Available online at www.sciencedirect.com
ScienceDirect
Computer Aided Design 00 (2014) 1–17
Journal Logo

Intrinsic Computation of Centroidal Voronoi Tessellation (CVT) on Meshes

Xiaoning Wang[a], Xiang Ying[a], Yong-Jin Liu[b], Shi-Qing Xin[c], Wenping Wang[d], Xianfeng Gu[e], Wolfgang Mueller-Wittig[f], Ying He[a,*]

[a]*School of Computer Engineering, Nanyang Technological University, Singapore*
[b]*Department of Computer Science and Technology, Tsinghua University, Beijing, China*
[c]*Faculty of Information Science and Engineering, Ningbo University, Zhejiang, China*
[d]*Department of Computer Science, Hong Kong University, Hong Kong, China*
[e]*Department of Computer Science, Stony Brook University, New York, USA*
[f]*Fraunhofer IDM@NTU, Nanyang Technological University, Singapore*

Abstract

Centroidal Voronoi tessellation (CVT) is a special type of Voronoi diagram such that the generating point of each Voronoi cell is also its center of mass. The CVT has broad applications in computer graphics, such as meshing, stippling, sampling, etc. The existing methods for computing CVTs on meshes either require a global parameterization or compute it in the restricted sense (that is, intersecting a 3D CVT with the surface). Therefore, these approaches often fail on models with complicated geometry and/or topology. This paper presents two intrinsic algorithms for computing CVT on triangle meshes. The first algorithm adopts the Lloyd framework, which iteratively moves the generator of each geodesic Voronoi diagram to its mass center. Based on the discrete exponential map, our method can efficiently compute the Riemannian center and the center of mass for any geodesic Voronoi diagram. The second algorithm uses the L-BFGS method to accelerate the intrinsic CVT computation. Thanks to the intrinsic feature, our methods are independent of the embedding space, and work well for models with arbitrary topology and complicated geometry, where the existing extrinsic approaches often fail. The promising experimental results show the advantages of our method.

Keywords: Voronoi diagram, centroidal Voronoi tessellation (CVT), discrete geodesics, exponential map, Riemannian center, center of mass, the Lloyd method, the L-BFGS method

1. Introduction

Centroidal Voronoi tessellation (CVT) is a special type of Voronoi diagram (VD) such that the generating point of each Voronoi cell is also its center of mass [1]. The CVT has broad applications in computer graphics, such as meshing, stippling, sampling, etc. Although the CVT in Euclidean space has been extensively studied, relatively little

*Corresponding author
Email addresses: WANG0684@e.ntu.edu.sg (Xiaoning Wang), yingxiang@ntu.edu.sg (Xiang Ying), liuyongjin@tsinghua.edu.cn (Yong-Jin Liu), xinshiqing@nbu.edu.cn (Shi-Qing Xin), wenping@cs.hku.hk (Wenping Wang), gu@cs.sunysb.edu (Xianfeng Gu), wolfgang.mueller-wittig@fraunhofer.sg (Wolfgang Mueller-Wittig), yhe@ntu.edu.sg (Ying He)

progress has been reported towards computing the CVT on curved surfaces. A key step in computing the CVT is to construct the Voronoi diagrams in each iteration. It is fairly simple to construct the Voronoi diagrams in Euclidean space (e.g. $\mathbb{R}^2$ and $\mathbb{R}^3$), since many efficient algorithms and software tools are readily available. However, it is technically challenging to compute VD on curved surfaces. Some researchers tackle this challenge by computing the restricted Voronoi diagrams [2], which is the intersection between the input mesh and the Voronoi diagram in $\mathbb{R}^3$. These approaches are embedding space dependent and may fail on models with complicated geometry and/or topology. Others adopt the global parametrization to map the surface to the parametric domain, such as Euclidean plane $\mathbb{E}^2$, the sphere $\mathbb{S}^2$, or hyperbolic disk $\mathbb{H}^2$, in which the 2-dimensional CVT is computed [3]. It is known that parameterizing models with complicated geometry and/or topology is computationally expensive and often suffers serious numerical issues. To our knowledge, there is no method for computing the CVT on *arbitrary* surfaces.

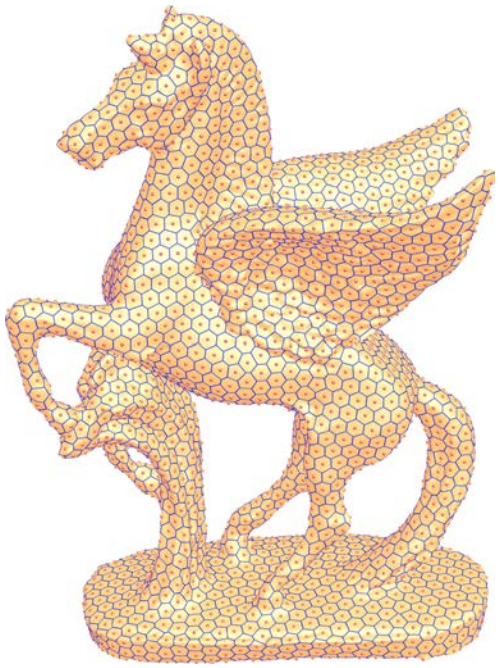


Figure 1. Our intrinsic method can compute a high-quality centroidal Voronoi tessellation on model with complicated geometry and topology. The CVT on the Pegaso model was created by 3000 sites.

To tackle the above-mentioned challenge, this paper presents two *intrinsic* algorithms for computing the centroidal Voronoi tessellation on arbitrary triangle meshes. Our first algorithm adopts the Lloyd framework, which iteratively moves the generator of each geodesic Voronoi diagram to its mass center. Based on the discrete exponential map, our method can efficiently compute the Riemannian center and the center of mass for any geodesic VD. Our second algorithm uses the L-BFGS method (limited-memory BFGS), which uses the CVT energy function and its gradients to approximate the Hessian matrix. The L-BFGS method has better performance due to its super-linear convergence rate. Thanks to the intrinsic feature, our methods are independent of the embedding space, and work well for models with arbitrary topology and complicated geometry, where the existing extrinsic approaches often fail. Figure 1 shows our result on the genus-5 Pegaso model. Moreover, our methods are insensitive to the mesh resolution and tessellation, and can be applied to surfaces embedded in arbitrary dimensional space. The promising experimental results demonstrate the efficacy of our methods.

The rest of the paper is organized as follows. Section 2 briefly reviews the existing work on CVT and the related topics. Then Section 3 and section 4 presents our intrinsic Lloyd and L-BFGS CVT algorithms in details. Section 5 shows the experimental results, compares our method to the existing techniques and discusses its advantages and limitations. Finally, Section 6 concludes the paper.

2. Existing Work

2.1. Voronoi Diagram & Centroidal Voronoi Tessellation in $\mathbb{R}^2$

Let $\mathbf{S}=(s_i)_{i=1}^m$ be a set of distinct sites in a connected compact region $\Omega \subset \mathbb{R}^2$. The Voronoi region Ω_i of s_i is defined as:

$$\Omega_i = \{x \in \Omega \mid ||x-s_i|| \leq ||x-s_j||, \forall i \neq j\},$$

where $\|\cdot\|$ denotes the Euclidean norm. The Voronoi regions, Ω_i, of all the sites form the Voronoi diagram of $\mathbf{S}$. The VD is a fundamental geometric tool that has a wide range of applications in science, engineering and even art. The classic algorithms for constructing Voronoi diagram in $\mathbb{R}^2$ are the sweep line algorithm [4] and the divide-and-conquer algorithm [5], which have optimal time complexity $O(m \log m)$.

Let the domain Ω be endowed with a density function $\rho(x) > 0$, which is assumed to be C^2. A typical energy function on Ω with regard to the Voronoi tessellation is defined as follows [1]:

$$F(\mathbf{S}) = \sum_{i=1}^{m} \int_{\Omega_i} \rho(x)||x-s_i||^2 d\sigma \triangleq \sum_{i=1}^{m} F_i,$$

where the term F_i expresses the compactness (or inertia momentum) of the Voronoi cell Ω_i associated with the site s_i. The Voronoi tessellation $\{\Omega_i\}$ is said to be a centroidal Voronoi tessellation if each site s_i coincides with the centroid c_i of its Voronoi cell, that is:

$$s_i = c_i (= \frac{\int_{\Omega_i} \rho(x) x d\sigma}{\int_{\Omega_i} \rho(x) d\sigma}).$$

The Lloyd algorithm [6] iteratively moves the generator of Voronoi cell to its mass center. Although it is conceptually simple and easy to implement, the Lloyd algorithm has only linear convergence rate. Liu et al. [7] proved that the CVT energy function is C^2 continuous. As a result, one can minimize the CVT energy by the Newton or quasi-Newton method, which converges much faster than the Lloyd algorithm.

2.2. Computing CVT on Surfaces

Both the Lloyd algorithm and the Newton algorithm require computing the Voronoi diagrams in each iteration. Although it is fairly simple to construct the VD in Euclidean space (e.g. $\mathbb{R}^2$ and $\mathbb{R}^3$), computing VD on curved surfaces is technically challenging. Alliez et al. [8] [9] conformally parameterized genus-0 open surface to a disk and evaluated the centroids over the density function expressed in parameter space rather than on the surface. Thanks to the angle-preserving and local isotropic properties of conformal parameterization, a well-shaped triangle in parameter space will not be deformed too much once lifted back into $\mathbb{R}^3$, except for its size, which can be easily compensated by the weighted density function in $\mathbb{R}^2$. Rong et al. [10] generalized the CVT energy function from $\mathbb{R}^2$ to spherical space $\mathbb{S}^2$ and hyperbolic space $\mathbb{H}^2$ and then combined all of them into a unified framework - the CVT in universal covering space (UCS). They adapted Lloyd's iteration to compute the CVT in the embedded fundamental domain of the UCS. If a centroid is outside of the fundamental domain by one side, they performed a rigid motion to move it to the oppositeside of the fundamental domain. The adjusted centroids are all inside the fundamental domain and are used as the new sites in the next iteration. Rong et al. [11] proposed a GPU-based method for computing the CVT on the plane and observed significant speedup of these GPU-based methods over their CPU counterparts. Their method also works for some 3D models that can be represented as a geometry image. However, as mentioned above, global parametrization is computationally expensive and may suffer from serious numerical issues if the surface has complicated geometry and non-trivial topology. For example, the Bunny's ears are shrunk to very tiny regions under spherical conformal parameterization [12], which poses a great numerical challenge to compute the CVT on $\mathbb{S}^2$.

Yan et al. [2] proposed a different approach. Rather than constructing the CVT on the mesh directly, they repeatedly computed restricted Voronoi diagrams (RVD), defined as the intersection between the input mesh and a Voronoi diagram in $\mathbb{R}^3$. Their method uses a kd-tree to quickly identify and compute the intersection of each triangle face with its incident Voronoi cells. The time complexity for computing RVD is $O(m \log n)$, where n is the number of seed points and m is the number of triangles of the input mesh. Their method also adopted the quasi-Newton method for fast convergence. They demonstrated that the restricted RVD-based method is flexible for computing the CVT with a non-constant density function. However, their method is embedding space dependent and may fail on models

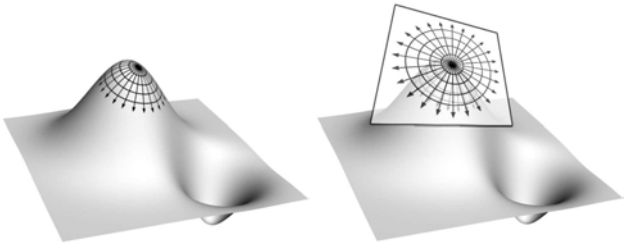


Figure 2. Exponential map naturally defines a geodesic polar coordinate system on curved surfaces.

with complicated geometry/topology. Recently, Lévy and Bonneel [13] proposed an elegant method for constructing curvature-adaptive anisotropic meshes. Their idea is to transform the 3D anisotropic space into a higher dimensional isotropic space, in which the mesh is optimized by computing a CVT. Lévy and Bonneel's method overcomes the d-factorial cost of computing a Voronoi diagram of dimension d by directly computing the restricted Voronoi cells with an algorithm called Voronoi Parallel Linear Enumeration, which can be easily parallelized. Their method is extrinsic due to the computation of intersection between the (higher dimensional) Voronoi cells and the surface.

2.3. Discrete Geodesics & Exponential Map

For any two points p and q on a 2-manifold surface, the geodesic path between p and q is a local shortest path connecting p and q on the surface [14]. If the surface is smooth, the geodesic is a curve on the surface whose geodesic curvature is always zero. Since geodesic curvature is only dependent on the first fundamental form, the geodesic is intrinsic to the surface. To compute the discrete geodesic on a triangle mesh of arbitrary topological type, two broad classes of methods exist. The first class treats the mesh as the first-order approximation of a smooth surface and uses numerical methods to solve a characterizing partial differential equation on the mesh; as a typical example, the Eikonal equation is solved on a mesh in [15]. The second class treats the mesh as a polygonal domain and uses computational geometry methods to build the shortest paths. The second class can compute the discrete geodesic exactly, and is typified by the Mitchell-Mount-Papadimitriou (MMP) algorithm [16], the Chen-Han (CH) algorithm [17] and their many variants[18][19][20][21][22]. The time complexities of the MMP and CH algorithms are $O(n^2 \log n)$ and $O(n^2)$, respectively, where n is the number of mesh vertices. Recently, Ying et al. [23] proposed the saddle vertex graph (SVG), which is a pre-computation technique for efficiently computing various types of discrete geodesics.

With the above-mentioned discrete geodesic algorithms, one can compute the exponential map, which defines a geodesic polar coordinate system on meshes (see Figure 2). Let $p \in M$ be an arbitrary point on a smooth surface M. The exponential map $\exp_p : T_pM \to M$ at p is a map from the tangent plane at p to p's local neighborhood. Given a radial line on the tangent plane which originates at p and has direction $\mathbf{v} \in T_pM$, the exponential map sends $p + t\mathbf{v}$ to a unique geodesic curve γ originating at p such that $\gamma'(0) = \mathbf{v}$ and $\|\gamma(t)\| = 1$. Conversely, given an arc-length parameterized geodesic γ originating at p, $\gamma(0) = p$, there is a unique tangent direction $\gamma'(0)$ on the tangent plane $\exp_p^{-1}(\gamma) = \gamma'(0)$. The exponential map has been used in various graphics applications, such as decal compositing [24], texture mapping [25], Poisson disk sampling [26], etc.

2.4. Geodesic Voronoi Diagram on Meshes

To compute the geodesic Voronoi diagram on a 2-manifold mesh M, the Euclidean norm is replaced by the geodesic metric. Such a metric change results in fundamental change in the bisector and Voronoi regions between the Euclidean plane and a curved 2-manifold. For example, a bisector in $\mathbb{R}^2$ is a line segment, while a bisector on a triangle mesh may contain both line segments and hyperbola segments. Given a genus-g mesh M, it was shown in [27] that the bisector of two distinct points on M can have at most $g+1$ separated components and a Voronoi region of a point in $\mathbf{S}$ can be bounded by one or more closed bisectors (see Figure 3). Liu [28] proved that the combinatorial complexity of geodesic Voronoi diagram on M is $O((m+g)n)$, where m is the number of points in $\mathbf{S}$ and n is the number of triangles in the mesh. Edelsbrunner and Shah [29] showed that for a general topological space, if a closed ball property is satisfied, then the dual Delaunay triangulation of Voronoi diagram exists. Recently, Liu et al. [30] showed that the intrinsic Delaunay triangulation on mesh M can be obtained by the duality of a geodesic Voronoi diagram

on M. They proved that this duality exists under two practical assumptions and proposed an efficient algorithm for constructing the Delaunay triangulation.

We refer the readers to [31] for in-depth discussion on the properties of Delaunay triangulation and Voronoi diagram on a 2-manifold mesh.

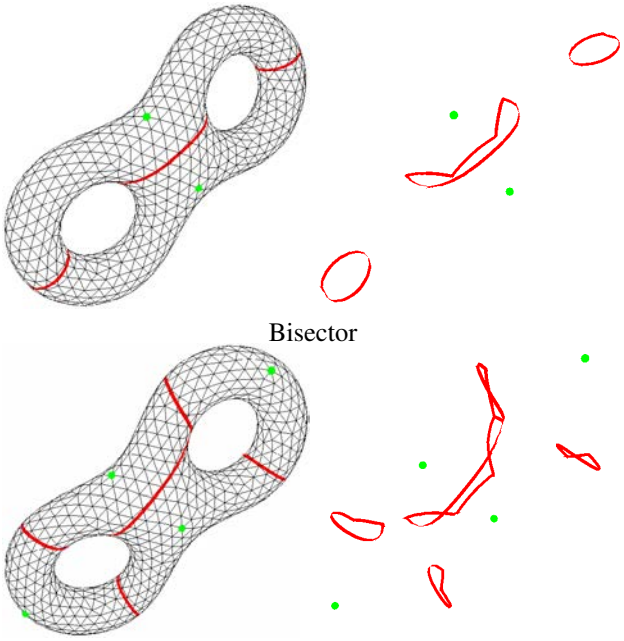


Figure 3. Bisector and geodesic Voronoi diagram. Row 1: The bisector (red) of two sites (green) on the double-torus has three separated components. Besides line segments, a bisector on triangle mesh may also contain hyperbola segments. Row 2: The geodesic Voronoi diagram of four sites. Each Voronoi cell is bounded by two or three closed bisectors.

3. The Lloyd Framework

3.1. Overview

Let $M = (V, E, F)$ be a triangle mesh representing a 2-manifold surface, where V, E and F are the set of vertices, edges and faces, respectively. Let $S = \{s_i | s_i \in M, i = 1, \cdots, m\}$ denote the set of sites on M.

Our algorithm adopts the Lloyd framework, which iteratively computes the geodesic CVT on meshes. For each iteration, we first compute the multiple-source-all-destination geodesic distance with the sites s_i, $i = 1, \cdots, m$, as the sources. This geodesic distance field on M induces a geodesic Voronoi diagram. Then for each geodesic Voronoi cell, say $\Omega_j \in M$, we compute its Riemannian center r_j, which is defined as the average of its corners. Next, we compute the exponential map $\exp(r_j)$ at the Riemannian center r_j, and map the Voronoi cell Ω_j to the tangent plane T_{r_j}, on which we can compute the center of mass c_j. Finally, we map the mass center from the tangent plane to the mesh using the exponential map $\exp(r_j)$. We update each site s_i to the new mass center and then repeat the above-mentioned procedure until the offsets of the sites are below the user-specified threshold.

3.2. Computing the Geodesic Voronoi Diagram

Taking $\{s_i\}_{i=1}^m$ as sources, we apply the ICH algorithm [21][1] to compute the multiple-source geodesic distance field. As a result, each mesh vertex is assigned a geodesic distance to its *closest* source. Then we label an edge LE

[1]The ICH algorithm in [21] computes the single-source geodesic distance. But it can be extended to multi-source case easily by adding a stoping criteria during the window propagation procedure: when two windows from different sources cover the same vertex, both windows stop propagation.

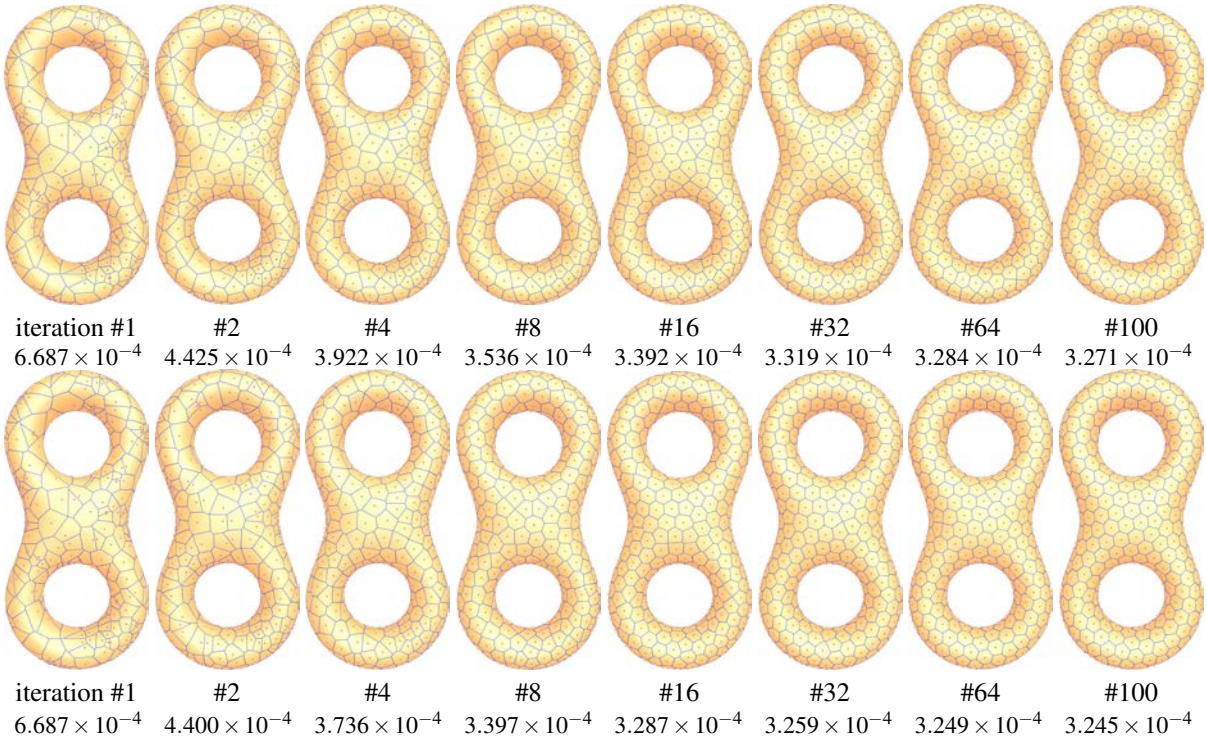
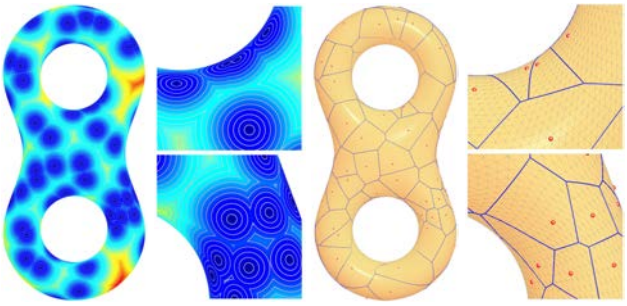


Figure 4. Iteratively computing geodesic CVT on meshes. Row 1: the Lloyd algorithm; Row 2: the L-BFGS algorithm.



(a) Geodesic distance field (b) Geodesic Voronoi diagram

Figure 5. The multiple-source geodesic distance field induces a geodesic Voronoi diagram. The cold (resp. warm) color in (a) indicates the small (resp. large) geodesic distance.

Algorithm 1 Intrinsic computation of centroidal Voronoi tessellation on meshes

Require: A triangle mesh $M = (V, E, F)$, the set of sites $S = \{s_i | s_i \in M, i = 1, \cdots, m\}$ and the convergence threshold ε;

Ensure: The centroidal Voronoi tessellation on M;

```
do
    Compute geodesic distance field with {s_i}_{i=1}^m as sources;
    Form the geodesic Voronoi diagram;
    for i = 1 to m do
        Compute the Riemannian center r_i for Voronoi cell Ω_i;
        Compute the center of mass c_i for Ω_i;
        d_i ← d(s_i, c_i);
        s_i ← c_i;
    end for
while (Σ_{i=1}^m d_i)/m > ε
```

if its two endpoints have different sources. Clearly, a *LE* edge is passed by a bisector. We further collect into a list *LT* all the triangles in M that are incident to any *LE* edge. As [27] shows, if a triangle $t \in LT$ has all its three edges labelled *LE*, then t contains a branch point in the geodesic Voronoi diagram; otherwise t is passed through by a single piece of a bisector. Based on the lists of *LE* and *LT*, we run the marching algorithm [27] for extracting the geodesic Voronoi diagram on M.

Assume the sites s_i are uniformly distributed. This assumption is reasonable, since the distribution of the sites is improved after only a few Lloyd iterations (see Figure 4). The ICH algorithm takes worst-case $O(\frac{n^2}{m}\log(\frac{n}{m}))$ time and empirical $O(\frac{n^2}{m})$ time, where n is the number of mesh vertices. The geodesic Voronoi diagram is then built in $O(k \log k)$ time, where k be the number of triangles in *LT*. Figure 5 shows the geodesic distance field and its induced geodesic Voronoi diagram on the double-torus model.

3.3. Computing the Riemannian Center

Let $v_1, v_2, \cdots, v_k$ be the corners of a Voronoi cell $\Omega_i \in M$. The Riemannian center is defined as the local minima x of the following function

$$U(x) = \sum_{i=1}^{k} d^2(x, v_i), \tag{1}$$

where $d(p, q)$ is the geodesic distance between p and q. If M has zero Gaussian curvature (that is, developable), the Riemannian center exists and is unique. However, in general, the function $U(x)$ is not convex, and the minimizer may not be unique. Kendall [32] and Karcher [33] showed the conditions to ensure the existence and uniqueness of the Riemannian center of mass. Intuitively speaking, if the points v_i are not too far from each other, there exists a *unique* Riemannian center of mass. Refer to [34][35] for the rigorous results.

Let $x^* \in M$ be the local minimal of Equation (1). Then x^* satisfies

$$\vec{0} = \nabla U(x^*) = \sum_{i=1}^{k} \nabla d^2(x^*, v_i) = 2 \sum_{i=1}^{k} d(x^*, v_i) \nabla d(x^*, v_i). \tag{2}$$

Since $d(,)$ is the geodesic distance, $\nabla d(x^*, v_i) \in T_{x^*}M$ is a unit tangent vector. Therefore, $d(x^*, v_i)\nabla d(x^*, v_i)$ represents a tangent vector with length $d(x^*, v_i)$, denoted by $\vec{t}_i$. Equation (2) requires $\sum_{i=1}^{k} \vec{t}_i = \vec{0}$, which means x^* is the center of the terminal points of $\vec{t}_i$.

We iteratively compute the local minimal x^*. Let x be the initial point, which could be either one of the corner points v_i or the site's location s_i. We compute the exponential map $\exp_x$ at x. The exponential map $\exp_x : T_xM \to M$ builds a geodesic polar coordinate system at x. The inverse map $\exp_x^{-1}$ maps the point $v_i \in M$ to the tangent plane T_xM. Let $\hat{x} \in T_xM$ be the average of the points $\exp_x^{-1}(v_1), \cdots, \exp_x^{-1}(v_k)$. If $\hat{x}$ does not equal x, we send $\hat{x}$ to the mesh M by the exponential map $\exp_x(\hat{x})$. Setting $x = \exp_x(\hat{x})$, we then repeat the above procedures until the average $\hat{x}$

Algorithm 2 Computing the Riemannian center

Require: A set of points $v_1, v_2, \cdots, v_k$ on M; the convergence threshold δ;
Ensure: The Riemannian center x;

1: $x \leftarrow v_1$;
2: **do**
3: $x_0 \leftarrow x$;
4: Compute the exponential map $\exp_x$ at x
5: The inverse map $\exp_x^{-1}$ brings the points v_i, $i = 1, \cdots, k$, back to the tangent plane T_xM;
6: $\hat{x} \leftarrow \frac{\sum_{i=1}^{k} \exp_x^{-1}(v_i)}{k}$;
7: $x \leftarrow \exp_x(\hat{x})$;
8: **while** $d(x, x_0) > \delta$

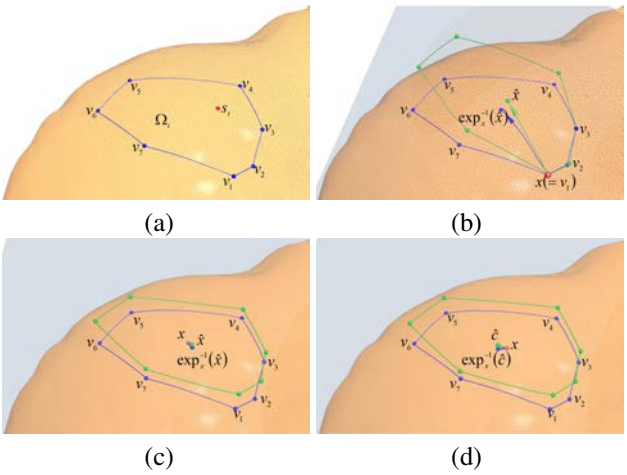


Figure 6. Computing the center of mass for the Voronoi cell Ω_i. It takes two iterations (b) and (c) to obtain the Riemannian center c. The blue dot $\exp_x^{-1}(\hat{c})$ in (d) is the center of mass.

agrees with x. Note that the exponential map, in general, does not preserve the area. However, when the Voronoi cells are small with respect to the injectivity radius[34], we observe that our algorithm can generate fairly good results. In our implementation, we set the initial point $x = s_i$, which is the center of the Voronoi region in the previous iteration. During the CVT iterations, the sites are getting closer to the Riemannian center, making finding the Riemannian center faster. We have observed that the iterative algorithm for finding Riemannian center converges very fast, took only two or three steps for all test models in our paper.

3.4. Computing the Center of Mass

Although the Riemannian center r is not the center of mass, it is *close* to all corners of the Voronoi cell. Therefore, it is very natural to use r to compute the center of mass. Let $\exp_r$ be the exponential map at the Riemannian center and $\hat{v}_i = \exp_r^{-1}(v_i)$ the pre-image of v_i, which lies on the tangent plane T_rM. Since the points $\hat{v}_i$, $i = 1, \cdots, k$, form a polygon on the tangent plane, its center of mass $\hat{c} \in T_rM$ is given by

$$x = \frac{1}{6A} \sum_{i=1}^{k-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

$$y = \frac{1}{6A} \sum_{i=1}^{k-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

where (x,y) are the coordinates of $\hat{c}$, (x_i,y_i) are the coordinates of $\hat{v}_i$, and A is the area of the polygon

$$A = \frac{1}{2}\sum_{i=1}^{k-1}(x_i y_{i+1} - x_{i+1}y_i)$$

Finally, the center of mass for Voronoi cell Ω is given by $c = \exp_r(\hat{c})$.

4. The L-BFGS Framework

Liu et al. [7] proved that the CVT energy function F has C^2 smoothness, thus, one can use the Newton or quasi-Newton method to optimize the energy F. In this Section, we adopt the L-BFGS method to accelerate the CVT computation. To compute the numerical integration on meshes, we modify the ICH algorithm [21] for computing the geodesic distance between any point (not necessarily a vertex) to the source point. We use [36] to compute numerical integration on each triangle. The detail of the modified ICH algorithm is in Section 5.1. The L-BFGS method requires the gradient of the energy function for approximating the approximated Hessian matrix. Given the energy function F in Section 2, the gradient of the CVT energy is [37][1]:

$$\frac{\partial F}{\partial x_i} = 2m_i(x_i - c_i),$$

where $m_i = \int_{\Omega_i}\rho(x)d\sigma$, c_i is the center of mass of the Voronoi cell Ω_i. We use the methods in Section 3.3 and Section 3.4 to compute c_i. Note that the seeds are restricted on the input mesh M, and the gradients are also constrained on the tangent space T_x. Using the exponential map $exp_x : T_xM \to M$, we can compute the projection $\hat{c}_i$ of c_i on T_x. During the L-BFGS optimization process, we use $2m_i(x_i - \hat{c}_i)$ as the gradient, so that it is on the tangent plane at point x_i. During each iteration in L-BFGS method, we get $\hat{x_i'}$ on T_x for each Voronoi cell, and use the inverse map exp_x^{-1} to get $x_i' = exp_x^{-1}(\hat{x_i'})$. Figure 7 shows the energy plot comparison between Lloyd method and L-BFGS method.

Algorithm 3 The L-BFGS algorithm for intrinsic CVT

Require: A triangle mesh $M = (V,E,F)$, the set of sites $S = \{s_i | s_i \in M, i = 1,\cdots,m\}$ and the convergence threshold ε;

Ensure: The centroidal Voronoi tessellation on M;

do
  Compute geodesic distance field with $\{s_i\}_{i=1}^m$ as sources;
  Form the geodesic Voronoi diagram;
  for $i = 1$ to m **do**
    Compute the Riemannian center r_i for Voronoi cell Ω_i;
    Compute the center of mass $\hat{c}_i$ on tangent plane T_i;
    Compute energy $F_i(x)$ and gradient $\frac{\partial F}{\partial x_i}$ for Ω_i;
  end for
  Using L-BFGS method to compute all seeds $\hat{s}_i$ on their tangent plane T_i ;
  Compute all updated seeds s_i' on Ω_i using exp map;
while $\|\nabla F(x)\| > \varepsilon$

5. Experimental Results

5.1. Implementation

The ICH algorithm [21] has linear space complexity and can compute the exact single-source geodesic distance in an $O(n^2 \log n)$ time (The empirical time complexity is $O(n^{1.5}\log n)$), where n is the number of vertices. The ICH algorithm computes the exact geodesic distances between any mesh vertex to the source vertex. However, it cannot compute the exact geodesic distance between any mesh points, i.e., non-vertex points on the mesh. We modify the

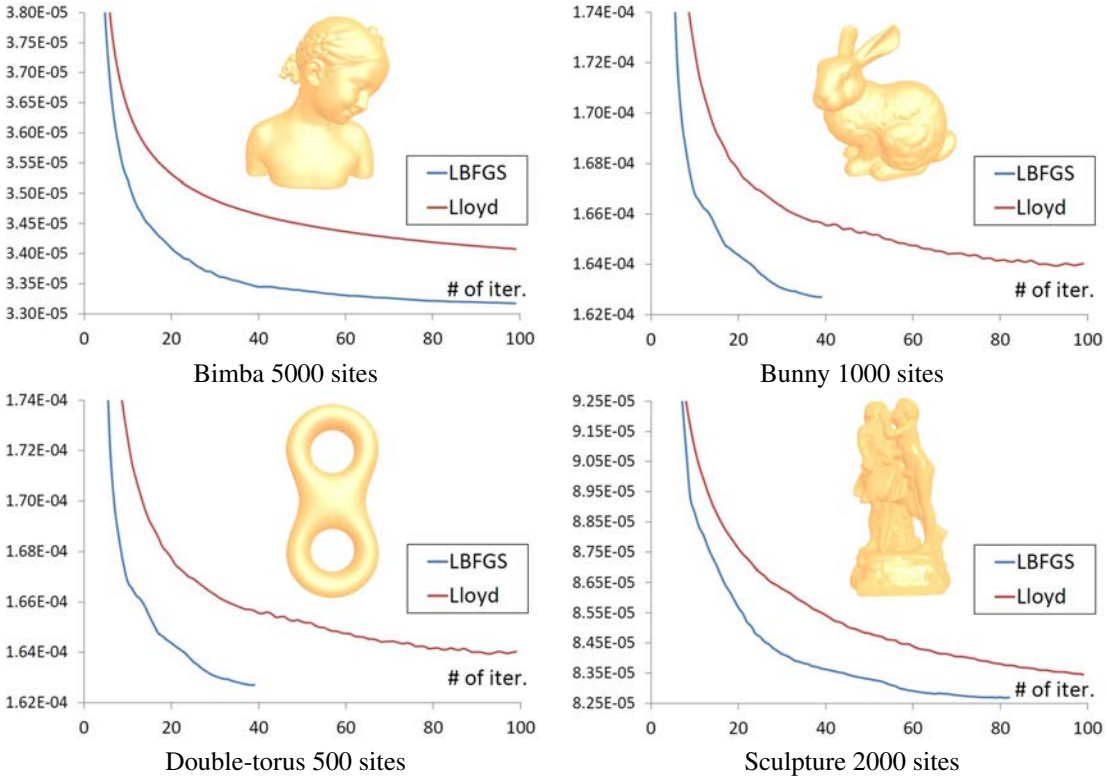


Figure 7. Convergence rate comparison between the Lloyd methd and the L-BFGS method.

ICH algorithm by sacrificing its space complexity: we store all the windows (a data structure that carries the geodesic distance from an edge interval to the source) generated in the window propagation procedure. When computing the distance from the source to a non-vertex point, say $p \in t$, which is inside a triangle t, we consider all windows covering t's sides, and find the one which can provide the shortest distance to p.

The modified ICH algorithm can also compute the discrete exponential map on triangle meshes. Thanks to the parallel structure of the Lloyd iteration, our algorithm can be easily implemented in parallel. Each ICH thread takes a point (not necessarily a mesh vertex) as the source, the ICH algorithm partitions each mesh edge into a set of intervals, called windows, which encode both the geodesic distance and the direction of the geodesic path emanating from the source. The windows are maintained in a priority queue according to the distance from the source and are propagated across the mesh faces: pops a window from the queue and then computes its children windows which can add, modify, or remove existing windows, and updates the queue accordingly. When a window reaches a vertex v, it updates v's distance and direction, which are used for the polar coordinates. The ICH algorithm terminates if the wavefront has reached the user-specified radius.

The input mesh is encoded in the half-edge structure and stored in the CPU's global memory in a read-only manner. Each CPU thread maintains its own data (i.e., the source point, the wavefront windows and the priority queue) in its own memory pool. Even though two or more ICH threads may compute on overlapped regions, they do not have any data and control conflicts, so each thread can proceed independently.

5.2. Results & Comparison

We adopted OpenMP to implement our method on an Intel 2.50 GHz CPU with four cores. Our program asks the user to specify the number of sites, then it generates the sites on the mesh randomly. We set the convergence threshold $\varepsilon = 10^{-6}$ in our implementation. Table 1 lists the model complexity and the performance of our algorithm and Figure 8 shows the computed CVT on some 3D models. Figure 9 shows CVT on high genus models.

Model	g	$\|V\|$	m	T	Q_{min}	Q_{avg}	θ_{min}	θ_{avg}
Armadillo	0	172,974	1,500	2.12	0.555	0.914	23.5	52.9
Bimba	0	74,764	1,500	0.91	0.639	0.926	35.4	53.9
Happy Buddha	6	488,217	1,500	8.27	0.456	0.901	21.5	51.4
Bunny	0	72,020	5,000	0.63	0.665	0.918	32.2	53.4
Double-torus	2	12,286	500	0.076	0.651	0.935	29.8	54.6
Dragon	0	422,558	1,500	6.18	0.445	0.901	21.9	51.9
Knotty-bottle	2	96,830	2,000	1.44	0.401	0.913	25.3	52.8
Pegaso	5	333,727	3,000	3.61	0.401	0.913	23.4	52.7
Sculpture	3	199,837	2,000	1.92	0.658	0.923	35.7	54.1
Spring	0	313,874	20,000	5.25	0.455	0.904	22.6	52.1

Table 1. Statistics of the mesh complexity and the timing. g: genus; $|V|$: the number of vertices; m: the number of sites; $\#iter$: total number of iterations; T: *average* time for each Lloyd iteration measured in seconds on an Intel 2.50GHz CPU with four cores. The last four columns are the quality measures for the dual Delaunay triangulation.

To evaluating the quality of our results, we compute the Delaunay triangulation, which is the dual graph of CVT. Then we adopt the following measures [38] [2]:

- Triangle quality: Let $Q(t) = 6S_t/(\sqrt{3}p_t h_t)$ be the quality of a triangle t, where p_t, S_t and h_t are the inradius, area, and the length of the longest edge of t, respectively. Let Q_{min} (resp. Q_{avg}) be the minimal (resp. average) quality measure. The closer the value to 1.0, the more isotropic of the Delaunay triangulation, therefore, the higher quality of the CVT one obtains.

- Minimal angle: Let θ_{min} be the minimal of the smallest angle of all triangles and θ_{avg} the average of minimal angles of all triangles. The closer the values of θ_{min} and θ_{avg} to 60 degrees, the more isotropic of the triangulation one obtains.

Figure 10 shows the quality improvement via the Lloyd iteration.

Compared to the parameterization-based methods [9], [10] [11], our method avoids the inaccuracy due to the approximation and metric distortion in parameterization. Furthermore, our method can apply to models of arbitrary geometry and topology, for which the parameterization is not easy to obtain. As Figure 12 shows, our method outperforms the UCS method [10] and the RVD method [2] in terms of quality (higher angle measure Q_{ave} and lower number of singularities).

The restricted Voronoi diagram methods [2] [39] approximate the CVT on surface by computing the intersection of a 3D CVT and the input mesh. Although it works fairly well for models with simple geometry, this approximation is extrinsic, that is, embedding space dependent. Figure 11 shows a Coil Spring model, where the coil almost touches itself and leaves very small gap. The RVD method cannot distinguish the geometrically-close-but-topologically-far pieces, and produces the wrong result. Our method is completely intrinsic in that all the computations are based on the metric only. So it can clearly distinguish these geometric "ambiguity". To further demonstrate the efficacy of our intrinsic method, we apply it to the Lion model in various poses. As Figure 13 shows, the computed CVTs are consistently among the near-isometric poses. Figure 14 shows the CVTs with very few sites. Since each Voronoi cell is big, we can clearly see the difference between the extrinsic RVD and our intrinsic CVT.

6. Conclusion

This paper presents an intrinsic algorithm for computing centroidal Voronoi tessellation on arbitrary triangle meshes. Our algorithm adopts the Lloyd framework, which iteratively moves the generator of each geodesic Voronoi diagram to its mass center. Based on the discrete exponential map, our method can efficiently compute the Riemannian center and the center of mass for any geodesic VD. Thanks to its intrinsic feature, our method works well for models with arbitrary topology and complicated geometry, where the existing extrinsic approaches often fail. The promising experimental results show the advantages of our method.

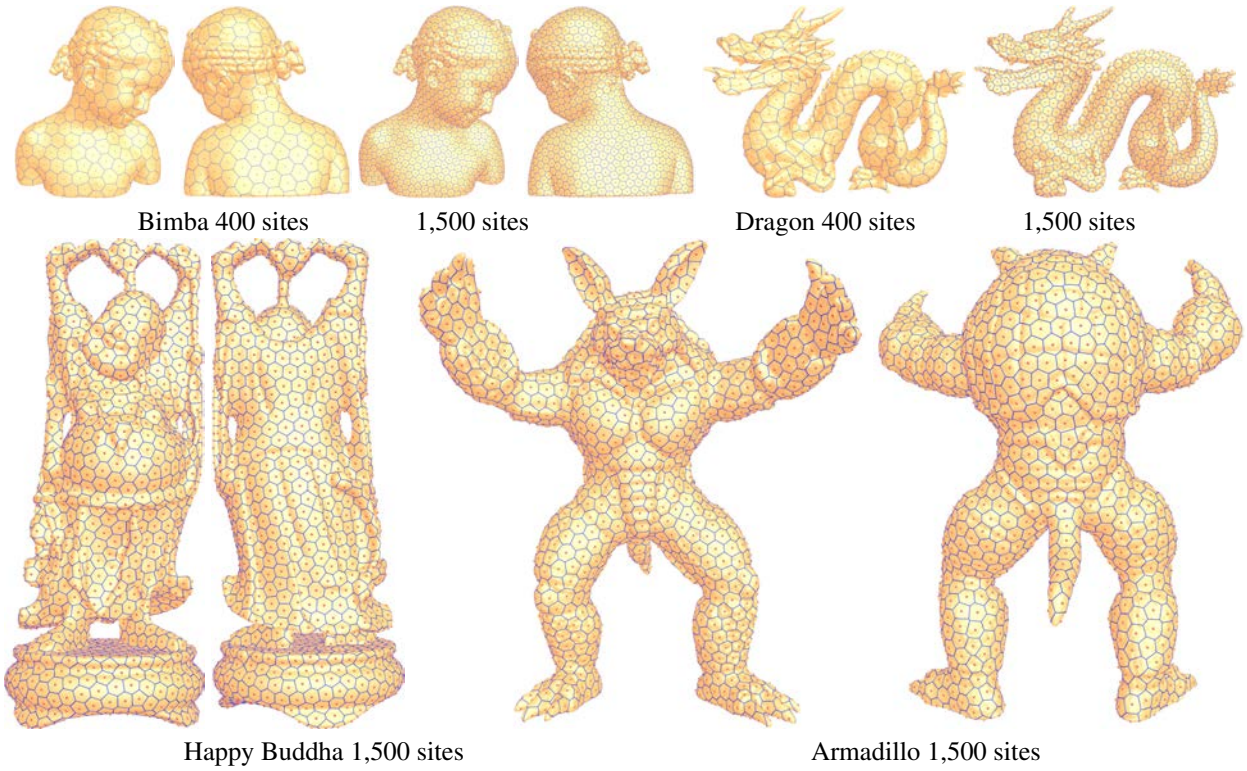


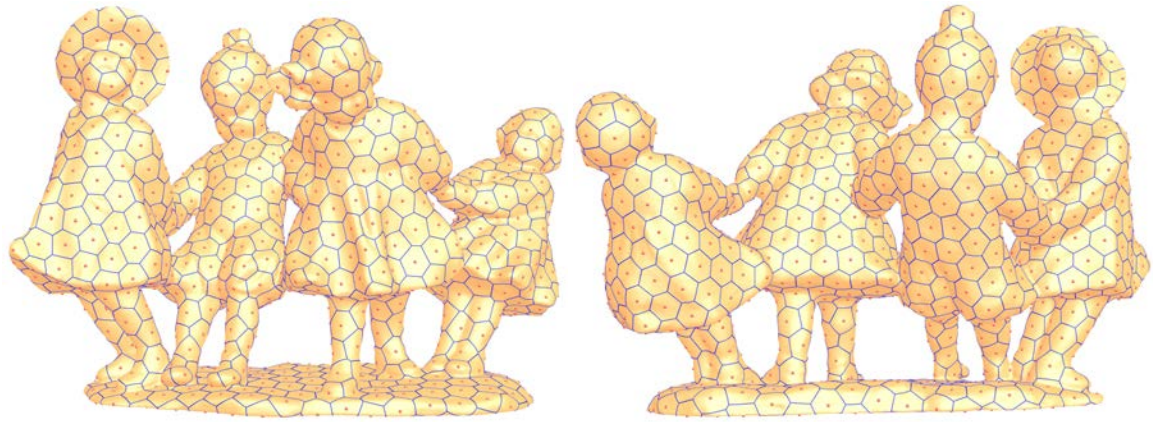
Figure 8. Experimental results. Images are rendered in high resolution that allows close-up examination.

Acknowledgement

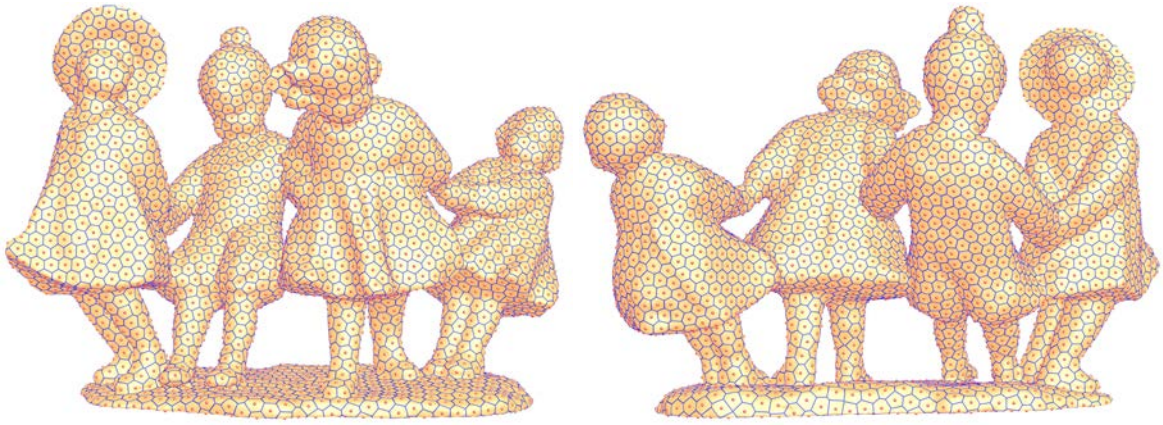
This research was done for Fraunhofer IDM@NTU, which is funded by the National Research Foundation (NRF) and managed through the multi-agency Interactive & Digital Media Programme Office (IDMPO) hosted by the Media Development Authority of Singapore (MDA). The NTU authors are supported by Singapore Ministry of Education (MOE) Grants RG40/12 and MOE2013-T2-2-011. Y.-J. Liu is partially supported by the Natural Science Foundation of China (61322206) and the National High Technology Research and Development Program of China (2012AA011801). X. Gu acknowledges the grants AFOSR FA9550-10-1-0294, NSF DMS-1221339, and Nets-1016829. We thank Prof. Xiaohu Guo for sharing his results, Dr. Yang Liu for his HLBFGS codes, and the anonymous reviewers for their valuable comments.

References

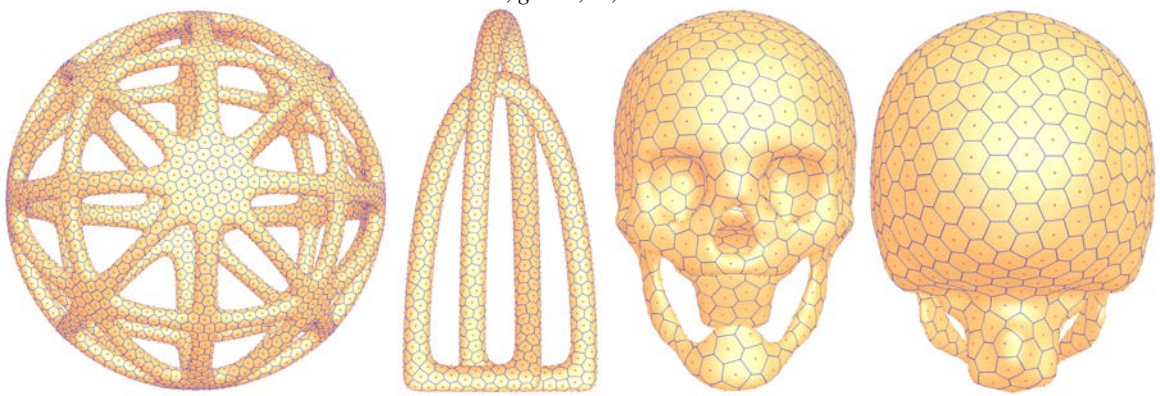
[1] Q. Du, V. Faber, M. Gunzburger, Centroidal voronoi tessellations: Applications and algorithms, SIAM Rev. 41 (4) (1999) 637–676.
[2] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, W. Wang, Isotropic remeshing with fast and exact computation of restricted Voronoi diagram, Comput. Graph. Forum 28 (5) (2009) 1445–1454.
[3] L. Shuai, X. Guo, M. Jin, Gpu-based computation of discrete periodic centroidal voronoi tessellation in hyperbolic space, Computer-Aided Design 45 (2) (2013) 463–472.
[4] S. Fortune, A sweepline algorithm for Voronoi diagrams, in: Proceedings of Symposium on Computational Geometry (SCG '86), 1986, pp. 313–322.
[5] M. Shamos, D. Hoey, Closest-point problems, in: Proc. 16th Annu. IEEE Sympos. Found. Comput. Sci., 1975, pp. 151–162.
[6] S. Lloyd, Least squares quantization in PCM, IEEE Transactions on Information Theory 28 (2) (1982) 129 – 137.
[7] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, C. Yang, On centroidal Voronoi tessellation - energy smoothness and fast computation, ACM Trans. Graph. 28 (4) (2009) 101:1–101:17.
[8] P. Alliez, É. C. de Verdière, O. Devillers, M. Isenburg, Isotropic surface remeshing, in: Shape Modeling International, 2003, pp. 49–58.
[9] P. Alliez, É. C. de Verdière, O. Devillers, M. Isenburg, Centroidal voronoi diagrams for isotropic surface remeshing, Graphical Models 67 (3) (2005) 204–231.



1,000 sites

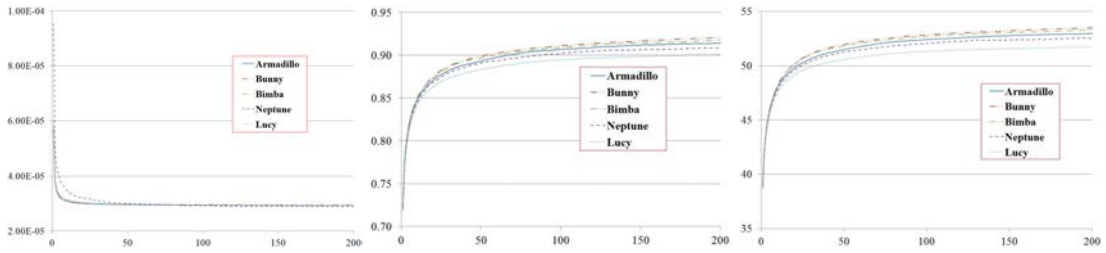


4-kid, $g = 8$, 5,000 sites



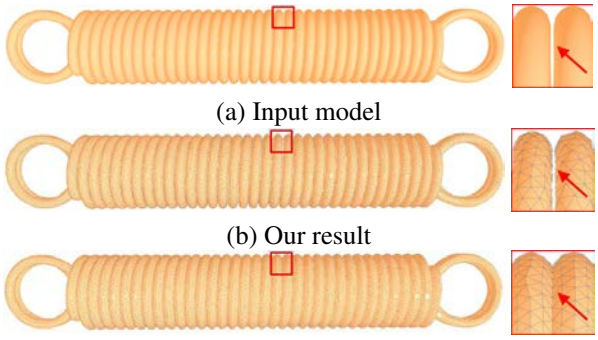
$g = 47$, 4000 sites Unity Knot, $g = 2$, 1,000 sites Skull, $g = 2$, 1,000 sites

Figure 9. Experimental results on high-genus models



(a) Normalized CVT energy function (b) Triangle quality Q_{avg} (c) Minimal angle θ_{avg}

Figure 10. Energy function and quality measures. The horizontal axis in the plots shows the iteration number. The vertical axis in (a) is the *normalized* CVT energy function, that is, $\frac{F(\mathbf{S})}{A^2}$, where A is the area of the model.



(a) Input model

(b) Our result

(c) [2]

Figure 11. Intrinsic vs. extrinsic. Consider the Coil Spring model, where the pitch of the helix equals the diameter of the coil. Therefore, the coil almost touches itself and leaves very small gap. See the closeup view. As an intrinsic method, our method is independent of the embedding space and it can correctly separate the coil. The extrinsic method [2] computes the CVT by intersecting a 3D CVT with the model, which cannot distinguish the two geometrically-close-but-topologically-separate pieces. The Delaunay triangulations, the dual of the computed CVTs, are shown in this figure.

[10] G. Rong, M. Jin, L. Shuai, X. Guo, Centroidal voronoi tessellation in universal covering space of manifold surfaces, Comput. Aided Geom. Des. 28 (8) (2011) 475–496.
[11] G. Rong, Y. Liu, W. Wang, X. Yin, X. Gu, X. Guo, GPU-assisted computation of centroidal Voronoi tessellation, IEEE Transactions on Visualization and Computer Graphics 17 (3) (2011) 345–356.
[12] X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, S.-T. Yau, Genus zero surface conformal mapping and its application to brain surface mapping, IEEE Trans. Med. Imaging 23 (8) (2004) 949–958.
[13] B. Lévy, N. Bonneel, Variational anisotropic surface meshing with Voronoi parallel linear enumeration, in: Proceedings of the 21st International Meshing Roundtable, 2013, pp. 349–366.
[14] M. P. do Carmo, Differential Geometry of Curves and Surfaces, Prentice-Hall, Inc., 1976.
[15] R. Kimmel, J. A. Sethian, Computing geodesic paths on manifolds, in: Proc. Natl. Acad. Sci. USA, 1998, pp. 8431–8435.
[16] J. S. B. Mitchell, D. M. Mount, C. H. Papadimitriou, The discrete geodesic problem, SIAM J. Comput. 16 (4) (1987) 647–668.
[17] J. Chen, Y. Han, Shortest paths on a polyhedron, in: SCG '90, 1990, pp. 360–369.
[18] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, H. Hoppe, Fast exact and approximate geodesics on meshes, ACM Trans. Graph. 24 (3) (2005) 553–560.
[19] Y.-J. Liu, Q.-Y. Zhou, S.-M. Hu, Handling degenerate cases in exact geodesic computation on triangle meshes, The Visual Computer 23 (9-11) (2007) 661–668.
[20] Y.-J. Liu, Exact geodesic metric in 2-manifold triangle meshes using edge-based data structures, Computer-Aided Design 45 (3) (2013) 695–704.
[21] S.-Q. Xin, G.-J. Wang, Improving Chen and Han's algorithm on the discrete geodesic problem, ACM Trans. Graph. 28 (4) (2009) 104:1–104:8.
[22] X. Ying, S.-Q. Xin, Y. He, Parallel Chen-Han (PCH) algorithm for discrete geodesics, ACM Transactions on Graphics 33 (1) (2014) 9:1–9:11.
[23] X. Ying, X. Wang, Y. He, Saddle vertex graph (SVG): a novel solution to the discrete geodesic problem, ACM Trans. Graph. 32 (6) (2013) 170:1–170:12.
[24] R. Schmidt, C. Grimm, B. Wyvill, Interactive decal compositing with discrete exponential maps, ACM Trans. Graph. 25 (3) (2006) 605–613.
[25] Q. Sun, L. Zhang, M. Zhang, X. Ying, S.-Q. Xin, J. Xia, Y. He, Texture brush: an interactive surface texturing interface, in: ACM Symposium on Interactive 3D Graphics and Games (I3D'13), 2013, pp. 153–160.
[26] X. Ying, S.-Q. Xin, Q. Sun, Y. He, An intrinsic algorithm for parallel Poisson disk sampling on arbitrary surfaces, IEEE Transactions on

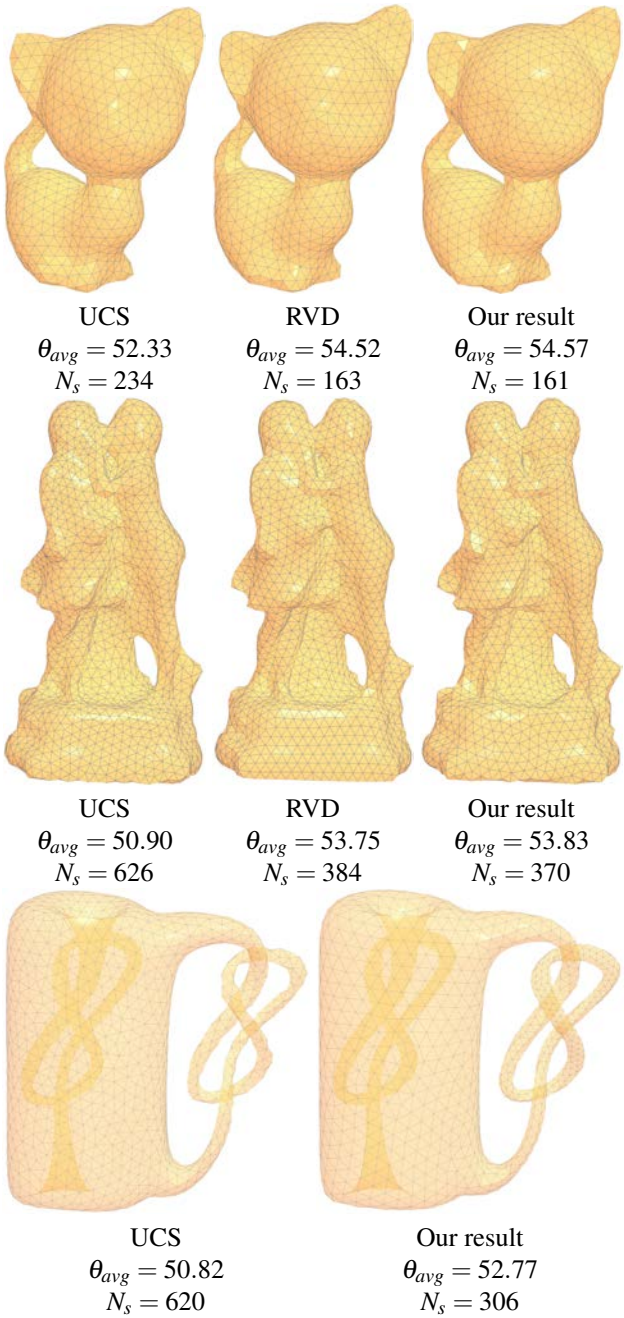
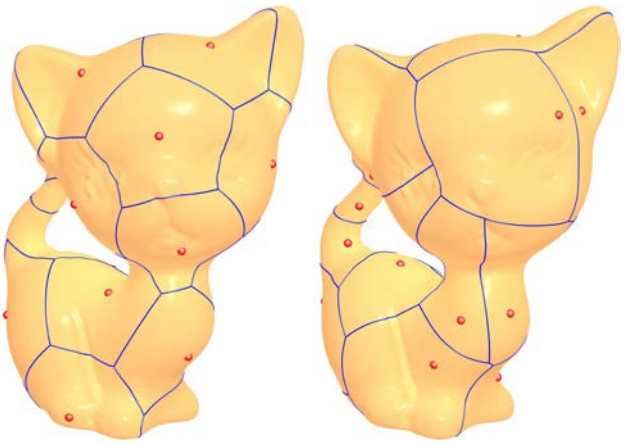


Figure 12. Comparison to the RVD method [39] and the UCS method [10]. N_s denotes the number of singularities (i.e., vertices whose valence is not six).

Figure 13. Thanks to its intrinsic property, our method can produce consistent results on the various poses of the Lion model.



Kitten, 20 sites

Our result RVD [2]

Figure 14. The intrinsic CVT and extrinsic RVD with very few sites.

Visualization and Computer Graphics 19 (9) (2013) 1425–1437.
[27] Y. Liu, Z. Chen, K. Tang, Construction of iso-contours, bisectors and voronoi diagrams on triangulated surfaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (8) (2011) 1502–1517.
[28] Y. Liu, K. Tang, The complexity of geodesic voronoi diagrams on triangulated 2-manifold surfaces, Information Processing Letters 113 (4) (2013) 132–136.
[29] H. Edelsbrunner, N. Shah, Triangulating topological spaces, International Journal of Computational Geometry and Applications 7 (4) (1997) 365–378.
[30] Y.-J. Liu, C.-X. Xu, Y. He, D.-S. Kim, The duality of geodesic Voronoi/Delaunay diagrams for an intrinsic discrete Laplace-Beltrami operator on simplicial surfaces, in: Proceedings of the 26th Canadian Conference on Computational Geometry (CCCG '14), 2014.
[31] R. Dyer, H. Zhang, T. Moller, Surface sampling and the intrinsic Voronoi diagram, in: Proceedings of the Symposium on Geometry Processing, 2008, pp. 1393–1402.
[32] W. Kendall., Probability, convexity, and harmonic maps with small image i: uniqueness and fine existence, Proc. London Math. Soc. 61 (2) (1990) 371?06.
[33] H. Karcher., Riemannian center of mass and mollifier smoothing, Comm. Pure Appl. Math 30 (1977) 509–541.
[34] X. Pennec, Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements, Journal of Mathematical Imaging and Vision 25 (1) (2006) 127–154.
[35] R. M. Rustamov, Barycentric coordinates on surfaces, Comput. Graph. Forum 29 (5) (2010) 1507–1516.
[36] A. Genz, R. Cools, An adaptive numerical cubature algorithm for simplices, ACM Trans. Math. Softw. 29 (3) (2003) 297–308. doi:10.1145/838250.838254.
URL http://doi.acm.org/10.1145/838250.838254
[37] M. Iri, K. Murota, T. Ohya, A fast voronoi-diagram algorithm with applications to geographical optimization problems, in: System Modelling and Optimization, Springer, 1984, pp. 273–288.
[38] P. Frey, H. Borouchaki, Surface mesh evaluation, in: 6th International Meshing Roundtable, 1997, pp. 363–374.
[39] D.-M. Yan, W. Wang, B. Lévy, Y. Liu, Efficient computation of 3d clipped Voronoi diagram, in: Geometric Modeling and Processing (GMP), 2010, pp. 269–282.