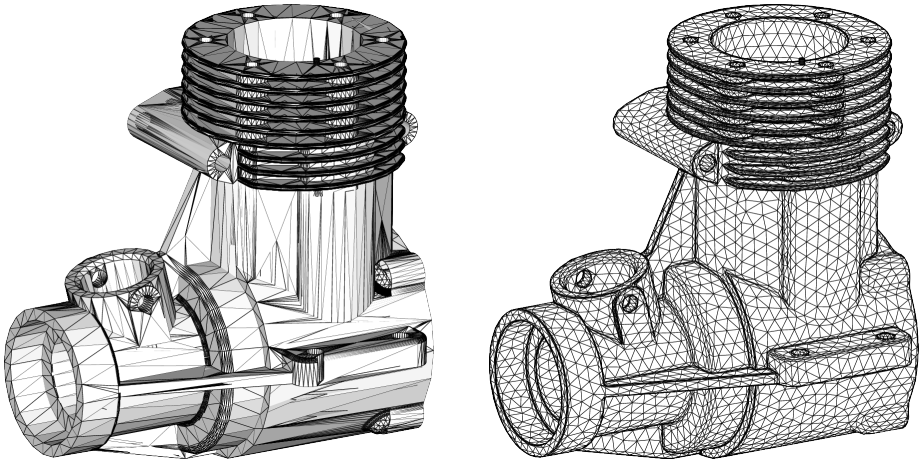
Variational Anisotropic Surface Meshing with Voronoi Parallel Linear Enumeration

Bruno Lévy[1] and Nicolas Bonneel[1,2]

[1] Project ALICE, INRIA Nancy Grand-Est and LORIA
[2] Harvard University
Bruno.Levy@inria.fr, nbonneel@seas.harvard.edu



This paper introduces a new method for anisotropic surface meshing. From an input polygonal mesh and a specified number of vertices, the method generates a curvature-adapted mesh. The main idea consists in transforming the 3d anisotropic space into a higher dimensional isotropic space (typically 6d or larger). In this high dimensional space, the mesh is optimized by computing a Centroidal Voronoi Tessellation (CVT), i.e. the minimizer of a C^2 objective function that depends on the coordinates at the vertices (quantization noise power). Optimizing this objective function requires to compute the intersection between the (higher dimensional) Voronoi cells and the surface (Restricted Voronoi Diagram). The method overcomes the d-factorial cost of computing a Voronoi diagram of dimension d by directly computing the restricted Voronoi cells with a new algorithm that can be easily parallelized (Vorpaline: Voronoi Parallel Linear Enumeration). The method is demonstrated with several examples comprising CAD and scanned meshes.

1 Introduction

In this paper, we propose a new method for anisotropic meshing of 3d surfaces, based on an anisotropic variant of Centroidal Voronoi Tessellation. The input of the algorithm is an initial polygon mesh. From a user-specified desired number of vertices and a parameter that specifies the desired amount of anisotropy, our method generates a curvature-adapted anisotropic mesh.

Anisotropic mesh generation is an important and difficult topic. For instance, anisotropic meshes are used by numerical simulation in computational fluid dynamics, that need meshes with anisotropic elements adapted to a prescribed metric tensor field [2]. This improves the accuracy of the simulation without increasing the number of elements too much. The main difficulty is to generate high cell aspect ratio (typically 10,000:1) for a metric tensor field with sharp variations of orientation [27, 24]. In this paper, we consider anisotropic surface meshing, with the goal of reducing the required number of elements to represent a surface with the same level of accuracy. Our idea is based on the notion of Centroidal Voronoi Tessellation (Section 2.1) adapted to the anisotropic setting (Section 2.2) by operating in a higher-dimensional space (Section 3). To avoid the $d!$ space-time factor of d-dimensional Voronoi diagrams, we introduce Vorpaline (Voronoi Parallel Linear Enumeration), an algorithm that computes the Voronoi cells by iterative half-space clipping (Section 4). A superset of the contributing half-spaces is determined by a simple geometric characterization (Section 4.1). Some anisotropic surface remeshings obtained with the method are shown in Section 5. The limitations of the method and suggestions for future work are discussed in Section 6.

2 Background and Previous Work

2.1 Centroidal Voronoi Tessellation

Voronoi and Delaunay

Given a set of points $\mathbf{X} = (\mathbf{x}_1 \dots \mathbf{x}_n) \in I\!R^d$, the Voronoi diagram $\mathrm{Vor}(\mathbf{X})$ is the collection of subsets $\mathrm{Vor}(\mathbf{x}_i)$ defined by :

$$\mathrm{Vor}(\mathbf{X}) = \{\mathrm{Vor}(\mathbf{x}_i)\} \quad ; \quad \mathrm{Vor}(\mathbf{x}_i) = \left\{\mathbf{x} \in I\!R^d | d(\mathbf{x}, \mathbf{x}_i) \leq d(\mathbf{x}, \mathbf{x}_j) \forall j\right\}$$

where $d(.,.)$ denotes the Euclidean distance. The subset $\mathrm{Vor}(\mathbf{x}_i)$ is called the Voronoi cell of $\mathbf{x}_i$. The dual of the Voronoi diagram is called the Delaunay triangulation. Each pair of Voronoi cells $\mathrm{Vor}(\mathbf{x}_i), \mathrm{Vor}(\mathbf{x}_j)$ that has a non-empty intersection defines an edge $(\mathbf{x}_i, \mathbf{x}_j)$ in the Delaunay triangulation, and each triplet of Voronoi cells $\mathrm{Vor}(\mathbf{x}_i), \mathrm{Vor}(\mathbf{x}_j), \mathrm{Vor}(\mathbf{x}_k)$ that has a non-empty intersection defines a triangle $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$. The Delaunay triangulation has several interesting geometric properties, such as maximizing the smallest angle (see, e.g., [6]), and is used by a wide class of mesh generation algorithms.

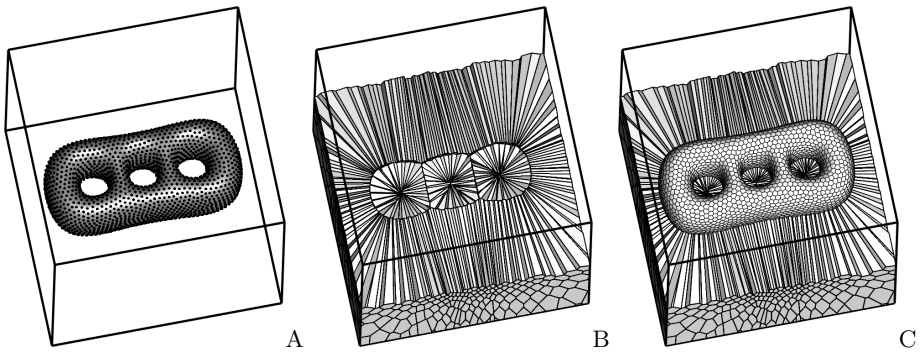


Fig. 1. A: a surface $\Omega \subset \mathbb{R}^3$ and a set of points $\mathbf{X} \in \mathbb{R}^3$ (picked on Ω in this example but it is not mandatory, they could be anywhere in $\mathbb{R}^3$). B: The 3d Voronoi diagram $\mathrm{Vor}(\mathbf{X})$ in $\mathbb{R}^3$ (cross-section). C: the 3d Voronoi diagram $\mathrm{Vor}(\mathbf{X})$ with the restricted Voronoi diagram $\mathrm{Vor}(\mathbf{X})|_\Omega$ superimposed. Each restricted Voronoi cell Ω_i is the intersections between the 3d Voronoi cell $\mathrm{Vor}(\mathbf{x}_i)$ and the surface Ω (a closer view is shown in Figure 2).

Restricted Voronoi Diagram

We quickly introduce the notion of Restricted Voronoi Diagram, that plays a central role in several mesh generation algorithms (see e.g. [12] and the references herein). We consider a domain $\Omega \subset \mathbb{R}^d$. In our specific case, Ω will be a surface embedded in $\mathbb{R}^d$ (but the definitions below apply to any subset). The Voronoi diagram of $\mathbf{X}$ restricted to Ω, denoted by $\mathrm{Vor}(\mathbf{X})|_\Omega$, is the collection of subsets Ω_i of Ω defined by :

$$\mathrm{Vor}(\mathbf{X})|_\Omega = \{\Omega_i\} \quad ; \quad \Omega_i = \mathrm{Vor}(\mathbf{x}_i) \cap \Omega = \{\mathbf{x} \in \Omega | d(\mathbf{x}, \mathbf{x}_i) \leq d(\mathbf{x}, \mathbf{x}_j) \forall j\} .$$

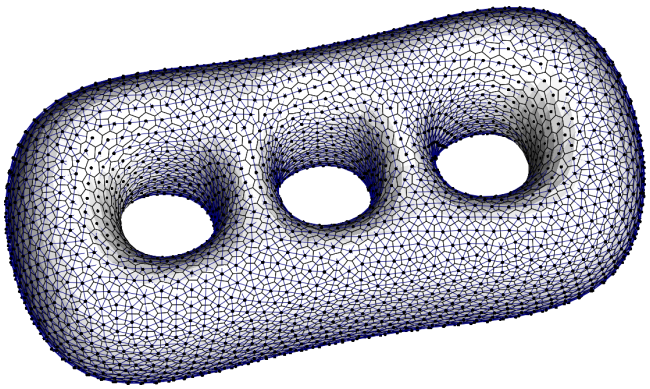
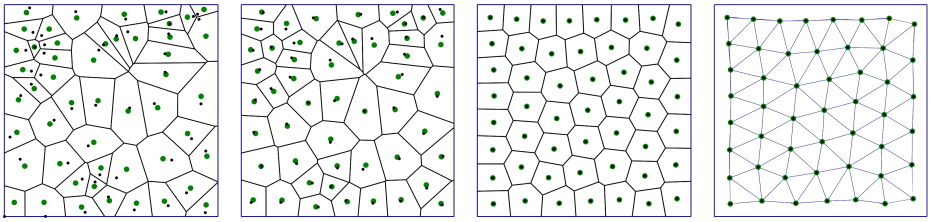


Fig. 2. A Restricted Voronoi Diagram with the associated Restricted Delaunay Triangulation superimposed.



A B C D

Fig. 3. A: Voronoi diagram of a random pointset $\mathbf{X}$ (black dots), and centroids of the Voronoi cells (green dots). B: configuration after one iteration of Lloyd relaxation. C: after 100 iterations. D: Delaunay triangulation.

The subset Ω_i is called the restricted Voronoi cell of $\mathbf{x}_i$ (see Figure 1). Similarly to the standard (unrestricted) case, one can define the dual of a restricted Voronoi diagram, called the Restricted Delaunay Triangulation (RDT for short). Each triangle of the RDT corresponds to three Restricted Voronoi Cells that have a non-empty intersection (see Figure 2).

As a direct consequence of their definition, the restricted Voronoi cells can be also given by :

$$\Omega_i = \Omega \cap \left(\bigcap_j \Pi^+(i,j) \right)$$

where $\Pi^+(i,j) = \{\mathbf{x} | d(\mathbf{x},\mathbf{x}_i) \leq d(\mathbf{x},\mathbf{x}_j)\}$ denotes the halfspace bounded by the bisector of the segment $[\mathbf{x}_i,\mathbf{x}_j]$ that contains $\mathbf{x}_i$. In other words, the Voronoi cell Ω_i can be obtained by starting from the entire space Ω and iteratively clipping it with the bisectors $\Pi^+(i,j)$ defined by all other sites $\mathbf{x}_j$. Directly applying this iterative clipping procedure clearly has no practical value, since the resulting algorithm has superquadratic complexity. For this reason, most existing algorithms that compute Voronoi diagrams are based on different considerations, such as properties of the Delaunay triangulation (see [6] for a survey). However, computing a Delaunay triangulation has time and space complexity proportional to $d!$, where d denotes the dimension. While the $d!$ constant is small for $d = 2$ or $d = 3$, in our context of anisotropic mesh generation, we use a higher dimensional space ($d = 6$ typically, see below), for which this $d!$ time/space requirement is prohibitive. In Section 4, we will show a simple theorem that makes the iterative clipping point of view practical in our specific context, and that avoids the $d!$ factor.

Centroidal Voronoi Tessellation

A Centroidal Voronoi Tessellation (CVT for short) is a Voronoi Diagram that satisfies the following property :

$$\forall i, \mathbf{x}_i = \mathbf{g}_i$$

where $\mathbf{g}_i$ denotes the centroid of the Voronoi cell Ω_i. A CVT can be computed by the so-called *Lloyd relaxation* [31], that iteratively moves each vertex $\mathbf{x}_i$ to the centroid $\mathbf{g}_i$. Lloyd relaxation generates a regular sampling[18, 17], from which a Delaunay triangulation with well-shaped isotropic elements can be extracted [19, 21, 3]. An example of CVT and its dual Delaunay triangulation is shown in Figure 3. In the case of surface meshing, it is possible to generalize this definition by using a geodesic Voronoi diagram over the surface Ω [38], where the Euclidean distance d that appears in the definition of the Voronoi cells is replaced with the geodesic distance measured on the surface Ω. However, computing a geodesic Voronoi diagram is costly, and becomes prohibitive in our context, that require multiple iterations. The notion of Restricted Centroidal Voronoi Tessellation[20] can be used instead. The idea consists in letting the points $\mathbf{x}_i$ move in the entire ambient space $\mathbb{R}^d$, and replace the geodesic Voronoi diagram on Ω with the restricted Voronoi diagram $\mathrm{Vor}\mathbf{X}|_\Omega$ introduced above. The algorithm has the same structure as Lloyd relaxation in 3d (a 3d Voronoi diagram is computed). The only difference is that the centroids $\mathbf{g}_i$ are computed from the *restricted* Voronoi cells $\Omega_i = \mathrm{Vor}(\mathbf{x}_i) \cap \Omega$ instead of the full Voronoi cells $\mathrm{Vor}(\mathbf{x}_i)$. Since the centroids are not necessarily on Ω, it may be required to reproject them onto Ω after each iteration (hence defining a Constrained CVT). With an efficient algorithm to compute the Restricted Voronoi Diagram, Restricted and Constrained CVT can be used for isotropic surface remeshing [44]. We will show further how Restricted CVT can be implemented in higher-dimensional space and used to optimize all the vertices of an anisotropic mesh simultaneously.

2.2 Anisotropy

Definition

We consider now that a metric $\mathbf{G}(.)$ is defined over the domain $\mathbb{R}^d$. In other words, at a given point $\mathbf{x} \in \mathbb{R}^d$, the dot product between two vectors $\mathbf{v}$ and $\mathbf{w}$ is given by :

$$< \mathbf{v}, \mathbf{w} >_{\mathbf{G}(\mathbf{x})} = \mathbf{v}^t \mathbf{G}(\mathbf{x}) \mathbf{w}.$$

In practice, the metric $\mathbf{G}(\mathbf{x})$ can be represented by a symmetric $d \times d$ matrix (with coefficients that possibly vary in function of the coordinates of $\mathbf{x}$).

Given a metric $\mathbf{G}$ and an open curve $\mathcal{C} \subset \mathbb{R}^d$, it is possible to define the length of $\mathcal{C}$ relative to $\mathbf{G}$ as :

$$l_{\mathbf{G}}(\mathcal{C}) = \int_{t=0}^{1} \sqrt{\mathbf{v}(t)^t \mathbf{G}(\mathbf{x}(t)) \mathbf{v}(t)} dt$$

where $\mathbf{x}(.)$ denotes a parameterization of $\mathcal{C}$ and $\mathbf{v}(t) = \partial \mathbf{x}(t)/\partial t$ the tangent vector. Then, the anisotropic distance $d_{\mathbf{G}}(\mathbf{x}, \mathbf{y})$ between two points $\mathbf{x}$ and $\mathbf{y}$

is defined as the length of the (possibly non-unique) shortest curve $\mathcal{C}$ that connects $\mathbf{x}$ and $\mathbf{y}$:

$$d_{\mathbf{G}}(\mathbf{x},\mathbf{y}) = \min_{\{\mathcal{C}|\mathcal{C}(0)=\mathbf{x},\mathcal{C}(1)=\mathbf{y}\}} l_{\mathbf{G}}(\mathcal{C}).$$

Since $d_{\mathbf{G}}(.,.)$ is very difficult to compute in practice, some methods also use the (simpler to compute) anisotropic length of the segment $(\mathbf{x},\mathbf{y})$:

$$\bar{d}_{\mathbf{G}}(\mathbf{x},\mathbf{y}) = l_{\mathbf{G}}([\mathbf{x},\mathbf{y}])$$

Relation with the Theory of Approximation and Applications

Anisotropy is related with the theory of approximation[15, 39], that studies how to best approximate a function with a given budget of points. Under some circumstances, it is possible to characterize the anisotropy of the optimal mesh, and design algorithms to optimize the approximation properties of a mesh. Some methods based on an anisotropic version of point insertion in Delaunay triangulation were proposed and successfully applied to many practical cases [8, 9, 7, 16]. The continuous mesh framework [32, 33] is based on an equivalence between some set of ellipses circumscribed to the triangles of a mesh and a continuous metric. In this setting, a metric is considered as the "limit case" of a mesh with infinitely small anisotropic triangles. In practice, this consideration results in highly efficient anisotropic mesh adaptation algorithms. The connections between anisotropic meshes and approximation theory were studied, not only for piecewise linear functions, but also for higher-order finite elements [34, 35, 13]. This theoretical analysis leads to an efficient greedy bissection algorithm that generates optimal meshes. In the specific case of computing a polygonal approximation of a given shape, some metrics were proposed[14, 42], together with discrete (per-facet) Lloyd-like optimization algorithms.

Anisotropic Generalizations of Voronoi Diagrams

Our goal is to experiment Lloyd relaxation in the anisotropic setting, with a continuous formulation, independent on the shape of the original triangles. Thus, we need a definition of Voronoi diagrams that accounts for anisotropy. The most general setting is given by Riemann Voronoi diagrams [28], that replace the distance $d(\mathbf{x},\mathbf{y})$ with the anisotropic distance $d_{\mathbf{G}}(\mathbf{x},\mathbf{y})$ defined above. Some theoretical results are known, however, a practical implementation in 3d is still beyond reach. For this reason, two simplifications are used :

$$\begin{aligned}
\mathrm{Vor}_{\mathrm{Labelle}}(\mathbf{x}_i) &= \{\mathbf{y}|d_{\mathbf{x}_i}(\mathbf{x}_i,\mathbf{y}) \le d_{\mathbf{x}_j}(\mathbf{x}_j,\mathbf{y})\forall j\} \\
\mathrm{Vor}_{\mathrm{Du}}(\mathbf{x}_i) &= \{\mathbf{y}|d_{\mathbf{y}}(\mathbf{x}_i,\mathbf{y}) \le d_{\mathbf{y}}(\mathbf{x}_j,\mathbf{y})\forall j\} \\
\text{where: } d_{\mathbf{x}}(\mathbf{y},\mathbf{z}) &= \sqrt{(\mathbf{z}-\mathbf{y})^t\mathbf{G}(\mathbf{x})(\mathbf{z}-\mathbf{y})}
\end{aligned}$$

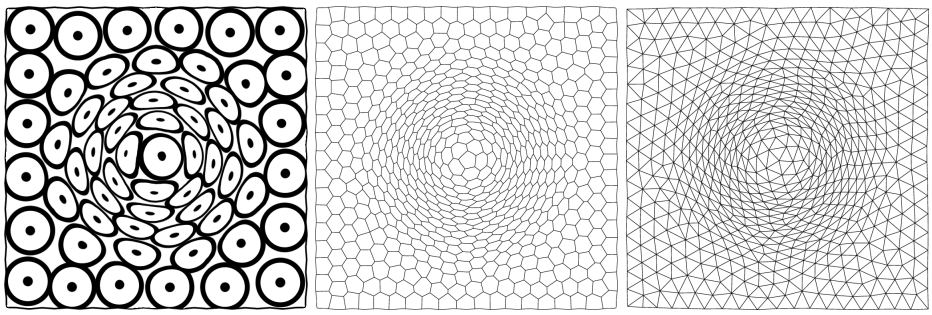


Fig. 4. Anisotropic Voronoi Diagram and Delaunay Triangulation. A: prescribed anisotropy field; B: anisotropic Voronoi diagram and C: Delaunay triangulation.

The first definition $\mathrm{Vor}_{\mathrm{Labelle}}$ [27] defines an object that is easier to analyze theoretically. The bisectors are quadratic surfaces, known in closed form, and a provably correct Delaunay refinement algorithm can be defined. The so-defined anisotropic Voronoi diagram may be also thought of as the projection of a higher-dimensional power diagram [5]. The second definition $\mathrm{Vor}_{\mathrm{Du}}$ [22] is best suited to a practical implementation of Lloyd relaxation. In particular, the midpoints can be determined. An approximation of the Voronoi cells and their centroids can be reasonably easily computed for 2d domains and parametric surfaces. However, generalizing the algorithms to arbitrary 3d surfaces may be difficult.

In what follows, we propose a general method to apply Lloyd relaxation to a surface embedded in a higher dimensional space. For a given arbitrary metric $\mathbf{G}$ (e.g. for solution-based adaptivity), one may use the embedding defined in [5]. In this paper, though the algorithm may be applicable to more general settings, we will limit ourselves to a simpler embedding explained in the next section, well suited to generate curvature-adapted surfacic meshes.

3 Trading Anisotropy for Additional Dimensions

3.1 A simple example

We consider the 2d metric $\mathbf{G}$ depicted in Figure 4-A. The deformed circles correspond to the points equidistant to the black dots in terms of the distance $d_{\mathbf{G}}$ defined by the metric. Our goal is to generate an anisotropic Voronoi diagram governed by this metric (Figure 4-B) and deduce an anisotropic mesh from its dual (Figure 4-C). For this simple example, one can see that Figures 4-A,B,C can be considered as the surfaces in Figures 5-A,B,C (next page) "seen from above". In other words, by embedding the flat 2d domain as a curved surface in 3d, one can recast the anisotropic meshing problem as the isotropic meshing

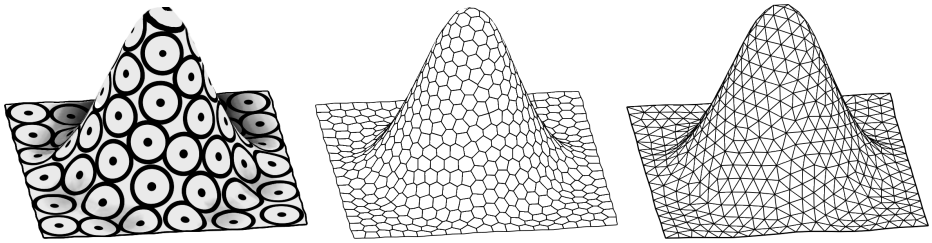


Fig. 5. A 3d surface that generates the prescribed anisotropy in Figure 4 when "seen from above" (in general, more dimensions are needed).

of a surface embedded in higher-dimensional space. Isotropic surface meshing can be computed as a restricted CVT (see the previous section).

In general (i.e., for an arbitrary metric $\mathbf{G}$), a higher-dimensional space will be needed [37]. In the case of 3d surface or volume meshing, using Labelle and Shewchuk's discrete definition of anisotropic Voronoi diagrams with Boissonnat et.al's characterization[5], one needs dimension $d = 10$ (it uses a power diagram of dimension 9 that can be seen as the projection of a Voronoi diagram of dimension 10). For this reason, we propose an algorithm to compute a restricted CVT of a surface embedded in higher-dimensional space. Before presenting the algorithm, we present a simple embedding, well suited to generate curvature-adapted anisotropic mesh.

3.2 The Gauss map

In this section, we present a 6d embedding, that uses both positions and normals. This embedding was used to generate anisotropic quad meshes by contouring in 6d[10], to classify the features of meshes[26] or to compute anisotropic quadrangulations[25]. It has interesting connections with surface normal approximation[11].

Given a surface $\Omega \subset \mathbb{R}^3$, the Gauss map is the application that maps Ω onto the unit sphere S^2, defined by $\mathbf{x} \in \Omega \mapsto \mathbf{n}(\mathbf{x}) \in S^2$, where $\mathbf{n}(\mathbf{x})$ denotes the unit normal vector at $\mathbf{x}$. The Gauss map plays a fundamental role in the definition of curvature. For instance, Gauss curvature is defined as the Jacobian of the Gauss map. In other words, Gauss curvature is a measure of how areas are locally distorted by the Gauss map. As a consequence, a uniform sampling of the unit sphere results in a curvature-adapted mesh when transformed back onto Ω. As shown in Figure 6, the tips of the fingers cover a wide area of the unit sphere when transformed by the Gauss map, and therefore have a high density of vertices in the sampling (region a in the figure). In contrast, the tubular regions of the fingers (region b) just cover an

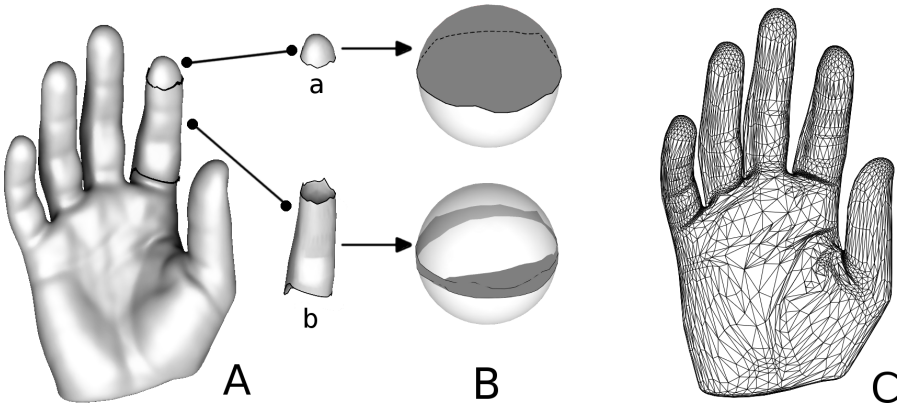


Fig. 6. A: a surface with two highlighted zones (a and b). B: the Gauss map of zones a and b (dark gray). Zone a covers half of the sphere, whereas zone b only covers an annulus around the equator of the sphere. C: an isotropic meshing in Gauss space mapped back into Euclidean space is a curvature-adapted anisotropic mesh. Zone a contains more vertices than zone b since it covers a larger space in the Gauss map.

annulus around the equator of the unit sphere. A uniform sampling of this annulus, when transformed back onto Ω, results in triangles that are elongated along the axis of the tubular region. Unfortunately, for general surfaces (i.e. non-convex), the Gauss map is not bijective. For this reason, we propose instead to use the embedding $\Phi : \Omega \rightarrow I\!R^6$ defined by:

$$\Phi(\mathbf{x}) = \begin{bmatrix} x \\ y \\ z \\ sn_x \\ sn_y \\ sn_z \end{bmatrix}$$

where n_x, n_y, n_z denotes the normal to Ω at $\mathbf{x}$, and where $s \in (0, +\infty)$ is a constant, user-defined factor that specifies the desired amount of anisotropy. For a small value of s, the normal components are ignored, and an isotropic mesh is obtained. For high values of s, more importance is given to the Gauss map components, and a highly anisotropic mesh is obtained. In what follows, we will use this embedding. Other embeddings can account for different anisotropies. They will be discussed in conclusion.

We now consider a surface Ω given as a triangle mesh. One can evaluate the normals at the vertices of Ω, and embed Ω into $I\!R^6$ with Φ. At this point, a possibility would be to use Algorithm 1 below. Step 1 is detailed further in Section 4.2. Step 2 may use an implementation of Delaunay triangulation in arbitrary dimension, available in QHull[4] or as an experimental package of CGAL[1]. Step 3 could use two nested loops, one over the m facets of Ω and

(1) $\mathbf{X} \leftarrow$ initial random sampling of Ω in $\mathbb{R}^6$
while *minimum not reached* **do**
  (2) Compute the Voronoi diagram $Vor(\mathbf{X})$ in $\mathbb{R}^6$
  (3) Compute the restricted Voronoi diagram $Vor(\mathbf{X})|_\Omega$
  foreach i **do**
    Compute $\mathbf{g}_i = \int_{\Omega_i} \mathbf{x} d\mathbf{x} / \int_{\Omega_i} d\mathbf{x}$
    $\mathbf{x}_i \leftarrow \mathbf{g}_i$
  end
end

Algorithm 1: *Naive implementation of anisotropic CVT in* $\mathbb{R}^6$

one over the n Voronoi cells of $\mathrm{Vor}(\mathbf{X})$. The $O(mn)$ complexity of these two nested loops can be replaced with $O(m+n)$, by a propagation along both the facet graph of Ω and the cell graph of $\mathrm{Vor}(\mathbf{X})$[44]. This propagation generates all couples (f,i) such that the Voronoi cell $\mathrm{Vor}(\mathbf{x}_i)$ has a non-empty intersection with the facet f. From the (f,i) couples, the restricted Voronoi cells Ω_i are obtained by accumulating the contributions of all facets $\mathrm{Vor}(\mathbf{x}_i) \cap f$.

Our experiments with this approach, for $d = 6$ and $n = 5000$ vertices showed that 100 iterations of Lloyd relaxation requires more than a day, even with a high-end 16Gb computer, that exhausts its physical memory and starts swapping (due to the $d!$ cost in memory). For this reason, we propose an alternative in the next section, that not only avoids the $d!$ cost, but also can be easily parallelized. Our algorithm scales-up well and can optimize an anisotropic mesh with one million vertices in minutes on an 8 cores machine.

4 CVT in high-dimensional space with Vorpaline

The basic operation in the step 3 of Algorithm 1 above consists in clipping a facet of Ω with a Voronoi cell of $\mathrm{Vor}(\mathbf{X})$. A Voronoi cell is a convex polytope, that can be defined as the intersection of halfspaces :

$$\mathrm{Vor}(\mathbf{x}_i) = \bigcap_{k=1}^{v_i} \Pi^+(i, j_k)$$

where $\Pi^+(i,j)$ denotes the halfspace bounded by the bisector of $(\mathbf{x}_i, \mathbf{x}_j)$ that contains $\mathbf{x}_i$, the j_k's denote the indices of the v_i Voronoi cells that have a facet (of dimension $d-1$) in common with $\mathrm{Vor}(x_i)$. In terms of Delaunay triangulation, they correspond to the indices of the vertices connected to $\mathbf{x}_i$ by a Delaunay edge (and v_i denotes the valence of the vertex i in the Delaunay 1-skeleton). This representation of the Voronoi cells (H-representation) is well suited to the computation of the intersections with the facets of Ω in step 3, that can be efficiently computed by Sutherland & Hodgman's re-entrant clipping[40].

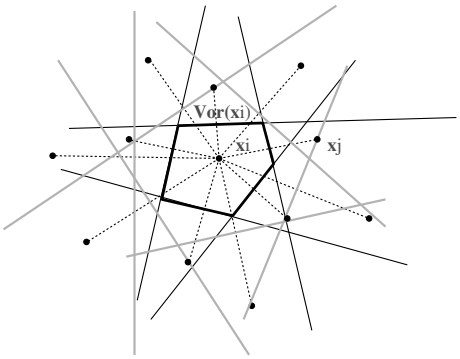


Fig. 7. The Voronoi cell $\mathrm{Vor}(\mathbf{x}_i)$ is the result of clipping the entire space by all the bisectors (continuous lines) of the segments $[\mathbf{x}_i, \mathbf{x}_j]$ (dashed lines). Among the bisectors, some are contributing (black) and some are non-contributing (gray).

Alternatively, as indicated in Section 2.1, the definition of the Voronoi cell $\mathrm{Vor}(\mathbf{x}_i)$ directly gives :

$$\mathrm{Vor}(\mathbf{x}_i) = \bigcap_{j \neq i} \Pi^+(i,j)$$

One can see that all the possible bisectors are used, in contrast with the previous definition of $\mathrm{Vor}(\mathbf{x}_i)$ that only uses the neighbors. As a consequence, we can classify the bisectors $\Pi^+(i,j)$ into two sets (see Figure 7).

The bisector $\Pi^+(i,j)$ is said to be :

- *non-contributing* if $\mathrm{Vor}(\mathbf{x}_i) \subset \Pi^+(i,j)$;
- *contributing* otherwise.

In other words, clipping a Voronoi cell by a non-contributing bisector does not change the result. Therefore, the Voronoi cell can be computed by intersecting a superset of the contributing bisectors. We now propose an algorithm that provably finds all the contributing bisectors of a Voronoi cell.

4.1 Radius of Security

We now suppose that the vertices $\mathbf{X}$ are organized in a geometric search data structure, such that for any $\mathbf{x}_i$ one can efficiently compute the list of nearest neighbors $\mathbf{x}_j$ sorted by increasing distance to $\mathbf{x}_i$. We use the ANN[3] library [36]. The algorithm in ANN has linear space-time complexity for construction in function of both the number of points and the dimension, and $O(d \log(n))$ complexity for nearest neighbors queries.

[3]Approximate Nearest Neighbors. Note: it returns exactly the nearest neighbors when the parameter ϵ is set to 0.

Let $\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \ldots \mathbf{x}_{j_{n-1}}$ denote the vertices sorted by increasing distance from $\mathbf{x}_i$. Let V_k denote the intersection of the k first bisectors and R_k denote its $\mathbf{x}_i$-centered radius :

$$\begin{aligned} V_k(\mathbf{x}_i) &= \bigcap_{l=1}^{k} \Pi^+(i, j_l) \\ R_k &= \max\{d(\mathbf{x}_i, \mathbf{x}) | \mathbf{x} \in V_k(\mathbf{x}_i)\}. \end{aligned}$$

We have then the following simple theorem :

Theorem 1. *For all j such that $d(\mathbf{x}_i, \mathbf{x}_j) > 2R_k$, the bisector $\Pi^+(i, j)$ is non-contributing, i.e. $V_k(\mathbf{x}_i) \subset \Pi^+(i, j)$.*

Proof. Consider $\mathbf{x} \in V_k(\mathbf{x}_i)$ and $\mathbf{x}_j$ such that $d(\mathbf{x}_i, \mathbf{x}_j) > 2R_k$.
By definition of R_k, $d(\mathbf{x}, \mathbf{x}_i) < R_k$.
We have $d(\mathbf{x}_i, \mathbf{x}) + d(\mathbf{x}, \mathbf{x}_j) > (\mathbf{x}_i, \mathbf{x}_j)$ (triangular inequality)
and $d(\mathbf{x}_i, \mathbf{x}_j) > 2R_k$,
therefore $d(\mathbf{x}, \mathbf{x}_j) > R_k > d(\mathbf{x}, \mathbf{x}_i)$ and $\mathbf{x} \in \Pi^+(i, j)$
■

As a direct consequence of Theorem 1 :

$$d(\mathbf{x}_i, \mathbf{x}_{j_{k+1}}) > 2R_k \implies V_k = \mathrm{Vor}(\mathbf{x}_i).$$

We call *radius of security* the first value of R_k that satisfies this condition. Note that this theorem does not have a practical value in the case of the (unrestricted) Voronoi diagram, since some cells are unbounded and have infinite R_k (therefore, for an infinite cell, all the bisectors are considered), leading to prohibitive computation time. However, it is clear that the theorem applies to the restricted Voronoi diagram $\mathrm{Vor}(\mathbf{X})|_\Omega$. It also gives a practical way of computing $\mathrm{Vor}(\mathbf{x}_i) \cap f$ for a bounded set f (e.g., a facet of Ω). This configuration is detailed in Algorithm 2, that we will use to compute restricted CVT in $\mathbb{R}^d$.

Data: a facet $f = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ of Ω and a set of points $\mathbf{X} \in \mathbb{R}^d$
Result: $V = \mathrm{Vor}(\mathbf{x}_i) \cap f$

$V \leftarrow f$
$R \leftarrow \max\{d(\mathbf{x}_i, \mathbf{p}_1), d(\mathbf{x}_i, \mathbf{p}_2), d(\mathbf{x}_i, \mathbf{p}_3)\}$
$k \leftarrow 1$
while $d(\mathbf{x}_i, \mathbf{x}_{j_k}) < 2R$ **and** $k < n$ **do**
  $V \leftarrow V \cap \Pi^+(i, j_k)$
  $R \leftarrow \max\{d(\mathbf{x}, \mathbf{x}_i) | \mathbf{x} \in V\}$
  $k \leftarrow k + 1$
end

Algorithm 2: *Clipping a facet with a Voronoi cell in $\mathbb{R}^d$.*

Data: a surface Ω, the desired number of points n and anisotropy s
Result: an anisotropic sampling $\mathbf{X}$ of Ω

(1) embed Ω in $\mathbb{R}^6$ using Φ_s (see Section 3.2)
(2) compute an initial random sampling $\mathbf{X}$ in $\mathbb{R}^6$ (see Algorithm 4 below)
for $k = 1$ **to** *nbiter* **do**
  Update the ANN data structure with $\mathbf{X}$
  $\mathbf{g}_{1\ldots n} \leftarrow \mathbf{0}$; $m_{1\ldots n} \leftarrow 0$
  (3) **foreach** $\{(t, \mathbf{x}_i) | t \cap Vor(\mathbf{x}_i) \neq \varnothing\}$ **do**
    (4) $V \leftarrow t \cap \mathrm{Vor}(\mathbf{x}_i)$ (Algorithm 2)
    (5) $m \leftarrow \mathrm{mass}(V)$; $m_i \leftarrow m_i + m$; $\mathbf{g}_i \leftarrow \mathbf{g}_i + m \times \mathrm{centroid}(V)$
  end
  for $i = 1$ **to** n **do**
    (6) $\mathbf{x}_i \leftarrow 1/m_i \times \mathbf{g}_i$
  end
end

Algorithm 3: *Anisotropic Lloyd Relaxation in* $\mathbb{R}^d$

4.2 Implementation

Anisotropic Lloyd relaxation is detailed in Algorithm 3. Step (3) is based on a recursive propagation on the facet graph of Ω and the cell graph of $\mathrm{Vor}(\mathbf{X})$ detailed in [44]. The mass (measure in 6d) and centroid of each clipped facet is accumulated in intermediary variables, from which the centroids are computed (5),(6). Finally, the Restricted Delaunay triangulation is computed as the dual of the Restricted Voronoi Diagram, (or the dual of its connected components[29] to avoid some degenerate configurations). Note that convergence can be significantly accelerated by replacing Lloyd relaxation with quasi-Newton optimization[30]. The latter was used in the results shown in this paper.

Initialization is computed as in Algorithm 4. In step (1), the area of a triangle in $\mathbb{R}^d$ can be computed with Heron's formula $A = \sqrt{s(s-a)(s-b)(s-c)}$

Data: a surface Ω embedded in $\mathbb{R}^d$ and the desired number of points n
Result: an initial sampling $\mathbf{X}$ of Ω

Generate a vector (s_i) of n random numbers
Sort (s_i)
(1) **foreach** $f \in \Omega$, compute the normalized area $a_f = \mathrm{Area}(f) / \sum_f \mathrm{Area}(f)$
$f \leftarrow 1$; $A \leftarrow 0$
for $i = 1$ **to** n **do**
  while $(s_i > A)$ $f \leftarrow f + 1$; $A \leftarrow A + a_f$
  (2) $\mathbf{x}_i \leftarrow$ random point in f
end

Algorithm 4: *Computing the initial random sampling in* $\mathbb{R}^d$

where a, b, c denote the length of the three edges and $s = a + b + c$. Step (2) uses Turk's method to generate a random point in a triangle[41].

Due to space constraints, the implementation is not fully detailed here (but will be in a future publication). We outline two important aspects :

- **parallelism :** the computation of a specific Voronoi cell is independent on the others. Thus, we first partition the input surface into N parts (typically N = number of cores) with METIS[23]. We use one thread per part to compute the masses and centroids of the restricted Voronoi cells in each part (Alg. 3 (3)-(5)). Finally, each point $\mathbf{x}_i$ is updated by accumulating the contribution of each part to the centroid of its restricted Voronoi cell.
- **genericity :** Our implementation is based on C++ templates, in the spirit of CGAL[1] library. With a "Kernel" class, we parameterize the algorithm in terms of dimension d (that can be known at compile time or dynamic) and exact geometric predicates (that use arithmetic filters).

5 Results

We have experimented our implementation on several databases, comprising CAD data and scanned meshes. Some representative results are shown in Figure 8 and 9. To compute the embedding, we used the normal to the CAD surface whenever available, or estimated it at the vertices of the input triangle mesh for scanned meshes. Input meshes have between 30K and 4M triangles. We have generated anisotropic meshes with up to 1M vertices. Computation time ranges between a few seconds and 10 minutes for 100 iterations of Newton optimization on a 8-cores PC.

6 Discussion, Conclusions and Future Work

We proposed an efficient methodology to implement Lloyd relaxation in high dimensional space, and applications to anisotropic surface remeshing. A limitation of our method is that the Restricted Delaunay Triangulation may contain flipped triangles, in particular in zones where the direction of the anisotropy varies too quickly. This difficulty is pointed-out in [27]. This may be fixed with some post-processing (e.g. Delaunay refinement). Another limitation concerns creases and sharp features, that are oversampled in the direction orthogonal to the crease. We think that this issue can be solved by explicitely sampling the creases, then using constrained optimization combined with "protecting balls" [12]. This means computing a power diagram, that can be obtained from a $d+1$ Voronoi diagram in which our "security radius" theorem can be used. We also mention that our algorithmic framework can be easily adapted to tetrahedral meshes embedded in nd, by replacing the $3d$ tetrahedral clipping routine in [43] by our nd Vorpaline method. This may be used to generate $3d$ anisotropic tetrahedral meshes.

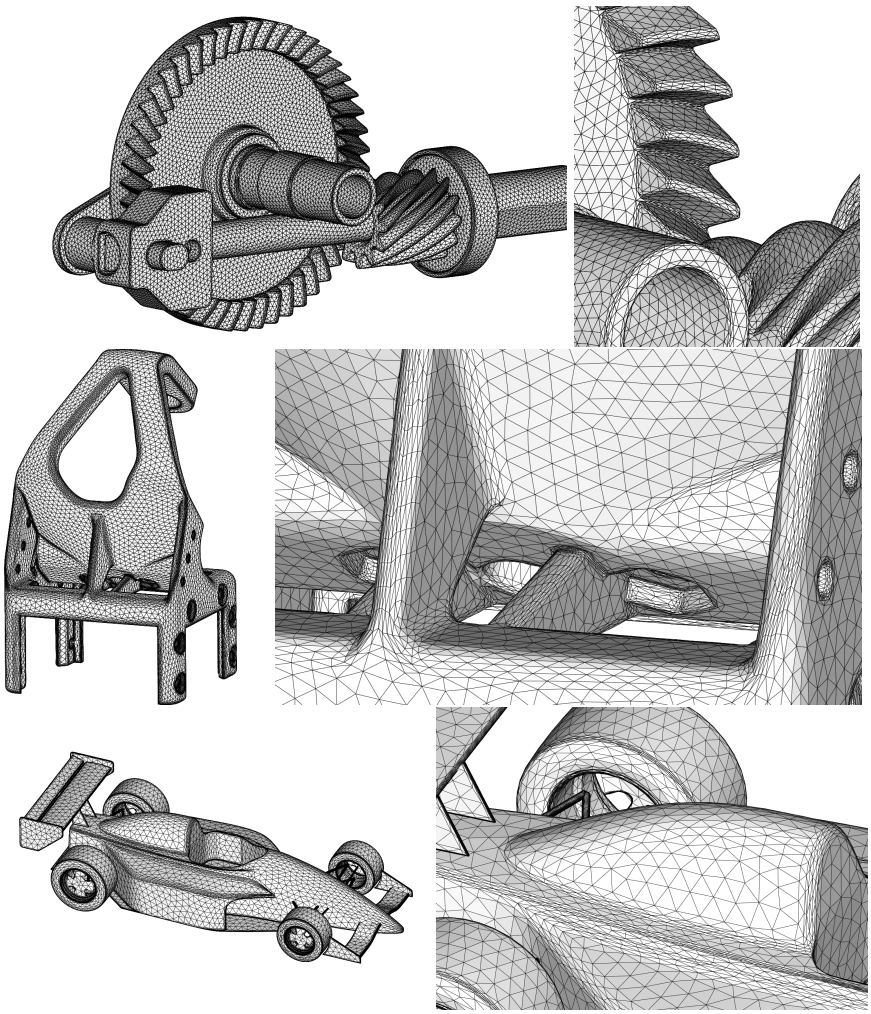


Fig. 8. Anisotropic meshes - mechanical parts.

In a more general perspective, this work shows the practicality of geometric computing in high-dimensional space, and in particular the possibility of computing a restricted Voronoi diagram in $\mathbb{R}^d$. Besides the $6d$ (Euclidean × Gauss-map) embedding experimented here, we plan to experiment with more general metric fields and the $10d$ embedding [5].

We also conducted experiments with a much larger number of coordinates, obtained by the Finite Element discretization of the eigenfunctions of a symmetric operator. This allows to implement CVT for kernel-based distances. In our preliminary results, with our generic implementation, we obtained reasonable timings (minutes) for remeshing in dimension spaces as large as $d = 5000$.

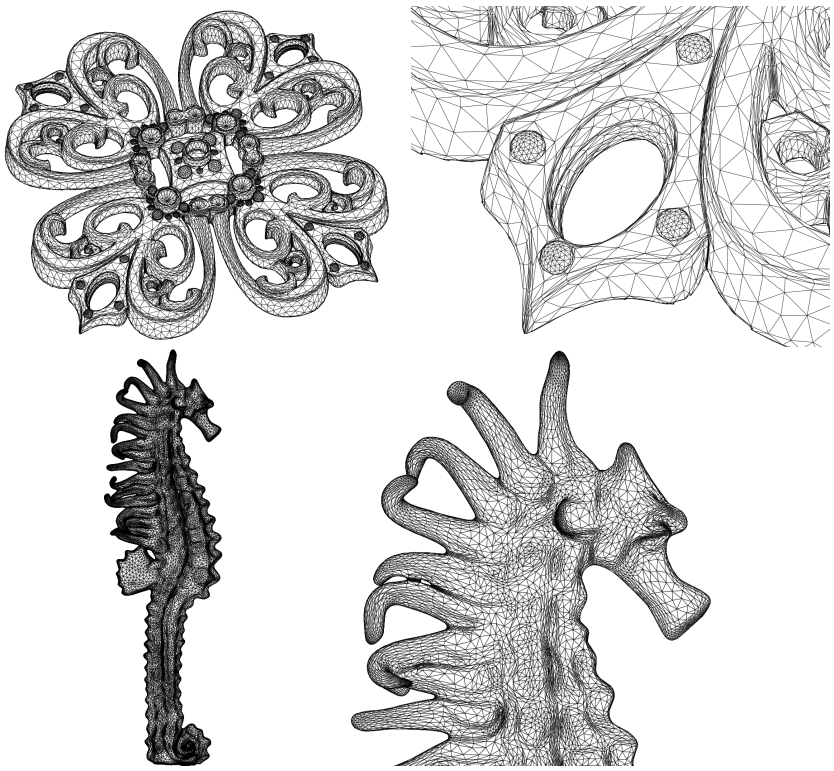


Fig. 9. Anisotropic meshes - scanned data.

Acknowledgments

This work is partly supported by the European Research Council (GOODSHAPE, ERC-StG-205693), the ANR (MORPHO) and NSF (grant IIS 11109555). The authors wish to thank Wenping Wang for many discussions. The meshes shown in the paper are courtesy of Jean-François Remacle, Christophe Geuzaine, Nicolas Saugnier and AimAtShape. The F1 car is from www.opencascade.org.

References

1. CGAL. http://www.cgal.org.
2. Frédéric Alauzet and Adrien Loseille. High-order sonic boom modeling based on adaptive methods. *J. Comput. Physics*, 229(3):561–593, 2010.
3. P. Alliez, E. Colin de Verdière, O. Devillers, and M. Isenburg. Centroidal Voronoi diagrams for isotropic remeshing. *Graphical Models*, 67(3):204–231, 2003.
4. C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

5. J.-D. Boissonnat, C. Wormser, and M. Yvinec. Anisotropic Diagrams: Labelle Shewchuk approach revisited. *Theo. Comp. Science*, (408):163–173, 2008.
6. Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic Geometry*. Cambridge University Press, 1998.
7. H. Borouchaki and P.-L. George. *Delaunay Triangulation and Meshing: Application to Finite Elements*. Hermès, 1998.
8. H. Borouchaki, P.-L. George, F. Hecht, P. Laug, and E. Saltel. Delaunay mesh generation governed by metric specifications. part 1: Algorithms. *Finite Elements in Analysis and Design*, 25:61–83, 1997.
9. H. Borouchaki, P.-L. George, F. Hecht, P. Laug, and E. Saltel. Delaunay mesh generation governed by metric specifications. part 2: Application examples. *Finite Elements in Analysis and Design*, 25:85–109, 1997.
10. Guillermo D. Canas and Steven J. Gortler. Surface remeshing in arbitrary codimensions. *Vis. Comput.*, 22(9):885–895, September 2006.
11. Guillermo D. Canas and Steven J. Gortler. Shape operator metric for surface normal approximation. In *International Meshing Roundtable Conference Proceedings*, Salt Lake City, November 2009.
12. Siu-Wing Cheng, Tamal K. Dey, and Joshua A. Levine. A practical delaunay meshing algorithm for alarge class of domains. In *16th International Meshing Roundtable conf. proc.*, pages 477–494, 2007.
13. A. Cohen, N. Dyn, F. Hecht, and J.-M. Mirebeau. Adaptive multiresolution analysis based on anisotropic triangulations. *Math. Comput.*, 81(278), 2012.
14. David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23:905–914, 2004.
15. E. d'Azevedo. Optimal triangular mesh generation by coordinate transformation. Technical report, University of Waterloo, 1989.
16. Cécile Dobrzynski and Pascal J. Frey. Anisotropic Delaunay mesh adaptation for unsteady simulations. In *Proceedings of the 17th international Meshing Roundtable*, pages 177–194, États-Unis, 2008.
17. Qiang Du, Maria Emelianenko, and Lili Ju. Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. *SIAM Journal on Numerical Analysis*, 44(1):102–119, 2006.
18. Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.
19. Qiang Du and Max Gunzburger. Grid generation and optimization based on centroidal Voronoi tessellations. *Applied Mathematics and Computation*, 133(2-3):591–607, 2002.
20. Qiang Du, Max. D. Gunzburger, and Lili Ju. Constrained centroidal Voronoi tesselations for surfaces. *SIAM Journal on Scientific Computing*, 24(5):1488–1506, 2003.
21. Qiang Du and Desheng Wang. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *International Journal for Numerical Methods in Engineering*, 56:1355–1373, 2003.
22. Qiang Du and Desheng Wang. Anisotropic centroidal Voronoi tessellations and their applications. *SIAM Journal on Scientific Computing*, 26(3):737–761, 2005.
23. George Karypis and Vipin Kumar. Metis - unstructured graph partitioning and sparse matrix ordering system, version 2.0. Technical report, 1995.
24. William Kleb. Nasa technical brief - advanced computational fluid dynamics and mesh generation, 2009. `http://www.techbriefs.com/component/content/article/5075`.

25. Denis Kovacs, Ashish Myles, and Denis Zorin. Anisotropic quadrangulation. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, SPM '10, pages 137–146, New York, NY, USA, 2010. ACM.
26. Yu kun Lai, Qian yi Zhou, Shi min Hu, Johannes Wallner, and Helmut Pottmann. Robust feature classification and editing. *IEEE Trans. Visualization and Computer Graphics*, 2007.
27. F. Labelle and J.-R. Shewchuk. Anisotropic Voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 191–200, 2003.
28. Greg Leibon and David Letscher. Delaunay triangulations and voronoi diagrams for riemannian manifolds. In *SCG conf. proc.*, pages 341–349. ACM.
29. Bruno Lévy and Yang Liu. Lp Centroidal Voronoi Tesselation and its applications. *ACM Transactions on Graphics*, 29(4), 2010.
30. Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, and Chenglei Yang. On centroidal Voronoi tessellation—energy smoothness and fast computation. *ACM Transactions on Graphics*, 28(4):1–17, 2009.
31. Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.
32. Adrien Loseille and Frédéric Alauzet. Continuous mesh framework part i: Well-posed interpolation error. *SIAM J. Numerical Analysis*, 49(1):38–60, 2011.
33. Adrien Loseille and Frédéric Alauzet. Continuous mesh framework part ii: Validations and applications. *SIAM J. Numerical Analysis*, 49(1):61–86, 2011.
34. Jean-Marie Mirebeau and Albert Cohen. Anisotropic smoothness classes: From finite element approximation to image models. *Journal of Mathematical Imaging and Vision*, 38(1):52–69, 2010.
35. Jean-Marie Mirebeau and Albert Cohen. Greedy bisection generates optimally adapted triangulations. *Math. Comput.*, 81(278), 2012.
36. D. M. Mount and S. Arya. ANN: A library for approximate nearest neighbor searching. In *CGC Workshop on Computational Geometry*, pages 33–40, 1997.
37. J. F. Nash. The imbedding problem for riemannian manifolds. *Annals of Mathematics*, 63:20–63, 1956.
38. Gabriel Peyré and Laurent Cohen. Surface segmentation using geodesic centroidal tesselation. In *2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2004)*, pages 995–1002, 2004.
39. Jonathan Richard Shewchuk. What is a good linear finite element? - interpolation, conditioning, anisotropy, and quality measures. Technical report, In Proc. of the 11th International Meshing Roundtable, 2002.
40. Ivan Sutherland and Gary W. Hodgman. Reentrant polygon clipping. *Communications of the ACM*, 17:32–42, 1974.
41. Greg Turk. Graphics gems. chapter Generating random points in triangles, pages 24–28. Academic Press Professional, Inc., San Diego, CA, USA, 1990.
42. Sébastien Valette, Jean-Marc Chassery, and Rémy Prost. Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):369–381, 2008.
43. D.-M. Yan, W. Wang, B. Lévy, and Y. Liu. Efficient computation of 3d clipped voronoi diagram. In *GMP conf. proc.*, pages 269–282, 2010.
44. Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum*, 28(5):1445–1454, 2009.