightly longer time to compute geodesic distances. The reported runtime of the heat method does not include the preprocessing time.*

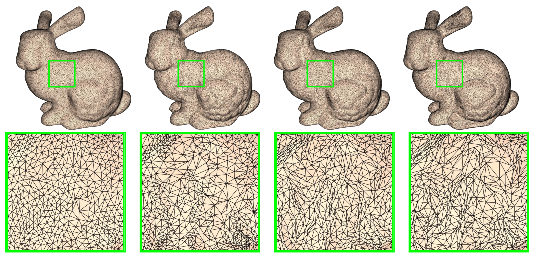
fortunately, as pointed out by Wardetzky et al. [2007], **discrete LBOs defined on *general* triangle meshes cannot satisfy all natural properties**, namely symmetry, non-negative weights, local support, linear precision, maximum principle and positive semi-definiteness. Wardetzky et al's theoretical analysis explains the diversity of existing discrete LBOs.

The cotan weights [Pinkall and Polthier 1993] and their variants, are arguably the most widely used discrete LBO in computer graphics. Given an edge $e_{ij} = \{v_i, v_j\}$, the cotan weight is defined as follows:

$$w_{ij} = \begin{cases} \frac{1}{2}(\cot\alpha_{ij} + \cot\alpha_{ji}) & \text{if } e_{ij} \text{ is an internal edge} \\ \frac{1}{2}\cot\alpha_{ij} & \text{otherwise} \end{cases}$$

where α_{ij} and α_{ji} are the two angles facing edge e_{ij}. This operator satisfies all the above-mentioned properties except non-negative weights. Since non-negative weights are a sufficient condition for discrete maximum principle, failing to respect it may result in unexpected numerical issues when solving discrete Laplacian and Poisson equations. To avoid such issues, many geometry processing algorithms require a preprocessing (e.g., remeshing) to improve mesh quality. Remeshing, however, is time consuming and in most cases, does change the geometry of the input model.

In this section, we show that Delaunay mesh is a natural solution to this problem. Recall that a DM is a manifold triangle mesh which has exactly the same geometry as the input mesh. Due to the empty geodesic circumcircle property, the sum of angles α_{ij} and α_{ji} subtended by an internal edge e_{ij} is no more than π. For a boundary edge, condition C2 in Theorem 4 in Supplementary Material also ensures the angle α_{ij} opposite to e_{ij} is no more than $\pi/2$. As a result, the cotan weight has guaranteed non-negative weights for all edges. In other words, **the DM induced cotan LBO satisfies all natural properties inherent to the continuous setting**. Note this result does not contradict to Wardetzky et al.'s claim, since a DM is **not** a general triangle mesh. Thanks to these favorable properties, many existing algorithms can benefit the numerical stability of DMs *without changing any codes*. We demonstrate DM on discrete geodesics and discrete harmonic maps.


$\tau = 1.336$ $\tau = 1.522$ $\tau = 1.728$ $\tau = 2.211$

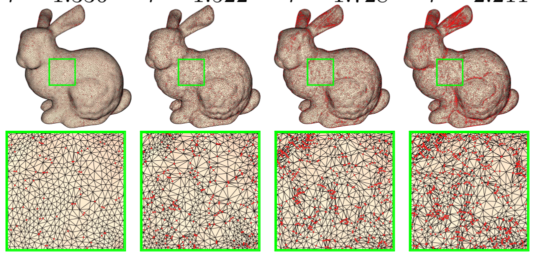

$|V_D| = 27,811$ $|V_D| = 31,659$ $|V_D| = 33,288$ $|V_D| = 47,538$

Figure 18: *The complexity of Delaunay meshes depends on the quality of the input triangulation. Row 1 shows the 25K-vertex Bunny in various tessellations and Row 3 shows the corresponding DMs. τ is the anisotropy measure: $\tau = 1$ for equilateral triangles; the larger τ, the higher degree of anisotropy of triangles.*

Discrete Geodesics. The heat method [Crane et al. 2013] is an elegant method for computing geodesic distances. Given a triangle mesh M and a user-specified point source $s \in M$, the heat method first solves the heat flow $u_s(t)$ for a fixed time t. Then it computes the normalized gradient field $X = -\frac{\nabla u}{\|\nabla u\|}$. Finally, it computes the geodesic distance field ϕ by solving the Poisson equation $\triangle\phi = \nabla \cdot X$. Since both the heat flow and the Poisson equation are elliptic linear systems, which can be prefactored once and subsequently solved in near-linear time, the heat method is highly efficient for large-scale models. However, its accuracy is highly sensitive to the Delaunay quality of the input mesh. As mentioned above, many real-world models are far from Delaunay triangulations, hereby the heat method yields poor approximation on such models. The proposed DMs can significantly improve the accuracy of the heat method. See Figure 17 and Table 2. More results can be found in Supplementary Material.

Model	$\|V\|$	$\|V_D\|$	Input mesh			DM			IDT		
			ϵ	T_p	T_s	ϵ	T_p	T_s	ϵ	T_p	T_s
Armadillo	30,002	67,254	2.76%	0.15	0.02	0.69%	0.26	0.03	0.81%	0.15	0.02
	172,974	293,126	0.40%	1.47	0.08	0.13%	2.70	0.13	0.29%	1.47	0.08
Bunny	25,007	41,172	1.87%	0.13	0.01	0.99%	0.23	0.01	1.04%	0.13	0.01
	72,020	144,233	1.32%	0.54	0.03	0.60%	1.31	0.06	0.65%	0.54	0.03
Gargoyle	35,001	60,047	1.73%	0.18	0.01	1.20%	0.34	0.03	1.58%	0.18	0.01
	349,999	514,208	0.43%	4.62	0.21	0.24%	7.16	0.31	0.36%	4.62	0.21
Lucy	20,002	48,077	4.43%	0.09	0.01	0.96%	0.24	0.02	1.38%	0.09	0.01
	262,909	424,954	1.95%	2.44	0.13	0.61%	4.33	0.13	1.02%	2.44	0.13

Table 2: *Computing geodesic distances using the heat method. $|V|$ and $|V_D|$ are the number of vertices in the input mesh M and the associated DM, respectively. The IDT has the same number of vertices as M. ϵ: relative mean error of the computed geodesic distances; The runtime, broken down into the preprocessing time T_p and solving geodesic distances T_s, was measured in seconds.*

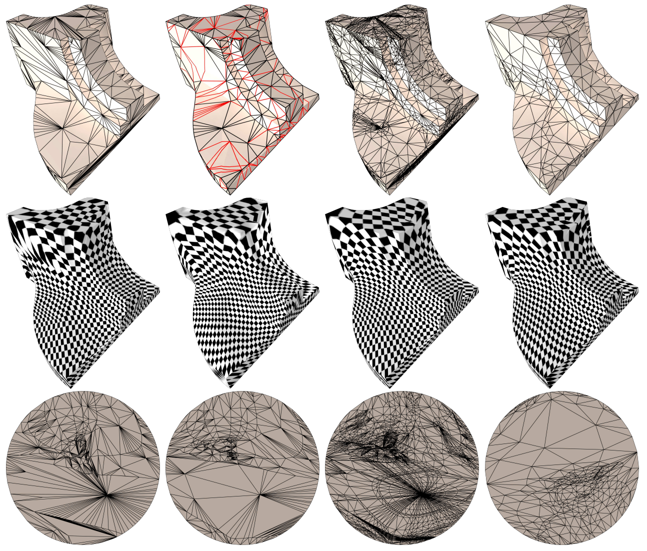
Discrete Harmonic Maps are a popular technique for surface parameterization. We parameterize the Fandisk model to a unit disk by solving a Laplace equation with Dirichlet boundary condition, where the boundary vertices are mapped to the unit circle by arc-length parameterization. We evaluate the parameterization quality by measuring the area and angle distortions. As Figure 19 shows, IDT, DM and simplified DM are very effective in terms of reducing angle distortions. Note that IDT edges are geodesic paths that may cross several triangles on the input mesh. As a result, IDT faces usually have significantly varying sizes, leading to even larger area distortion than the result obtained by the input mesh. The simplified Delaunay mesh (with exactly the same number of vertices as the input mesh) produces the least area distortion, since its triangle sizes are more uniform than those of the input mesh and DM. See Supplementary Material for more results.

7 Comparison

This section compares our method with other intrinsic Delaunay, Voronoi, and orthogonal dual structures. Table 3 summarizes the properties of various approaches.

7.1 Comparison with CDTs

A constrained Delaunay triangulation (CDT) [Chew 1987] is a generalization of the Delaunay triangulation that forces certain required segments into the triangulation. The CDT is useful for mesh generation since it respects user-specified constraints (such as boundaries and vertices) as well as having many of the properties of the Delaunay triangulation. Chew [1993] developed an elegant algorithm (also known as Chew's second algorithm) for creating quality CDTs on polyhedral surfaces embedded in $\mathbb{R}^3$. Chew's second algorithm



(a) Input mesh $|V| = 397$ $T = 0.001$s $(1.701, 2.848)$ (b) IDT $|V| = 397$ $T = 0.001$s $(1.565, 3.361)$ (c) DM $|V_D| = 3,633$ $T = 0.012$s $(1.353, 2.570)$ (d) Simplified DM $|V_D| = 397$ $T = 0.001$s $(1.463, 2.302)$

Figure 19: *Parameterizing the Fandisk model using harmonic map. The tuple shows the mean angle distortion and area distortion, respectively. The closer the value to one, the better quality the parameterization has. The DM induced parameterization has the least angle distortion, whereas the simplified DM (with the same number of vertices) produces the result with least area distortion.*

allows user to control triangle size so that small triangles in interesting areas and large triangles elsewhere. All triangles are guaranteed to have angles greater than 30 degrees, except for the badly shaped elements that may be required by the specified boundary.

The proposed DM distinguishes itself from the CDT in three aspects: Firstly, the CDT is well defined only if the given polyhedral surface is a smooth representation. Specifically, Chew's second algorithm assumes normals of adjacent vertices vary by less than $\frac{\pi}{2}$. Such an assumption often fails on models with large triangles and/or sharp features (e.g., the Fandisk model). As an intrinsic representation, our method works for arbitrary manifold triangle meshes. Secondly, the CDT is an approximate representation of the input mesh, whereas our DM has exactly the same geometry. Thirdly, Chew's second algorithm aims at creating high-quality triangle meshes, where all triangles (except the ones on the boundary) have a fairly good shape. In contrast, our method aims at fixing the non-Delaunay edges. As a result, the triangulation quality of a DM is not as good as that of a CDT. However, we do observe that our DM simplification algorithm can produce triangle meshes with much higher quality than the DMs. See Figure 16.

7.2 Comparison with IDTs

Both IDTs and DMs have exactly the same geometry of the input meshes and they satisfy the Delaunay condition everywhere. Each has its merits and limitations. An IDT has the same number of vertices, edges and faces as the input mesh. Since its edges are geodesic paths, the IDT is usually represented in an *abstract* manner, i.e., recording the length and connectivity of each Delaunay edge rather than its actual geometry. Obviously, such an abstract representation does not allow visualization. Even worse, the existing algorithms have to be adapted for IDTs.

As a manifold triangle mesh, the DM fits the existing graphics

Method	Arbitrary manifold mesh	Runtime performance	Numerical solver	Time complexity	Mesh complexity	Mesh representation	Strongly regular triangulation	Exact geometry
Constrained DTs	No	Fair	Yes	N.A.	User-specified	Manifold triangle mesh	Yes	No
CVTs	Yes	Fair	Yes	N.A.	User-specified	Manifold triangle mesh	Density dependent	No
Well-centered meshes	Yes	Fair	Yes	N.A.	n	Primal/Dual triangulation	Yes	No
Fisher et al's IDTs	Yes	Good	No	Unknown	n	Abstract	No	Yes
Liu et al's IDTs	Yes	Good	No	$O(n^2 \log n)$	n	Abstract	Yes	Yes
Dyer et al	Yes	Good	No	Unknown	Unknown	Manifold triangle mesh	Yes	Yes
Our method	Yes	Good	No	$O(nK \log K)$	$O(Kn)$	Manifold triangle mesh	Yes	Yes

Table 3: *Comparison of* ***intrinsic*** *Delaunay, Voronoi and orthogonal dual structures in terms of application domains, computational cost, space complexity, etc. A triangulation is called strongly regular if each triangle is incident with three different edges and three different vertices, the intersection of two triangles is either empty or one edge or one vertex, and the intersection of two edges is either empty or one vertex. If there are sufficient generators on the input model, the dual graph of the CVT is a strongly regular triangulation.*

pipeline, so that the existing algorithms can benefit from its favorable geometric and numerical properties without changing any codes. However, the price to pay for such a simple and versatile representation is the additional memory cost for storing auxiliary points. We prove DM has the $O(Kn)$ space complexity and observe that the number of splitting points is roughly twice the number of non-flippable NLD edges (see Table 1).

The algorithms for constructing IDTs and DMs are also different. The edge flipping algorithm [Fisher et al. 2007] for constructing IDTs is conceptually simple and easy to implement, however, it may produce self-loops and/or triangles with only two sides. Although it guarantees to stop in finite steps, the algorithm does not have a known time complexity. Compared with the existing IDT algorithms, our NLD edge refinement algorithm is easy to implement and it has a $O(nK \log K)$ time complexity.

7.3 Comparison with CVTs

Centroidal Voronoi tessellations are a powerful computational tool for generating highly regular tessellations in both Euclidean domains and curved domains. Since the CVT energy is highly nonlinear, both the Lloyd algorithm and the quasi-Newton solver compute only the local optimum. Although one can improve the results by some global optimization tools (e.g. [Lu et al. 2012]), the high computational cost diminishes its application to large-scale models. A CVT may not have a dual triangulation, especially when there are insufficient number of seeds on a high-genus model. For CVTs with dual triangulations, the dual mesh may have different geometry than the input model M, since the Delaunay edges are line segments, which usually do not stay on M. In contrast, every manifold triangle mesh has a DM with exactly the same geometry. Moreover, our algorithm is simple and efficient and it does not involve any numerical solver.

7.4 Comparison with Well-centered Meshes

Voronoi diagrams and Delaunay triangulations are orthogonal primal-dual structures. Such a dual structure can be defined in a more general setting, called weighted triangulation [de Goes et al. 2014; Glickenstein 2005]. Specifically, a weighted triangulation is a manifold triangle mesh equipped with a scalar weight per vertex. Well-centered meshes are a special type of weighted triangulation, where the weighted circumcenter of each triangle is in the triangle.

The proposed DMs have different application domains than weighted triangulations and are also built on a completely different foundation. First, the input mesh and its induced DM share exactly the same metric, whereas a weighted triangulation is based on a more general metric due to the more degrees of freedom brought by the weights. Therefore, weighted triangulations are desired to applications that require a solution space larger than the one that can be provided by the input mesh. Examples include self-supporting structures [de Goes et al. 2013; Liu et al. 2013], optimal transport [de Goes et al. 2012], etc. In contrast, DM is particularly useful for applications which require the original metric, such as computing geodesic distances and surface parameterization.

Second, the condition of well-centered meshes is much stronger than the local Delaunay condition. For example, all triangles must be acute in the case of uniform weights. Such a condition may not hold on general meshes. de Goes et al. [2014] proposed a novel algorithm for constructing well-centered meshes by optimizing both vertex weights and positions. Due to the change of vertex positions, the produced well-centered meshes have different shape than the input mesh. Using the local Delaunay condition (which is weaker than that of well-centered meshes), DM can be defined on arbitrary manifold triangle meshes without changing their geometries.

8 Conclusion & Future Work

This paper presents an efficient algorithm for constructing Delaunay meshes. Based on geometry-aware NLD edge refinement, our algorithm is conceptually simple and easy to implement. It also scales well due to the $O(nK \log K)$ time complexity. Following the powerful QEM framework, we also develop an efficient method for DM decimation. Due to its mesh-based representation and the empty circumcircle property, the DM-induced cotan Laplacian operator has guaranteed non-negative weights, making it an ideal input to the existing digital geometry processing algorithms. We observed that DM can improve the accuracy of the heat method for computing geodesic distances. Also, DM based discrete harmonic mapping is more robust than the one based on non-Delaunay mesh.

Within our current computational framework, the constructed DM has an $O(Kn)$ space complexity, where K is a model-dependent constant. Although K is fairly small for common 3D models, it is desired to develop a method which can produce DM with $O(n)$ vertices.

Acknowledgements

We thank the anonymous reviewers for their detailed and constructive comments. This project was partially supported by Natural Science Foundation of China (61322206, 61432003), 973 program (2011CB302202), TNList Foundation, AcRF RG40/12 and MOE2013-T2-2-011. Models are courtesy of Stanford University and AIM@SHAPE Shape Repository.

References

AMENTA, N., AND BERN, M. 1999. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry 22*, 4, 481–504.

BOBENKO, A. I., AND SPRINGBORN, B. A. 2007. A discrete laplace-beltrami operator for simplicial surfaces. *Discrete & Computational Geometry 38*, 4, 740–756.

BURAGO, Y. D., AND ZALLGALLER, V. A. 1960. Polyhedral embedding of a net. *Vestnik Leningrad. Univ. 15*, 66–80.

CHENG, S.-W., DEY, T. K., AND SHEWCHUK, J. R. 2012. *Delaunay Mesh Generation*. CRC Press.

CHEW, L. P. 1987. Constrained Delaunay triangulations. In *Proceedings of ACM SoCG '87*, 215–222.

CHEW, L. P. 1993. Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of ACM SoCG '93*, 274–280.

CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R. 1998. Metro: measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2, 167–174.

CRANE, K., WEISCHEDEL, C., AND WARDETZKY, M. 2013. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graph. 32*, 5, 152.

DE GOES, F., BREEDEN, K., OSTROMOUKHOV, V., AND DESBRUN, M. 2012. Blue noise through optimal transport. *ACM Trans. Graph. 31*, 6, 171:1–171:11.

DE GOES, F., ALLIEZ, P., OWHADI, H., AND DESBRUN, M. 2013. On the equilibrium of simplicial masonry structures. *ACM Trans. Graph. 32*, 4, 93:1–93:10.

DE GOES, F., MEMARI, P., MULLEN, P., AND DESBRUN, M. 2014. Weighted triangulations for geometry processing. *ACM Trans. Graph. 33*, 3, 28:1–28:13.

DEY, T. K., EDELSBRUNNER, H., GUHA, S., AND NEKHAYEV, D. V. 1999. Topology preserving edge contraction. *Publ. Inst. Math.(Beograd) (N.S) 66*, 80, 23–45.

DU, Q., AND WANG, D. 2005. Anisotropic centroidal Voronoi tessellations and their applications. *SIAM J. Scientific Computing 26*, 3, 737–761.

DYER, R., ZHANG, H., AND MÖLLER, T. 2007. Delaunay mesh construction. In *Proceedings of SGP '07*, 273–282.

DYER, R., ZHANG, H., AND MÖLLER, T. 2008. Surface sampling and the intrinsic Voronoi diagram. In *Proceedings of SGP'08*, 1393–1402.

EDELSBRUNNER, H., AND SHAH, N. R. 1997. Triangulating topological spaces. *International Journal of Computational Geometry and Applications 7*, 4, 365–378.

FISHER, M., SPRINGBORN, B., BOBENKO, A. I., AND SCHRÖDER, P. 2007. An algorithm for the construction of intrinsic Delaunay triangulations with applications to digital geometry processing. *Computing 82*, 2-3, 199–213.

GARLAND, M., AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. In *ACM SIGGRAPH '97*, 209–216.

GLICKENSTEIN, D. 2005. Geometric triangulations and discrete Laplacians on manifolds. *arXiv:math/0508188*.

GUIBAS, L. J., KNUTH, D. E., AND SHARIR, M. 1992. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica 7*, 1-6, 381–413.

INDERMITTE, C., LIEBLING, T., TROYANOV, M., AND CLÉMENÇON, H. 2001. Voronoi diagrams on piecewise flat surfaces and an application to biological growth. *Theoretical Computer Science 263*, 1-2.

LÉVY, B., AND LIU, Y. 2010. *Lp* centroidal Voronoi tessellation and its applications. *ACM Trans. Graph. 29*, 4, 119:1–119:11.

LIU, Y., WANG, W., LÉVY, B., SUN, F., YAN, D.-M., LU, L., AND YANG, C. 2009. On centroidal Voronoi tessellation - energy smoothness and fast computation. *ACM Trans. Graph. 28*, 4, 101:1–101:17.

LIU, Y.-J., CHEN, Z., AND TANG, K. 2011. Construction of iso-contours, bisectors, and Voronoi diagrams on triangulated surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 8, 1502–1517.

LIU, Y., PAN, H., SNYDER, J., WANG, W., AND GUO, B. 2013. Computing self-supporting surfaces by regular triangulation. *ACM Trans. Graph. 32*, 4, 92:1–92:10.

LLOYD, S. 1982. Least squares quantization in PCM. *IEEE Trans. Inform. Theory. 28*, 2, 129–137.

LU, L., SUN, F., PAN, H., AND WANG, W. 2012. Global optimization of centroidal Voronoi tessellation with Monte Carlo approach. *IEEE Transactions on Visualization and Computer Graphics 18*, 11, 1880–1890.

MAEHARA, H. 2011. On a proper acute triangulation of a polyhedral surface. *Discrete Mathematics 311*, 17, 1903–1909.

MULLEN, P., MEMARI, P., DE GOES, F., AND DESBRUN, M. 2011. HOT: Hodge-optimized triangulations. *ACM Trans. Graph. 30*, 4, 103:1–103:12.

OKABE, A., BOOTS, B., SUGIHARA, K., AND CHIU, S.-N. 2000. *Spatial Tessellations: Concept and Applications of Voronoi Diagrams*. Wiley.

PINKALL, U., AND POLTHIER, K. 1993. Computing discrete minimal surfaces and their conjugates. *Experiment. Math. 2*, 1, 15–36.

RIPPA, S. 1990. Minimal roughness property of the Delaunay triangulation. *Computer Aided Geometric Design 7*, 6, 489–497.

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

SARAF, S. 2009. Acute and nonobtuse triangulations of polyhedral surfaces. *European Journal of Combinatorics 30*, 4, 833–840.

VANDERZEE, E., HIRANI, A. N., GUOY, D., AND RAMOS, E. 2007. Well-centered planar triangulation-an iterative approach. In *Proceedings of IMR '07*, 121–138.

WARDETZKY, M., MATHUR, S., KÄLBERER, F., AND GRINSPUN, E. 2007. Discrete laplace operators: No free lunch. In *Proceedings of SGP '07*, 33–37.

YAN, D.-M., LÉVY, B., LIU, Y., SUN, F., AND WANG, W. 2009. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. In *Proceedings of SGP '09*, 1445–1454.

ZAMFIRESCU, T. 2002. Acute triangulations: a short survey. In *Proceedings of the Sixth Annual Conference of the Romanian Society of Mathematical Sciences*, 9–17.