Eurographics Symposium on Geometry Processing 2009
Marc Alexa and Michael Kazhdan
(Guest Editors)

Volume 28 (2009), Number 5

Isotropic Remeshing with Fast and Exact Computation of Restricted Voronoi Diagram

Dong-Ming Yan[1,2], Bruno Lévy[2], Yang Liu[2], Feng Sun[1] and Wenping Wang[1]

[1]Department of Computer Science, The University of Hong Kong, Hong Kong, China
[2]Project Alice, INRIA, Nancy, France

Abstract

We propose a new isotropic remeshing method, based on Centroidal Voronoi Tessellation (CVT). *Constructing CVT requires to repeatedly compute* Restricted Voronoi Diagram (RVD)*, defined as the intersection between a 3D Voronoi diagram and an input mesh surface. Existing methods use some approximations of RVD. In this paper, we introduce an efficient algorithm that computes RVD exactly and robustly. As a consequence, we achieve better remeshing quality than approximation-based approaches, without sacrificing efficiency. Our method for RVD computation uses a simple procedure and a kd-tree to quickly identify and compute the intersection of each triangle face with its incident Voronoi cells. Its time complexity is $O(m \log n)$, where n is the number of seed points and m is the number of triangles of the input mesh. Fast convergence of CVT is achieved using a quasi-Newton method, which proved much faster than Lloyd's iteration. Examples are presented to demonstrate the better quality of remeshing results with our method than with the state-of-art approaches.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Triangle mesh surfaces are commonly used for shape representation in geometry modeling, physical simulation and scientific visualization. Raw meshes from 3D digital scanners, MRI or computer vision algorithms are often noisy and may contain many degenerate triangles. Thus they are difficult to be used for subsequent geometric processing. Various algorithms, based on mesh optimization [HDD*93], simplification [HG97] and remeshing [AUGA08] have been proposed to approximate a given raw mesh with another mesh of better quality.

Isotropic remeshing receives much attention in the current research on remeshing. An effective approach to isotropic remeshing is based on *Centroidal Voronoi Tessellation (CVT)* [DFG99]. Given an input mesh and a number of seed points, or *seeds*, on the mesh, a CVT energy function is minimized iteratively to yield an optimal distribution of the seeds, and the dual of the *Voronoi Diagram (VD)* of these seeds on the input mesh gives an isotropic triangle mesh.

The most difficult and time-consuming step in this approach is to compute a Voronoi diagram on a large and complex mesh surface at each iteration. A Voronoi diagram on a surface is naturally defined with geodesic distance, which results in a *Geodesic Voronoi Diagram (GVD)*. The computation of an exact GVD is difficult and existing approximate algorithms for GVD computation are time consuming [PC06]. By approximating the geodesic distance between two points with their Euclidean distance in 3D space, an approximation of GVD, called *Restricted Voronoi Diagram (RVD)* [ES97], can be used instead. However, fast and exact computation of RVD is also a non-trivial problem. To our knowledge, so far no solution has been proposed that is both efficient and exact. Existing algorithms compute different kinds of approximations to RVD, such as parameterization-based methods [ACDI05], discrete clustering techniques [VCP08] and boundary super-sampling [ACSYD05].

In this paper, we propose a fast and exact RVD computation algorithm. The key to the improved efficiency of our RVD computation is a new algorithm that computes the intersection of triangle faces and their incident Voronoi cells in a fast and robust manner. The input to RVD computation is the set of triangle faces of the input mesh and the set of the 3D Voronoi cells of the seed points; these seed points are the vertices of the output mesh. The task is to identify the Voronoi cells that overlap with each triangle face of the input mesh and compute their intersection.

We use a *kd*-tree to speed up the search of Voronoi cells

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.
Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

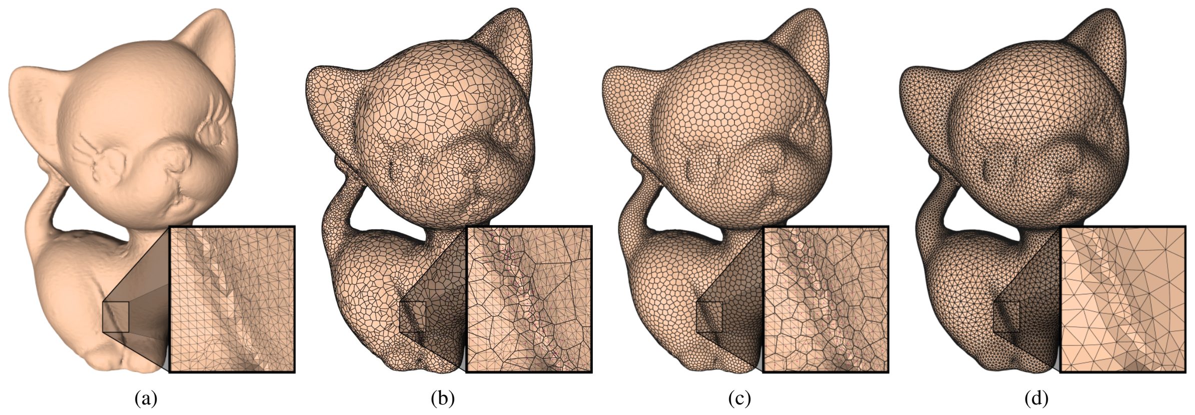


Figure 1: *Restricted Voronoi diagram computation and isotropic remeshing of the Kitten model (274k faces, 10k seeds). (a) Input mesh; (b) initial RVD; (c) optimized result (RCVT); (d) remeshing result.*

overlapping each triangle face, and use a novel symbolic polygon-clipping strategy to achieve simple and robust intersection computation. The complexity of the algorithm is $O(m \log n)$, where n is the number of seeds and m is the number of input triangles. Since we handle each triangle independently, our algorithm does not require any connectivity information of the input mesh. Our algorithm also scales up well with large meshes.

Contribution: To summarize, our main contribution is a practical remeshing framework based on existing theory (*constrained centroidal Voronoi tessellation, the topological ball property and* ε*-sampling*). The core of the framework is a new efficient and exact algorithm for computing restricted Voronoi diagrams. A key ingredient is our symbolic representation, used by both exact predicates and topological test. Our remeshing framework has the following advantages :

– *Simplicity*: The proposed cell-surface intersection algorithm is easy to implement since only plane-polygon clipping is required. There are less than 200 lines of code for the RVD computation functions;
– *Robustness*: Our framework uses exact geometric predicates, which ensures that the combinatorics of the RVD is correct. We propose a new symbolic clipping algorithm, that keeps track of relations of intersections with bisectors and initial vertices. It is used both to implement exact predicates and to ensure that the topology of the input surface is reproduced (The topological ball property [ES97]);
– *Efficiency*: Our method is fast. It computes the RVD of a very large mesh within a reasonable time, e.g., it takes only 2.1 seconds to compute RVD on the Kitten model (274k faces) with 10k seeds (see Figure 1). The complete remeshing process takes 228 seconds (67 iterations of RVD computation and Delaunay triangulation).

As an application, we present a new, efficient CVT based isotropic remeshing framework built on top of our exact RVD computation technique. The features and boundaries are preserved in a unified framework. The topological correctness of the output mesh is checked consistently during the optimization stage. The main advantage is the very well-shaped triangles that we obtain, the smallest angle is around 30°, compared with 2° obtained with previous works (see the section on results). The smallest angle is vital to numerical computations applied to the mesh, since this determines the conditioning of the FEM matrices [She02]. Figure 1 shows an example of RVDs and the remesh obtained by our method.

1.1. Previous work

An exhaustive survey of remeshing techniques is out of the scope of this paper. The reader is referred to [AUGA08] for the survey of remeshing techniques, and the references therein. We shall only review the type of methods based on CVT [DFG99], where RVD computation on a 3D mesh surface is the key problem. We will also mention *Delaunay Refinement* (see Section 2.3). We refer the reader to [Aur91, OBSC00] for fundamental notions and many previous works on Voronoi diagram and Delaunay triangulation.

From an intrinsic point of view, the Voronoi diagram on a surface is defined using geodesic distance, resulting in geodesic Voronoi diagram (GVD). Kunze *et al.* [KWR97] compute GVD on parametric surfaces. Peyré and Cohen [PC06] propose an approximation of GVD on a mesh surface using a discrete approximation of geodesic distance. However, exact GVD computation is still a difficult problem and the existing approximation algorithms are time consuming, which are difficult to be used in realtime applications.

Mesh parameterization [AMD02, ACDI05] is another technique for computing the CVT for mesh surfaces. The input mesh is first parameterized onto a 2D domain. Then the CVT is computed on the 2D parameter domain and the result is lifted back to the 3D surface. A major drawback with the parameterization-based remeshing approach is the need for complex strategy to "stitch" parameterized charts together for a surface of high genus. Surazhsky *et al.* [SAG03] present a local parameterization-based remeshing method that extends the work of [ACDI05] to deal with meshes with arbitrary genus. However, these methods still suffer from inaccuracies caused by the distortions of the parametrization and instabilities with poorly shaped triangles.

For tetrahedral meshing, Alliez *et al.* [ACSYD05] use dense sample points to approximate a mesh surface and compute a discrete RVD. Valette *et al.* [VCP08] compute an approximated RVD by clustering mesh triangles or vertices. These methods are fast but the remeshing result may be poor when the input mesh contains degenerate triangles.

2. Background

We will briefly introduce the preliminaries of the CVT-based remeshing framework in this section.

2.1. Centroidal Voronoi tessellation

Given n distinct seed points (or *seeds*) $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ in $\mathbb{R}^N$, the Voronoi Diagram of $\mathbf{X}$ in $\mathbb{R}^N$ is defined as n Voronoi cells $\mathbf{C} = \{\Omega_i\}_{i=1}^n$, where

$$\Omega_i = \{\mathbf{x} \in \mathbb{R}^N \mid \|\mathbf{x}-\mathbf{x}_i\| \leq \|\mathbf{x}-\mathbf{x}_j\|, \forall j \neq i\}.$$

The Voronoi cell Ω_i is the intersection of a set of N-dimensional half-spaces, delimited by *oriented* planes $\mathbf{Q}_i = \{P_k^i\}_{k=1}^{n_i}$, which are the bisecting planes of the Delaunay edges incident to the seed $\mathbf{x}_i$.

Centroidal Voronoi tessellation, or CVT for short, is a special kind of Voronoi tessellation such that each seed $\mathbf{x}_i$ coincides with the mass center $\mathbf{x}_i^*$ of its Voronoi region Ω_i, defined as :

$$\mathbf{x}_i^* = \frac{\int_{\Omega_i} \rho(\mathbf{x})\mathbf{x}\,d\sigma}{\int_{\Omega_i} \rho(\mathbf{x})\,d\sigma}, \tag{1}$$

where $\rho(\mathbf{x}) > 0$ is a user-defined density function. When ρ is constant, we get a uniform CVT.

Equivalently, CVT can be defined [DFG99] as a critical point of :

$$F(\mathbf{X}) = \sum_{i=1}^{n} \int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x}-\mathbf{x}_i\|^2 \, d\sigma. \tag{2}$$

2.2. Constrained CVT and Restricted CVT

For a compact surface $\mathbf{S} \subset \mathbb{R}^3$ and a set of seeds $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^3$, the restricted Voronoi diagram (RVD) on $\mathbf{S}$ is denoted by $\mathcal{R} = \{\mathbf{R}_i\}_{i=1}^n$, where $\mathbf{R}_i$ is the restriction of the Voronoi cell Ω_i of $\mathbf{x}_i$ on $\mathbf{S}$, that is, $\mathbf{R}_i = \Omega_i \cap \mathbf{S}$ [ES97]. We call $\mathbf{R}_i$ a *restricted Voronoi cell (RVC)* (see Figure 2).

As an extension of CVT, Du *et al.* [DGJ03] introduce the *constrained CVT* (CCVT) on a surface, as a critical point of :

$$F(\mathbf{X}) = \sum_{i=1}^{n} \int_{\mathbf{R}_i} \rho(\mathbf{x}) \|\mathbf{x}-\mathbf{x}_i\|^2 \, d\sigma, \tag{3}$$

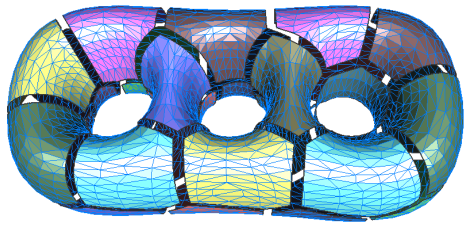


Figure 2: *Restricted Voronoi Cells (shrunk).*

with the seeds $\mathbf{X}$ being constrained on $\mathbf{S}$. For a CCVT, the seeds $\mathbf{x}_i \in \mathbf{S}$ are the constrained centroids of $\mathbf{R}_i$ given by

$$\mathbf{x}_i^* = arg \min_{\mathbf{y}\in\mathbf{S}} \int_{\mathbf{x}\in\mathbf{R}_i} \rho(\mathbf{x}) \|\mathbf{y}-\mathbf{x}\|^2 \, d\sigma. \tag{4}$$

In practice we want to compute a CVT given by a minimizer of this function instead of merely a critical point, which may be a saddle point.

If we minimize the same energy function as in Equation (3), but without constraining the seeds $\mathbf{X}$, then we obtain the so called *Restricted CVT*, or *RCVT*, because the Voronoi domains are still restricted to be $\mathbf{R}_i$ on $\mathbf{S}$. Both CCVT and RCVT can be used to obtain an optimal tessellation of the surface $\mathbf{S}$, from which an isotropic dual triangle mesh can be extracted. While CCVT is more suitable for a smooth surface, RCVT is faster and more robust for remeshing a noisy, irregular mesh surface.

Recently, Liu *et al.* [LWL*09] show that the CVT energy function has C^2 smoothness, and therefore they implement a quasi-Newton method, called *limited-memory BFGS method*, or *L-BFGS method* [LN89], to minimize the CVT function. They show that it converges much faster than Lloyd's method [Llo82]. We adopt this optimization method to compute CCVT and RCVT. In our specific case, the key to its successful application is the computation of the RVD, which requires to compute the intersection between Voronoi cells and triangle faces of the input mesh in each iteration of the optimization. The main contribution of this paper is to provide a simple and efficient solution to this problem, thus making it practical to use CVT to obtain high-quality remeshing of complex mesh surfaces. We show later a new framework to efficiently compute both CCVT and RCVT.

2.3. Restricted Delaunay Triangulation, Validity

Once the CCVT or RCVT is computed, the triangulation can be extracted, by considering the *Restricted Delaunay Triangulation (RDT)*, that is to say the dual of the RVD. The RDT is a simplicial complex with one vertex associated to each cell of the RVD, one edge associated to each couple of RVD cells that share an edge, and one triangle associated to each triple of RVD cells that share a vertex [ES97]. We need to check whether the RDT is valid. By being valid, we mean that the RDT needs to be homeomorphic to the initial surface $\mathbf{S}$. Two theorems give sufficient conditions. The first one characterizes the remeshing from a combinatorial point of view, and the second one characterizes the density of the sampling needed to capture the topology of **S**.

Theorem 1: *Topological Ball Property* [ES97]. If the intersection between each k-dimensional Voronoi face and **S** is a $k-1$ topological ball (Topological Ball Property), then the Restricted Delaunay Triangulation is homeomorphic to **S** .

Theorem 2: *ε-sampling* [AB99]. If for each point **p** of the surface **S** there is a sample of **X** at a distance of **p** smaller than $\varepsilon \times lfs(\mathbf{p})$, where $\varepsilon < 0.3$ and where lfs denotes the local feature size (i.e. distance to the medial axis), then **X** has the topological ball property, and therefore the restricted Delaunay triangulation of **X** is homeomorphic to **S**.

The notion of ε-sampling was later generalized to a wider class of objects, i.e. piecewise-smooth surfaces [BO06], by defining a generalization of lfs (Lipschitz radius).

In the context of surface reconstruction, the surface is unknown, and the conditions of Theorem 1 cannot be checked directly. For this reason, Theorem 2 is widely used by these methods [ABK98]. In contrast, in remeshing, the surface is known, allowing to check the precondition of both theorems. As such, Delaunay refinement methods [CDR07] use a combination of topological tests (Theorem 1) and geometric tests (Theorem 2) to iteratively insert vertices in the zones that do not meet the requirements to ensure a homeomorphic reconstruction. In [CDL07], a practical algorithm is proposed, that only requires to compute intersection between Voronoi edges and the surface.

In this paper, we are interested in the so-called *variational approaches* [ACSYD05], where 3D Lloyd relaxation (or a variant) is applied with a density function ρ computed from an approximation of lfs. In our context of surface remeshing, this idea naturally leads to define the density function $\rho(\mathbf{p}) = 1/lfs(\mathbf{p})^2$, to make the density of seeds match requirements of Theorem 2. However, in the variational setting, this strategy is in practice not sufficient for ensuring the validity of the result for the following two reasons. First, the density function uses an approximation of lfs, that may be not accurate enough, and second, more importantly, weighting Lloyd's energy with the density ρ tends to meet the ε-sampling criterion but does not ensure that it is satisfied. However, in our case, since we exactly compute the Restricted Voronoi Diagram, the topology of the restricted Voronoi cells is known exactly, thus we can use Theorem 1 to detect incorrect cells and iteratively insert new vertices where needed (see Section 4.4 further).

2.4. Algorithm overview

We propose a complete remeshing framework that uses the L-BFGS method [LWL*09] to minimize the CVT energy function. Since in CCVT the seeds need to be constrained on the surface **S**, directly minimizing the CCVT energy is a difficult problem, especially for noisy models or irregular meshes output from computer vision algorithms. Furthermore, to satisfy the seed constraint, in each iteration of optimization extra computation is needed to project updated seeds back onto the surface **S**. For these reasons, we start by computing an RCVT on **S** to provide a very good initialization that matches the overall desired vertex distribution of CCVT. Then, as explained in Section 4.3, we switch to CCVT optimization, which converges with a small number of iterations due to the good initialization. Our experiments have demonstrated the efficiency of this simple strategy.

The remainder of the paper is organized as follows. We will introduce a novel RVD computation algorithm in Section 3. The complete isotropic remeshing framework based on the exact RVD computation is described in Section 4. Experimental results are presented in Section 5 and conclusions are drawn in Section 6.

3. RVD computation

Suppose that the input mesh surface **S** is represented by a set of triangles $\{\mathbf{t}_j\}_{j=1}^m$. The RVD of the seeds $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ on **S** is the intersection between the Voronoi diagram and **S**. For each Ω_i this intersection is the union of the intersections between Ω_i and all the triangles of **S**, *i.e.* , $\mathbf{R}_i = \Omega_i \cap \mathbf{S} = \cup_{\mathbf{t}_j \in \mathbf{S}}\{\Omega_i \cap \mathbf{t}_j\}$. Since both Ω_i and triangle $\mathbf{t}_j$ are convex, their intersection is a convex polygon, if any. Triangles shared by different Voronoi cells need to be split along the bisecting planes of these cells.

Checking all pairs of Voronoi cells and mesh facets would take $O(mn)$ time, where n and m are the number of seeds and the number of triangles, respectively. We developed a more efficient algorithm with the following ideas. We process all triangle faces one by one. For each triangle, we use a pre-computed *kd*-tree and the Voronoi diagram to quickly identify the incident Voronoi cells of the triangle. An incident cell of a triangle is a cell that intersects or contains the triangle.

3.1. Outline

The main flow of the RVD computation is as follows. Firstly, a *kd*-tree is built for all the seeds **X**, and a Delaunay triangulation of these seeds is constructed to build the Voronoi cells $\{\Omega_i\}_{i=1}^n$. A Voronoi cell Ω_i is represented by its bounding planes $\mathbf{Q}_i = \{P_k^i\}$.

For each triangle face $\mathbf{t}_j$, we use the *kd*-tree to find the closest seed $\tilde{\mathbf{x}}$ to the centroid of $\mathbf{t}_j$. Clearly, the Voronoi cell $\tilde{\Omega}$ of $\tilde{\mathbf{x}}$ has nonempty intersection with $\mathbf{t}_j$,*i.e.* $\tilde{\Omega}$ is an incident cell of $\mathbf{t}_j$. In the following, we first explain the cell-triangle intersection algorithm in Section 3.2 and then we introduce the traversing algorithm to identify all the incident Voronoi cells of a triangle $\mathbf{t}_j$ in Section 3.3.

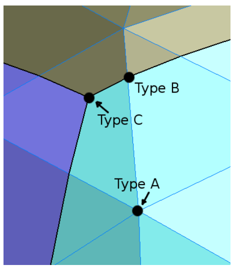
3.2. Cell-triangle intersection

To construct the *restricted Voronoi cells*, or RVCs for short (Figure 2), we need to compute the intersection of a Voronoi cell Ω_i and a triangle $\mathbf{t}_j$ of **S**. This is done by clipping $\mathbf{t}_j$ against all the bounding planes P_k of Ω_i. This procedure is similar to the clipping algorithm in [SH74]. In addition, we keep track of the defining equations of each vertex $\mathbf{v}$ generated by the algorithm, as functions of the vertices $\mathbf{q}_j$ of the original surface and the vertices $\mathbf{x}_i$ of the remesh, since each vertex $\mathbf{v}$ is the intersection of three planes. This additional symbolic information will be used twice, i.e. by our exact predicates (Section 3.4) and by our topology control method (Section 4.4).

These planes are either facets of the original surface or bisectors of the Delaunay triangulation.

There are three possible configurations (see the small figure on the right):

Type A: $\mathbf{v}$ is a vertex of the original surface;
Type B: $\mathbf{v}$ is the intersection between an edge of the original surface (i.e. two facets) and a bisector of two seeds $\mathbf{x}_i$, $\mathbf{x}_j$;
Type C: $\mathbf{v}$ is the intersection between a facet of the original surface and two bisectors $\mathbf{x}_i, \mathbf{x}_j$ and $\mathbf{x}_i, \mathbf{x}_k$.



In our implementation, this information is encoded as a set $Sym_{\mathbf{v}} = \{k_1, k_2, k_3\}$ of three integers attached to each vertex $\mathbf{v}$. A positive integer k corresponds to the Delaunay bisector of $[\mathbf{x}_1, \mathbf{x}_k]$, where $\mathbf{x}_1$ denotes the current Delaunay vertex (we are currently clipping $\mathbf{S}$ by $\Omega(\mathbf{x}_1)$) and a negative integer k corresponds to the $(-k)^{th}$ facet of the original mesh.

The intersection $\mathbf{v}$ between the edge $[\mathbf{v}_1, \mathbf{v}_2]$ and bisector $[\mathbf{x}_1, \mathbf{x}_k]$ is computed *symbolically* as follows :

$$Sym_{\mathbf{v}} = Sym_{\mathbf{v}_1} \cap Sym_{\mathbf{v}_2} \cup \{k\}$$

Note that a configuration of type 3 may be degenerate if two bisectors intersect exactly on a vertex or on an edge of the original surface. In this case, we represent the vertex by the intersection of the two bisectors and one of the facets incident to the vertex (resp. edge), i.e. configuration 3. With this convention, the sets Sym are of fixed size (3 elements), and sets intersection are computed in constant time. This also preserves the sets of bisectors, relevant to compute the Restricted Delaunay Triangulation (Section 4.4).

3.3. Clipping by incident cells

We use an FIFO queue Q to facilitate the intersect computation of an input triangle $\mathbf{t}_j$ with all its incident cells. We assign a boolean flag to each cell for recording whether this cell has been intersected with triangle $\mathbf{t}_j$. The flag of each cell is set to $false$ at the beginning of the algorithm.

The queue Q is initialized by the nearest seed $\tilde{\mathbf{x}}$ of $\mathbf{t}_j$. While Q is not empty, the seed in the front of Q is popped out, denoted as $\mathbf{x}_t$, we compute the intersection of Voronoi cell Ω_t of $\mathbf{x}_t$ and $\mathbf{t}_j$. During the intersection process, if a bounding plane of Ω_t has intersection with the current intersected polygon, we push the opposite seed of $\mathbf{x}_t$ into the queue if the opposite seed's intersection flag is $false$. Each time after intersection, the intersection flag of $\mathbf{x}_t$ is set to $true$ and the resulted intersected polygon is associated to the restricted Voronoi diagram of $\mathbf{R}_t$. This procedure is repeated until Q becomes empty. Figure 3 demonstrates the process of clipping a triangle with its three incident Voronoi cells. After processing a triangle, the flags of the incident cells are reset to $false$ before processing the next triangle.

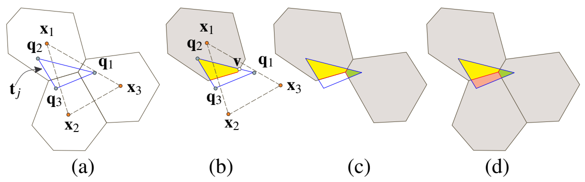


Figure 3: *(a) The dashed triangle is a Delaunay triangle formed by three seeds and the blue triangle $\mathbf{t}_j$ is an input triangle. (b) Clip $\mathbf{t}_j$ against $\mathbf{x}_1$'s Voronoi cell Ω_1, vertex $\mathbf{v}$ is the intersection between $[\mathbf{q}_1, \mathbf{q}_2]$ and bisector $[\mathbf{x}_1, \mathbf{x}_3]$; (c), (d) propagation to Ω_2 and Ω_3.*

3.4. Exact predicates

To compute RVD with certified combinatorics, we need the predicate $side(\mathbf{x}_1, \mathbf{x}_2, \mathbf{v})$ that returns the side of point $\mathbf{v}$ with respect to the bisector of $[\mathbf{x}_1, \mathbf{x}_2]$ where $\mathbf{v}$ is a vertex of the RVD and where $\mathbf{x}_1$ and $\mathbf{x}_2$ are Delaunay vertices (seeds).

At first sight, it seems impossible to use exact predicates, because we keep computing intersections between segments and planes, where the extremities of the segments can be the result of previous intersections. However, we can use our symbolic information, that relates all vertices $\mathbf{v}$ computed by the algorithm to the data (original mesh vertices $\mathbf{q}_j$ and Delaunay vertices $\mathbf{x}_i$). The three possible configurations for a vertex $\mathbf{v}$ (see Section 3.2) yield three versions of the predicate $side(\mathbf{x}_1, \mathbf{x}_2, \mathbf{v})$:

Case A : $\mathbf{v}$ is directly given, in other words $\mathbf{v}$ is a vertex $\mathbf{q}$ of the surface $\mathbf{S}$, and $Sym_{\mathbf{v}}$ contains only negative IDs). $side(\mathbf{x}_1, \mathbf{x}_2, \mathbf{v}) = sideA(\mathbf{x}_1, \mathbf{x}_2, \mathbf{q})$;

Case B : $\mathbf{v}$ is the intersection between a bisector and an edge of the surface $\mathbf{S}$. $Sym_{\mathbf{v}}$ has a positive ID $\longrightarrow \mathbf{x}_3$ and two negative IDs $\longrightarrow \mathbf{q}_1, \mathbf{q}_2$ that corresponds to an edge of the facet $\mathbf{t}$, $\mathbf{q}_1, \mathbf{q}_2$ are vertices of the edge (see Figure 3(b)). $side(\mathbf{x}_1, \mathbf{x}_2, \mathbf{v}) = sideB(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{q}_1, \mathbf{q}_2)$, where $\mathbf{v} = bisector(\mathbf{x}_1, \mathbf{x}_3) \cap [\mathbf{q}_1, \mathbf{q}_2]$;

Case C : $\mathbf{v}$ is the intersection between two bisectors and a facet of the surface $\mathbf{S}$. $Sym_{\mathbf{v}}$ has two positive IDs $\longrightarrow \mathbf{x}_3, \mathbf{x}_4$ and one negative ID that corresponds to a facet $\mathbf{t}$. $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$ are facet $\mathbf{t}$'s vertices. $side(\mathbf{x}_1, \mathbf{x}_2, \mathbf{v}) = sideC(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$, where $\mathbf{v} = bisector(\mathbf{x}_1, \mathbf{x}_3) \cap bisector(\mathbf{x}_1, \mathbf{x}_4) \cap plane(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$.

$sideA$, $sideB$, $sideC$ are rational fractions of low-degree (resp. 2, 4/2 and 6/4). To evaluate a predicate, we evaluate the sign of the numerator and denominator separately, and use a hierarchy of filters (almost static, interval arithmetics then MP_Float) to speed-up the evaluation. Our implementation is done in *CGAL* (`http://www.cgal.org/`) using Meyer and Pion's FPG predicate generator [MP08].

4. Isotropic remeshing

The CVT-based remeshing framework includes initialization, optimization and final mesh extraction. In this section, we explain each of these components.

4.1. Initialization

Good distribution of initial seeds can help saving much time in optimization. For an isotropic mesh with constant density, a user-specified number of initial seeds $\mathbf{X}$ are generated on $\mathbf{S}$ with a uniform random distribution according to surface areas. If a density function $\rho(\mathbf{x})$ is specified, then the initial seed should be distributed on $\mathbf{S}$ with the probability density function $g(\mathbf{x}) = \sqrt{\rho(\mathbf{x})} / \int_{\mathbf{S}} \sqrt{\rho(\mathbf{x})}\, d\sigma$, which is easily derived from the equilibrium of CVT energy values of all Voronoi cells in a CVT [DFG99]. A natural choice for ρ, suggested by the ε-sampling criterion (Theorem 1) is to use $\rho(\mathbf{x}) = 1/lfs(\mathbf{x})^2$. As done in [ACSYD05], we compute the poles [AB99] of an initial sampling of the surface, and use the distance to the nearest pole as an approximation of lfs.

4.2. Optimization

We use the L-BFGS based CVT computation, presented in [LWL*09], to optimize the seeds. L-BFGS requires gradients to estimate an approximate inverse Hessian for seed updates. We first explain this computation for a surface without features or boundaries. With RCVT (i.e. restricted CVT), the gradient of CVT energy is [IMO84]:

$$\frac{\partial F}{\partial \mathbf{x}_i} = 2m_i(\mathbf{x}_i - \mathbf{x}_i^*), \tag{5}$$

where $m_i = \int_{\mathbf{R}_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\sigma$, and $\mathbf{x}_i^*$ is the restricted centroid given by

$$x_i^* = \frac{\sum_{\mathbf{t}_k \in \mathbf{R}_i} \int_{\mathbf{t}_k} \rho(\mathbf{x})\mathbf{x}\, d\sigma}{\sum_{\mathbf{t}_k \in \mathbf{R}_i} \int_{\mathbf{t}_k} \rho(\mathbf{x})\, d\sigma}. \tag{6}$$

Here the $\mathbf{t}_k$ are the clipped polygons within the restricted Voronoi region $\mathbf{R}_i \subset \mathbf{S}$. To integrate this function, a polygon is split into triangles by simply connect one vertex with other non-connected vertices. The density function $\rho(\mathbf{x})$ in a triangle is defined by linear interpolation of its values at the vertices of the triangle.

When minimizing the CCVT (i.e. constrained CVT), since the seeds are constrained on the input surface $\mathbf{S}$, the gradient need to be constrained within the tangent space of $\mathbf{S}$. Thus we have

$$\left.\frac{\partial F}{\partial \mathbf{x}_i}\right|_{\mathbf{S}} = \frac{\partial F}{\partial \mathbf{x}_i} - \left[\frac{\partial F}{\partial \mathbf{x}_i} \cdot \mathbf{N}(\mathbf{x}_i)\right] \mathbf{N}(\mathbf{x}_i), \tag{7}$$

where $\frac{\partial F}{\partial \mathbf{x}_i}$ is computed as in Equation (5).

Once the gradient is available, the L-BFGS method will perform one iteration to update all the seeds. The details of its update formula can be found in [LN89] or [LWL*09]. Once the updated seeds are available, we compute the RVD of the seeds using the algorithm described in Section 3. This process is repeated until convergence.

4.3. Feature preservation

The features of the input mesh $\mathbf{S}$, including boundary curves, creases, and corners, are first identified, either automatically by some other algorithm or manually by the user. We will refer to both creases and boundary curves as feature curves. The feature curves are represented by a set of line segments, called *feature edges*. The vertices that have more than two incident feature edges, as well as tips, darts and cusps are considered as corner vertices, which will be used as *corner seeds* and not be updated by optimization. The Voronoi cell of a corner seed contributes a CVT energy term. However, since a corner seed is not to be updated, it will not be treated as a variable.

The key to preserving feature curves while ensuring a high quality remeshing result is to determine how many seeds should be allocated to each feature curve. We use a two-stage strategy for this task. We first compute an (unconstrained) Restricted Centroidal Voronoi Tessellation without considering the feature curves to obtain an optimal distribution of a prescribed number of seeds across $\mathbf{S}$. Once the RCVT has converged, the seeds whose Voronoi cells contain a feature curve are snapped onto the feature curve and become *feature seeds*. Then, we switch to (constrained) CCVT, explained in the previous section. In addition to projecting the seeds onto the surface and constraining their gradient in the tangent plane (Equation 7), we project each updated feature seed onto the nearest point of the feature curve $\mathcal{C}$ at each iteration. The gradient of a feature $\mathbf{x}_i$ is restricted to the tangent space of the feature curve as follows :

$$\left.\frac{\partial F}{\partial \mathbf{x}_i}\right|_{\mathcal{C}} = \left[\frac{\partial F}{\partial \mathbf{x}_i} \cdot \mathbf{T}(\mathbf{x}_i)\right] \mathbf{T}(\mathbf{x}_i), \tag{8}$$

where $\mathbf{T}(\mathbf{x}_i)$ is the unit tangent vector of the feature curve at $\mathbf{x}_i$, and $\frac{\partial F}{\partial \mathbf{x}_i}$ is computed by Equation (5).

The above way of defining feature seeds is slightly different from the method in [ACDI05], which does not optimize the feature seeds anymore after first running a 1-D CVT on feature curves. In contrast, our method continues to optimize the feature seeds throughout the entire CVT optimization to achieve better results.

4.4. Final mesh extraction

Restricted Delaunay triangulation After the optimization, the final mesh is extracted as the restricted Delaunay triangulation (RDT) (see Section 2.3). In our case, the RDT can be determined combinatorially, from the symbolic information computed in Section 3.2. When we compute the restricted Voronoi cell associated to each vertex $\mathbf{x}_i$, we detect the RDT faces as follows :

– if an RVD vertex $\mathbf{v}$ is on two bisectors $\mathbf{x}_i, \mathbf{x}_j$ and $\mathbf{x}_i, \mathbf{x}_k$, i.e. $Sym_{\mathbf{v}}$ has two positive IDs (i, j), then $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ is a triangle of the RDT;
– if an RVD vertex $\mathbf{v}$ is on one bisector $\mathbf{x}_i, \mathbf{x}_j$, i.e. $Sym_{\mathbf{v}}$ has one positive ID j, then $(\mathbf{x}_i, \mathbf{x}_j)$ is an edge of the RDT;

– the vertices of the RDT are the $\mathbf{x}_i$'s.

Topological ball property - topology control In addition, we check the validity of the remesh, that is to say we ensure that the RVD satisfies the Topological Ball Property. To do so, we compute the Euler-Poincaré characteristic of the RVCs, their number of connected components and number of borders to check whether they are topological discs. The symbolic information of the vertices is used as a unique identifier. Similarly we compute the number of components and number of extremities of the Restricted Voronoi edges, and we check that each Voronoi edge intersects the surface **S** at a single point. All these tests are purely combinatorial. However, they are computationally expensive. For this reason, we also use simpler tests, that detect duplicated Delaunay triangles (which is equivalent to the uniqueness of the Voronoi edge / surface intersection), non-manifold Delaunay edges (connected to less than 1 or more than 2 triangles) and isolated Delaunay vertices. These RDT manifoldness tests are all implemented using tables, indexed by the symbolic information (indices). This may miss configurations where a small connected component of **S** is completely included in a single Voronoi cell. For this reason, the more expensive Topological Ball Property test is applied if the RDT manifoldness test succeeds (RDT manifoldness is used as a filter for the Topological Ball Property test).

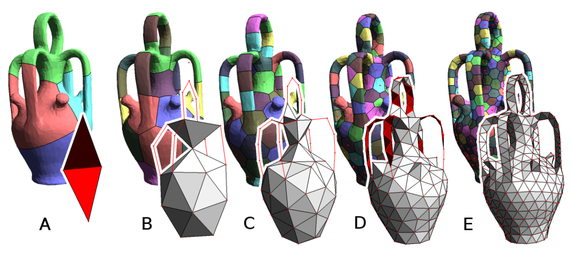
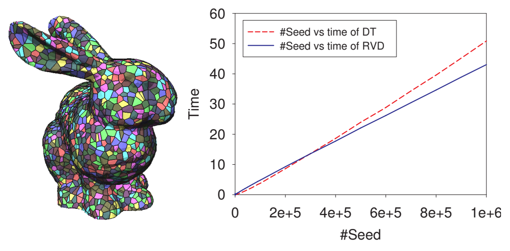


Figure 4: *Interleaved topology control / optimization.*

As shown in Figure 4, we use these topological tests in an interleaved Lloyd optimization / topology control scheme. In this example, we started with 4 vertices (Figure 4-A). Every 30 iterations, topology is checked. Non-manifold Delaunay simplices are detected (shown in red in the figure). Then we generate additional vertices, two vertices for each non-manifold Delaunay triangle, shifted above and below the center of the triangle along the normal, and three vertices for each non-manifold Delaunay edge, shifted from three directions that radiate around the center of the edge. In Figure (4-B) and (4-C), the handles are sampled by non-manifold edges, that correspond to cylindrical RVCs (they violate the Topological Ball Property). In Figure (4-D), they are sampled by "plates" of non-manifold Delaunay triangles. They correspond to RVCs that have two connected components (which also violate the Topological Ball Property). These triangles appear twice in the triangulation, once for each orientation. Finally, in Figure (4-E), after 4 iterations of topology control, the topology of the object is fully recovered.

5. Experimental results

In this section, we present experimental results of the presented RVD computation algorithm and isotropic remeshing framework. We use *CGAL* to compute Delaunay triangulation and use the *ANN* library [MA97] to build the *kd*-tree. All the experiments were conducted on a laptop PC with a 2.2 GHz Intel Duo-Core processor and 2GB memory.



We now evaluate the performance of our RVD computation algorithm. We used an input model (Bunny) with 5k faces. We test our algorithm with different numbers of seeds, from 10 to 1 million, sampled on the surface. From the timing curves, we see that our method for RVD is about as fast as computing the Delaunay triangulation of the seeds.

Quality measurements. We use the criteria in [FB97] to measure the remeshing quality (see Table 1 and 2). The quality of a triangle is measured by $Q_\mathbf{t} = \frac{6}{\sqrt{3}} \frac{S_\mathbf{t}}{p_\mathbf{t} h_\mathbf{t}}$, where $S_\mathbf{t}$ is the area of **t**, $p_\mathbf{t}$ the half-perimeter of **t** and $h_\mathbf{t}$ the the longest edge length of **t**. $Q_{\min}$ is the minimal quality of a triangle and Q_{ave} is the average triangle quality. $\theta_{\min}$ is the smallest angle of the minimal angles of all triangles and $\theta_{\min,\text{ave}}$ is the average of minimal angles of all triangles. $\theta < 30^\circ$ is the percentage of triangles with its minimal angle smaller than 30°.

Models with features. Figure 5 shows two examples of applying our remeshing method to meshes with sharp features. The Fandisk model has 13k facets and is down-sampled with 3k seeds, and the Joint model has 446 facets and is up-sampled with 3k seeds. Measurements of meshing qualities of these models are listed in Table 1.

Irregular or noisy models. Figure 6 shows the uniform remeshing of the Dancer model output from a visual hull algorithm. This model contains many badly shaped triangles, which are narrow and long. Our method is also robust for meshes with important noise, as shown in Figure 7. Here only RCVT is used, since the roughness of the input mesh makes the gradient evaluation of CCVT unreliable. Remeshing of noisy models is very challenging for parameterization-based method due to the severe local metric distortion. To our knowledge, no parameterization-based method is able to handle such noisy models. Our remeshing result here is also better than the clustering-based approach, as shown in Figure 7 and Table 2.

Comparison. Compared with the parameterization-based methods [ACDI05, SAG03], our method avoids the inaccuracy due to the approximation and metric distortion in parameterization, as well as the need to stitch together local

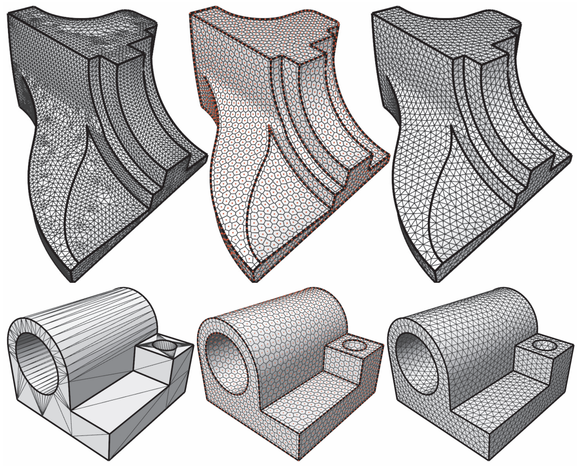


Figure 5: *Uniform CVT and remeshing results of models with shape features. Top: The Fandisk model (13k faces, 3k seeds). It takes 100 iterations and 40 seconds; Bottom: the Joint model (446 faces, 3k seeds), it takes 119 iterations and 49 seconds.*

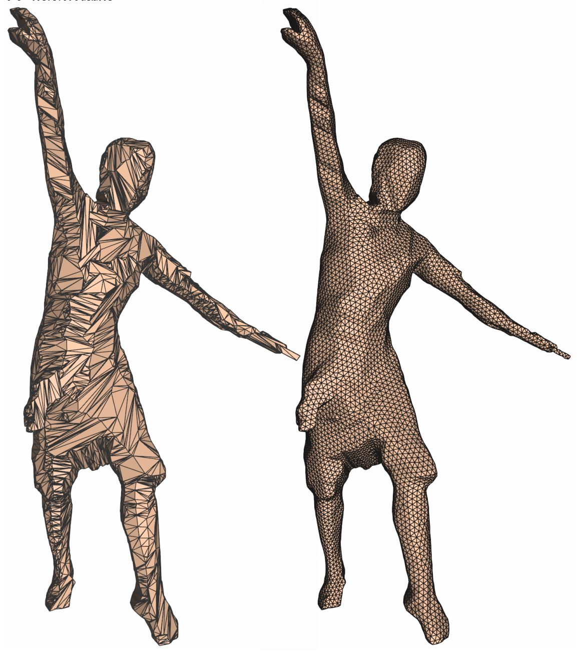


Figure 6: *Uniform remeshing of Dancer model (13.7k faces, 10k seeds), 51 iterations in 56.7 s.*

Model	$Q_{\min}$	Q_{ave}	$\theta_{\min}$	$\theta_{\min,\text{ave}}$	$\theta < 30^\circ$
Fandisk	0.541	0.897	24.35	51.68	6.04×10^{-4}
Joint	0.585	0.913	31.89	52.88	0
Dancer	0.604	0.919	30.81	53.22	0

Table 1: *Meshing quality of models with sharp features or boundaries and models output from computer vision algorithms.*

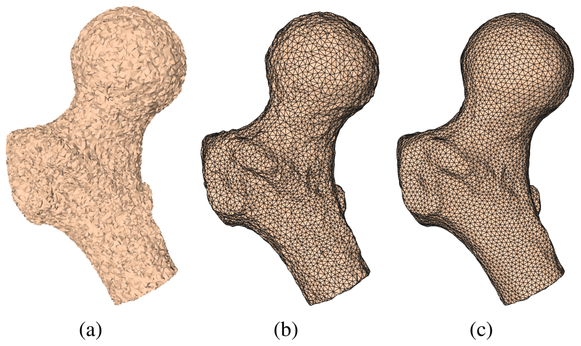


Figure 7: *Uniform remeshing results of the noisy Ball Joint (68.5k faces, 10k seeds). (a) input mesh; (b) remeshing result of [VCP08]; and (c) our result (259 iterations using 386 seconds). The quality comparison is given in Table 2.*

charts for a complex surface. Table 2 compares our results with several existing remeshing techniques on various models. The elk model has 10k facets and 5k vertices and it was up-sampled to 31.1k vertices (Figure 8). The David model has 700k facets and 350k vertices and it was down-sampled to 100k vertices. As can be seen, our method generates better remeshing results than all the other approaches, especially in terms of $Q_{\min}$ and $\theta_{\min}$. These two quality measures are especially important, since they define the condition number of numerical methods applied to the mesh [DWZ09]. The high quality of our results is due to the exact computation of restricted Voronoi diagram directly on mesh surfaces. Compared with the clustering-based approach [VCP08], our method is robust to irregular sampled or noisy input meshes, whereas clustering-based methods are highly dependent on the quality of the input mesh. In contrast, our method works well even when many Voronoi cells are entirely included in an initial triangle (Figure 8). Compared with Delaunay refinement [CDL07], as shown in Figure 9 and in Table 2, since our method optimizes all the vertices simultaneously, it generates a mesh of higher quality.

6. Conclusion

We have presented a CVT based remeshing algorithm. The key contribution is a new efficient technique for computing exact RVD on 3D mesh surfaces. Combined with the new L-BFGS based optimization technique, our remeshing algorithm is robust and efficient for large meshes, and achieves high quality remeshing results.

Limitations and future work. A benefit of our approach as compared to Delaunay refinement is that all vertices are simultaneously optimized, which gives more degrees of freedom and generates triangles of higher quality (see the comparison with [CDL07] in Figure 9). However, unlike Delaunay refinement, our strategy to check for the Topological Ball Property and iteratively insert new vertices is not guaranteed to terminate. One may switch to Delaunay refinement in the case where our algorithm does not terminate after a certain number of iterations. We did not encounter any problem in practice, even with difficult models (flat elephant

Model	Q_{min}	Q_{ave}	θ_{min}	$\theta_{min,ave}$	$\theta < 30°$	Mean(%bb)	RMS(%bb)
David [SAG03]	0.027	0.91	0.92	52.9	0.41	0.062	0.07
David [VCP08]	0.013	0.80	0.85	45.2	1.2	2.35×10^{-3}	3.7×10^{-3}
David [ours]	0.544	0.933	28.46	54.37	3.33×10^{-6}	8.1×10^{-3}	0.012
Elk [SAG03]	0.092	0.929	4.72	54.12	0.074	0.034	0.049
Elk [VCP08]	0	0.782	0	43.24	2.445	0.002	0.005
Elk [BH96]	0.285	0.908	16.96	51.94	4.65×10^{-5}	0.021	0.03
Elk [ours]	0.635	0.932	36.46	54.41	0	0.006	0.012
Balljoint [VCP08]	0.105	0.78	5.97	43.01	2.379	0.154	0.174
Balljoint [ours]	0.47	0.86	26.0	48.83	3.25×10^{-4}	0.166	0.184
Homer [CDL07]	0.08	0.79	2.79	42.98	0.018	0.081	0.099
Homer [ours]	0.536	0.536	26.0	48.83	1.1×10^{-4}	0.029	0.035

Table 2: *Comparison of remeshing qualities. Mean(%bb) and RMS(%bb) are the mean and RMS Hausdorff error between the input mesh and output mesh, divided by the diagonal of the bounding box of the input mesh.*

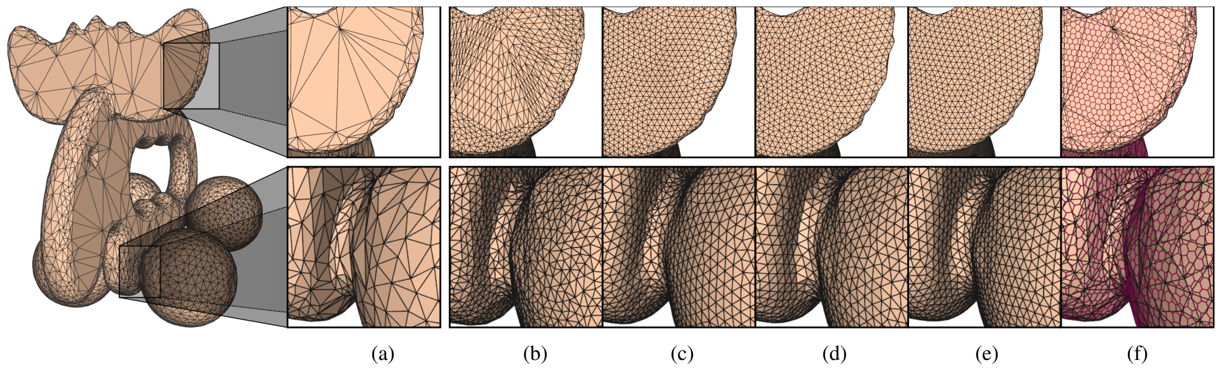


Figure 8: *Comparison of remeshing results (10.4k faces, 31.1k seeds). (a) Input Elk model; (b) [VCP08]; (c) [SAG03]; (d) [BH96]; (e) ours (48 iterations using 132 seconds); (f) RVD of our method.*

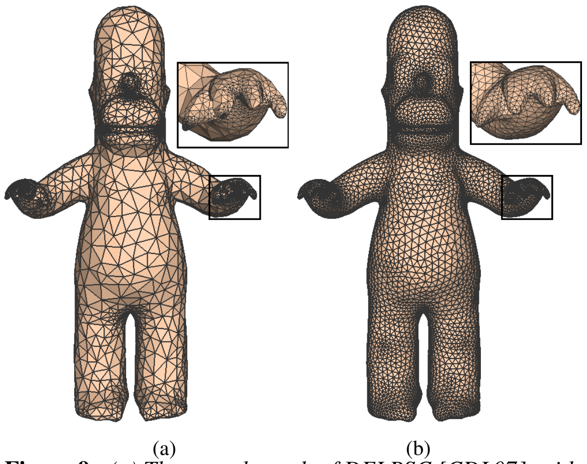


Figure 9: *(a) The remesh result of DELPSC [CDL07], with required minimal angle as 30°, DELPSC produces 7,588 seeds and $\theta_{min} = 2.79°$; (b) our result, with the same number of vertices and $\rho = 1/lfs^2$. We obtain $\theta_{min} = 26°$. The quality comparison is given in Table 2.*

ears and thin tail in Figure 10, David's hair in Figure 11). However, designing an algorithm that shares both properties (global optimization and termination guarantees) is still an open problem. As suggested in [ACSYD05], designing remeshing algorithms with certified minimum angle is also a research topic.

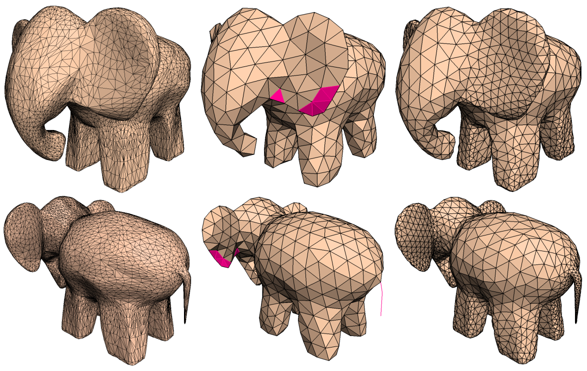


Figure 10: *Topology control. (a) Input mesh; (b) red triangles (ears) and edges (tail) violate the topological ball property; (c) final result, homeomorphic to the initial mesh.*

Acknowledgements

The models used in this paper are from the Aim@Shape and courtesy of the Stanford 3D Scanning Repository. The dancer model in Figure 6 is from Edmond Boyer. We would like to thank Pierre Alliez, Vitaly Surazhsky and Sébastien Valette for providing their remeshing results. We also thank Sylvain Pion for the FPG predicate generator and Tamal K. Dey for the DELPSC software. B. Lévy and Y. Liu are supported by GOODSHAPE (European Research Council starting grant FP7-ERC-StG-205693).

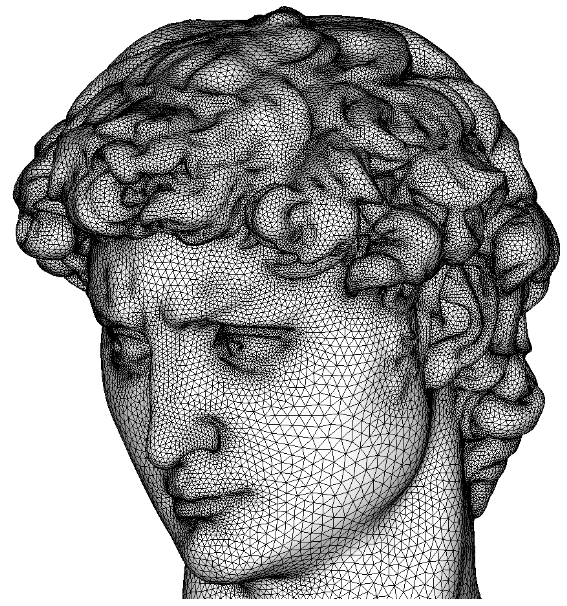


Figure 11: *Remeshed David, density* $\rho = 1/lfs^2$.

References

[AB99] AMENTA N., BERN M.: Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry 22* (1999), 481–504.

[ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new Voronoi-based surface reconstruction algorithm. In *Proc. Siggraph'98* (1998), pp. 415–421.

[ACDI05] ALLIEZ P., COLIN DE VERDIÈRE É., DEVILLERS O., ISENBURG M.: Centroidal Voronoi diagrams for isotropic surface remeshing. *Graphical Models 67*, 3 (2005), 204–231.

[ACSYD05] ALLIEZ P., COHEN-STEINER D., YVINEC M., DESBRUN M.: Variational tetrahedral meshing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) 24*, 3 (2005), 617–625.

[AMD02] ALLIEZ P., MEYER M., DESBRUN M.: Interactive geometry remeshing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (2002), 347–354.

[AUGA08] ALLIEZ P., UCELLI G., GOTSMAN C., ATTENE M.: Recent advances in remeshing of surfaces. In *Shape Analysis and Structuring* (2008), pp. 53–82.

[Aur91] AURENHAMMER F.: Voronoi diagrams: a survey of a fundamental geometric data structure. *ACM Computing Surveys 23* (1991), 345–405.

[BH96] BOSSEN F. J., HECKBERT P. S.: A pliant method for anisotropic mesh generation. In *5th Intl. Meshing Roundtable* (1996), pp. 63–74.

[BO06] BOISSONNAT J.-D., OUDOT S.: Provably good sampling and meshing of Lipschitz surfaces. In *Symposium on Computational Geometry Conf. Proc.* (2006), ACM, pp. 337–346.

[CDL07] CHENG S.-W., DEY T. K., LEVINE J. A.: A practical Delaunay meshing algorithm for a large class of domains. In *16th Intl. Meshing Roundtable* (2007), pp. 477–494.

[CDR07] CHENG S.-W., DEY T. K., RAMOS E. A.: Delaunay refinement for piecewise smooth complexes. In *Proc. 18th Annu. ACM-SIAM Sympos. Discrete Algorithms* (2007), pp. 1096–1105.

[DFG99] DU Q., FABER V., GUNZBURGER M.: Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review 41* (1999), 637–676.

[DGJ03] DU Q., GUNZBURGER M. D., JU L.: Constrained centroidal Voronoi tesselations for surfaces. *SIAM J. SCI. COMPUT. 24*, 5 (2003), 1488–1506.

[DWZ09] DU Q., WANG D., ZHU L.: On mesh geometry and stiffness matrix conditioning for general finite element spaces. *SIAM J. Numer. Anal. 47*, 2 (2009), 1421–1444.

[ES97] EDELSBRUNNER H., SHAH N. R.: Triangulating topological spaces. *IJCGA 7*, 4 (1997), 365–378.

[FB97] FREY P., BOROUCHAKI H.: Surface mesh evaluation. In *6th Intl. Meshing Roundtable* (1997), pp. 363–374.

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE. W.: Mesh optimization. In *Proceedings of ACM SIGGRAPH 1993 Conference* (1993), pp. 19–26.

[HG97] HECKBERT P., GARLAND M.: Survey of polygonal surface simplification algorithms. In *SIGGRAPH 97 Course Notes: Multiresolution Surface Modeling* (1997).

[IMO84] IRI M., MUROTA K., OHYA T.: A fast Voronoi diagram algorithm with applications to geographical optimization problems. In *IFIP Conf. Proc.* (1984), pp. 273–288.

[KWR97] KUNZE R., WOLTER F.-E., RAUSCH T.: Geodesic Voronoi diagrams on parametric surfaces. In *Proceedings of Computer Graphics International 1997* (1997), pp. 230–237.

[Llo82] LLOYD S.: Least square quantization in PCM. *IEEE Trans. Inform Theory 28* (1982), 129–137.

[LN89] LIU D., NOCEDAL J.: On the limited memory method for large scale optimization. *Mathematical Programming B 45*, 3 (1989), 503–528.

[LWL*09] LIU Y., WANG W., LÉVY B., SUN F., YAN D.-M., LU L., YANG C.: On centroidal Voronoi tessellation - energy smoothness and fast computation. *ACM Transactions on Graphics* (2009). To appear.

[MA97] MOUNT D. M., ARYA S.: ANN: A library for approximate nearest neighbor searching. In *Proc. 2nd Annual CGC Fall Workship on Computational Geometry* (1997), pp. 33–40.

[MP08] MEYER A., PION S.: FPG: A code generator for fast and certified geometric predicates. *RNC* (2008), 47–60.

[OBSC00] OKABE A., BOOTS B., SUGIHARA K., CHIU S. N.: *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd ed. Wiley, 2000.

[PC06] PEYRÉ G., COHEN L. D.: Geodesic remeshing using front propagation. *Int. J. of Computer Vision 69* (2006), 145–156.

[SAG03] SURAZHSKY V., ALLIEZ P., GOTSMAN C.: Isotropic remeshing of surfaces: a local parameterization approach. In *12th Intl. Meshing Roundtable* (2003), pp. 204–231.

[SH74] SUTHERLAND I. E., HODGMAN G. W.: Reentrant polygon clipping. *Communications of the ACM 17*, 1 (1974), 32–42.

[She02] SHEWCHUK J. R.: What is a good linear element? interpolation, conditioning, and quality measures. In *11th Intl. Meshing Roundtable* (2002), pp. 115–126.

[VCP08] VALETTE S., CHASSERY J.-M., PROST R.: Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 369–381.