Anisotropic Delaunay Meshes of Surfaces

Jean-Daniel Boissonnat INRIA Sophia-Antipolis
Kan-Le Shi Tsinghua University Jane Tournois GeometryFactory
Mariette Yvinec INRIA Sophia-Antipolis

Abstract

Anisotropic simplicial meshes are triangulations with elements elongated along prescribed directions. Anisotropic meshes have been shown to be well suited for interpolation of functions or solving PDEs. They can also significantly enhance the accuracy of a surface representation. Given a surface S endowed with a metric tensor field, we propose a new approach to generate an anisotropic mesh that approximates S with elements shaped according to the metric field. The algorithm relies on the well-established concepts of restricted Delaunay triangulation and Delaunay refinement and comes with theoretical guarantees. The star of each vertex in the output mesh is Delaunay for the metric attached to this vertex. Each facet has a good aspect ratio with respect to the metric specified at any of its vertices. The algorithm is easy to implement. It can mesh various types of surfaces like implicit surfaces, polyhedra or isosurfaces in 3D images. It can handle complicated geometries and topologies, and very anisotropic metric fields.

1 Introduction

Anisotropic simplicial meshes are triangulations with elements elongated along prescribed directions. Anisotropic meshes have been shown to be particularly well suited for interpolation of functions and solving PDE's [DS89, She02, Mir10]. They can also significantly enhance the accuracy of a surface representation if the anisotropy of the mesh conforms to the curvature tensor of the surface [HG99]. In this paper, we consider the problem of sampling and meshing a surface S endowed with a metric tensor field that varies over S. The metric tensor can be chosen arbitrarily provided that it satisfies a continuity condition over the surface S. The metric tensor may, for instance, be the curvature tensor of S or the Hessian matrix of a function defined over S.

1.1 Background

Due to its practical importance, several methods have been proposed in the literature to address the problem of approximating surfaces using anisotropic surface meshes. The case of a *parametric surface* S can be handled by working in the 2-dimensional space U of the parameters of S, using an appropriate metric field to account for the deformation induced by the mapping $U \to S$. The problem is then 2-dimensional [BGH$^+$97, BH96, LTÜ99]. Several methods have also been reported to *remesh a polyhedral surface* according to some specified metric tensor. Alliez et al. [ACSD$^+$03] first estimate the principal direction fields and trace a network of lines of curvature from which they deduce a quad-dominant anisotropic mesh. Jiao et al. [JCNH06] use the quadratic metric tensor

of Heckbert and Garland [HG99] and propose a set of operations to allow anisotropic adaptation of a triangular mesh to static or evolving surfaces. The case of *implicit surfaces* can be handled by first polygonizing the surface in a standard way and then remeshing the obtained surface using one of the anisotropic methods mentioned above. Although this approach is clearly not optimal, very few methods have been proposed to directly generate anisotropic meshes on implicit surfaces. One exception is the work of Azernikov and Fischer [AF05] who propose a grid-based approach to mesh implicit surfaces according to their curvature tensor. The method uses a deformed grid and produces quad-meshes. The anisotropic grid is obtained by deforming a cartesian grid while maintaining the structured topology, which is a limitation when considering complex shapes.

In order to settle more rigorous foundations and to precisely characterize anisotropic triangulations, several authors have introduced anisotropic Voronoi diagrams (AVD). An anisotropic triangulation, called anisotropic Delaunay triangulation (ADT), is then defined as the dual of the AVD. Various AVD and ADT have been proposed. Leibon and Letscher [LL00] introduced Delaunay triangulations and Voronoi diagrams for Riemannian manifolds. This approach requires to compute geodesic paths and intrinsic balls, which may be quite complicated in practice. To overcome this issue, two approaches have been proposed that approximate the geodesic distance between a given site and any point by considering the metric constant and equal to the metric at the site [LS03] or at the point [DW05, LL10]. Although nice results have been reported, it is recognized in [DW05] that "more precise theoretical analysis are needed in identifying conditions on the metric, geometry, and the generator distributions that guarantee the well-defined duality between the AVDs and the ADTs". The Labelle and Shewchuk approach [LS03] is easier to analyze and, in the 2-dimensional case, the authors have proposed a refinement algorithm that can provably produce anisotropic meshes. This approach has been extended by Cheng et al. [CDRW06] to produce anisotropic meshes of surfaces embedded in $\mathbb{R}^3$. The algorithm is however very complicated and no implementation has been reported. Although some progress has been recently reported in [CG11], it remains unclear under which conditions an anisotropic Voronoi diagram admits a dual embedded triangulation for 3 or higher dimensional domains.

A different approach has been proposed by Boissonnat, Wormser and Yvinec [BWY08]. They introduced a new variant of anisotropic Delaunay meshes, called *locally uniform anisotropic meshes*. Such meshes are not defined by duality from some anisotropic Voronoi diagram. Instead, it is defined as a simplicial mesh in which the star of each vertex is Delaunay for the metric attached to the vertex. It is shown that, when the metric field satisfies some continuity property, such a mesh can be obtained for bounded polyhedral domains of $\mathbb{R}^2$ or $\mathbb{R}^3$ with no sharp edges. The extension of the approach to surfaces has not been considered so far and no implementation has been reported for 3-dimensional domains.

1.2 Our Contributions

Our work uses the notion of locally uniform anisotropic mesh defined in [BWY08], combined with the notion of Delaunay triangulation restricted to a surface. As in the previous approaches, we assume that the anisotropy is prescribed by a metric field that associates to each point p of the surface, a symmetric positive definite 3×3 matrix M_p describing the metric at point p. Given a set of (punctual) sites V and $v \in V$, it is easy to compute the 3D Delaunay triangulation $\mathrm{Del}_v(V)$ for the metric M_v of v. Indeed, $\mathrm{Del}_v(V)$ is just the image of an Euclidean Delaunay triangulation under a stretching transformation. For each site $v \in V$, we consider the *star* T_v of v in $\mathrm{Del}_v(V)$, and the *surface star* S_v. The star T_v is the subcomplex of $\mathrm{Del}_v(V)$ formed by the tetrahedra that are

incident to v, while the surface star S_v is the subcomplex formed by the triangular facets incident to v and whose dual Voronoi edges intersect S. The surface star S_v is thus a collection of triangles incident to p.

In general, there are *inconsistencies* among the surface stars of the sites : a facet τ, appearing in the surface stars of some of its vertices, may not appear in the surface stars of all of them. As a result, the set of surface stars of V does not form a triangulated surface.

In [BWY08, BWY11], we showed that it is possible to refine the set V of sites until there is no more inconsistencies among the set $T(V)$ of 3D stars. This leads to a locally uniform anisotropic 3D mesh in which the star of each vertex is Delaunay with respect to the metric of this vertex. In the present paper, we are interested in generating surface meshes. The algorithm will incrementally compute a set V of sites on the surface and will maintain the associated surface star set $S(V)$. The set V is refined until there is no inconsistencies among the facets of $S(V)$. In contrast to [BWY08], we do not consider inconsistencies among the 3D stars. When there is no more inconsistencies among the surface stars, the facets of these surface stars form a locally uniform anisotropic surface mesh that approximates S and conforms to the specified metric field. This algorithm is provably correct and much more efficient than the 3D anisotropic mesh generator of [BWY08].

In addition to conforming to the given metric field, this new method has several notable advantages over previous methods.

- The algorithm relies on the well-established concepts of restricted Delaunay triangulations and Delaunay refinement and comes with theoretical guarantees.
- It can handle complicated geometries and topologies, as well as very anisotropic metric fields.
- It can handle various types of surfaces, e.g. implicit surfaces, polyhedra, isosurfaces in 3D images.
- It is easy to implement and robust since it relies on the usual Delaunay predicates (applied to some stretched spaces).
- The facets of the output mesh have a good aspect ratio as measured with respect to the metric field.

The idea of maintaining independent stars for each vertex of a mesh has been first proposed by Shewchuk [She05] for maintaining triangulations of moving points. This idea has also been used to build the dual of an anisotropic Voronoi diagram as defined by Labelle and Shewchuk [Sch08]. The idea of stitching stars restricted to a surface can be found in the work on the tangential complex [BG10]. The context is however different from ours since no anisotropy was considered and the main motivation was to reconstruct a k-manifold embedded in a d-dimensional space without constructing any global d-dimensional triangulation, which is not a critical issue in 3 dimensions (see however the discussion in Section 5.3). A major technical difference with our paper is that the stars considered in [BG10] are all contained in a global Euclidean d-dimensional Delaunay triangulation, which is not the case here. Combining ideas from the present paper and [BG10] would allow to define and generate anisotropic meshes of smooth submanifolds embedded in high dimensional spaces.

2 Preliminaries

2.1 Anisotropic Metric

An anisotropic metric in $\mathbb{R}^3$ is defined by a symmetric positive definite quadratic form represented by a 3×3 matrix M. The distance between two points a and b, as measured using a metric M, is defined as

$$d_M(a,b) = \sqrt{(a-b)^t M(a-b)}.$$

This definition can be extended to M-lengths and M-areas.

In the following, we often use the same notation, M, for a metric and the associated matrix in a given basis. Given the symmetric positive definite matrix M, we denote by F_M any matrix such that $\det(F_M) > 0$ and $F_M^t F_M = M$. Note that

$$d_M(a,b) = \sqrt{(a-b)^t F_M^t F_M (a-b)} = \|F_M(a-b)\| \tag{1}$$

where the notation $\|.\|$ stands for the Euclidean norm.

Given some metric M, an M-sphere $\mathcal{C}_M(c,r)$, with center c and radius r, is defined as the set of points p such that $d_M(c,p) = r$. Likewise, an M-ball $\mathcal{B}_M(c,r)$, is defined as the set of points p such that $d_M(c,p) \leq r$. Note that an M-sphere is an Euclidean ellipsoid, with its axes aligned along the eigenvectors of M.

Given a simplex τ in $\mathbb{R}^3$ and a metric M, we define the *M-circumsphere* $\mathcal{C}_M(\tau)$ as the smallest M-sphere that circumscribes τ. The *M-circumball* $\mathcal{B}_M(\tau)$ is the M-ball bounded by $\mathcal{C}_M(\tau)$ and the M-circumradius $r_M(\tau)$ of a simplex τ is the radius of $\mathcal{C}_M(\tau)$.

2.2 Distortion between Metrics

The following definition is due to Labelle and Shewchuk [LS03]. Given two metrics M and N, and their square-roots F_M and F_N, the *distortion* between M and N is defined as

$$\gamma(M,N) = \max\{\|F_M F_N^{-1}\|, \|F_N F_M^{-1}\|\},$$

where $\|.\|$ is the matrix norm operator associated with the Euclidean norm, i.e. for a 3×3 square matrix A, $\|A\| = \sup_{x \in \mathbb{R}^d} \frac{\|Ax\|}{\|x\|}$. Observe that $\gamma \geq 1$ and $\gamma = 1$ iff $M = N$. A fundamental property of the distortion $\gamma(M,N)$ is to relate distances d_M and d_N. Specifically, for any x, y, we have

$$1/\gamma(M,N)\ d_M(x,y) \leq d_N(x,y) \leq \gamma(M,N)\ d_M(x,y).$$

In the rest of the paper, we consider a compact closed surface S of $\mathbb{R}^3$ endowed with a continuous metric field $\{M_p, p \in S\}$ defined over S. For convenience, we will simply write X_p instead of X_{M_p} when a quantity X is measured using metric M_p. Hence, we will write for instance F_p for F_{M_p} and $d_p(a,b)$ for $d_{M_p}(a,b)$.

Given two points p and q of S, the *relative distortion* between p and q is defined as $\gamma(p,q) = \gamma(M_p, M_q)$. Note that $\Gamma = \sup_{x,y \in S} \gamma(x,y)$ is finite since S is compact.

Let γ_0 be some positive constant called the *distortion bound*. The *distortion radius* at p, $\mathrm{dr}(p,\gamma_0)$ is the upper bound on distances ℓ such that for all q and r in S,

$$\max(d_p(p,q), d_p(p,r)) \leq \ell \ \Rightarrow \ \gamma(q,r) \leq \gamma_0.$$

It can be shown that the *distortion radius* satisfies the following *continuity property* [BWY11] :

$$\begin{aligned} \frac{1}{\gamma(p,q)}(\mathrm{dr}(p,\gamma_0) &- d_p(p,q)) \leq \mathrm{dr}(q,\gamma_0) \\ \mathrm{dr}(q,\gamma_0) &\leq \gamma(p,q)\,(\mathrm{dr}(p,\gamma_0) + d_p(p,q)) \end{aligned} \tag{2}$$

2.3 Sizing Field

To control the distorsion of the metric on mesh elements, we use a sizing field that is pointwise smaller than the distorsion radius. In addition to controlling the distortion of the metric, it is important to control the topology of the output mesh. To do so, we adapt the notion of *local feature size* to the anisotropic setting. The (anisotropic) local feature size at a point $p \in S$, denoted by $\mathrm{lfs}(p)$, is defined as the M_p-distance from p to the M_p-medial axis of S. It can be proved that the anisotropic local feature size enjoys a continuity property analoguous to (2).

To control both distortion and topology, we use a sizing field $\mathrm{sf}(p,\gamma_0)$ defined as $\mathrm{sf}(p,\gamma_0) = \min(\mathrm{lfs}(p), \mathrm{dr}(p,\gamma_0))$. Plainly, the sizing field $\mathrm{sf}(p,\gamma_0)$ satisfies a continuity property analog to (2). In fact, the user may include his own sizing requirement by using as sizing field any function provided that it is pointwise smaller than $\mathrm{sf}(p,\gamma_0)$ and satisfies a continuity property analog to (2).

3 Stars and Refinement

3.1 Anisotropic Star Set

Given a metric M and a set of punctual sites V, the Delaunay triangulation $\mathrm{Del}_M(V)$ for metric M is the triangulation of V such that the M-circumball of each tetrahedron is *empty*. By empty, we mean that the interior of the M-circumball contains no points of V. By (1), the Delaunay triangulation $\mathrm{Del}_M(V)$ of a finite set of points V for metric M is simply obtained by computing the Euclidean Delaunay triangulation of the stretched image $F_M(V)$, and then stretching back the result using F_M^{-1}.

For each site v in V, we consider the Delaunay triangulation $\mathrm{Del}_v(V)$ of V for metric M_v. We define the *star* T_v of site v as the subcomplex of $\mathrm{Del}_v(V)$ formed by the tetrahedra that are incident to v. The collection of all the stars T_v, $v \in V$, is called the *star set* of V. We denote it by $T(V)$.

We now adapt to the anisotropic setting the definition of a restricted Delaunay triangulation. The *surface star* of v, denoted by S_v, is the subcomplex of $T(v)$ formed by the facets of $T(v)$ that are incident to v and whose dual Voronoi edge intersects S. The collection of the surface stars S_v, for $v \in V$, is called the *surface star set* of V. We denote it by $S(V)$.

Let N be a metric and τ be a triangle with its three vertices p_0, p_1 and p_2 on S. We define the *N-axis* of τ as the set of centers of all the N-circumballs of τ. If p_0, p_1 and p_2 are sufficiently close to each other, and if τ is well-shaped with respect to metric N, the *N-axis* of τ intersects S. We call *surface N-circumball* of τ, and note $\mathcal{B}_N(\tau)$, the smallest N-circumball of τ centered on S. A facet $\tau \in T_v$ incident to v belongs to S_v iff its surface M_v-circumball is empty.

The facet τ with vertices p_0, p_1 and p_2 on S has three surface circumballs $\mathcal{B}_i(\tau)$ associated respectively to the metrics M_i of each of its vertices. We define the *distortion* $\gamma(\tau)$ of τ as the maximal distortion between any pair of points of S which are both inside the same surface circumball $\mathcal{B}_i(\tau)$.

3.2 Inconsistencies

Two surface stars S_v and S_w are said to be *inconsistent* if some facet incident to v and w appears in only one of the two stars S_v and S_w. Any facet that appears in the stars of some of its vertices, but not in the stars of all of them, is also said to be *inconsistent* (see Figure 1).

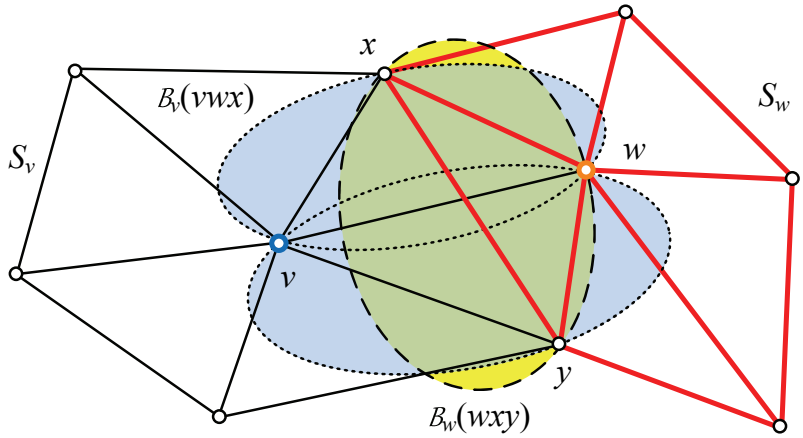

Figure 1: Example of inconsistent stars in 2D : stars S_v and S_w are inconsistent because edge $[vw]$ belongs to S_v but not to S_w.

Our algorithm incrementally inserts new sites in V to resolve inconsistencies among the surface stars.

3.3 Quasi-Cosphericity

Let $\gamma_0 > 1$ be a bound on the distortion and M be a metric. Following [BWY08], we now introduce the notion of (γ_0, M)-cosphericity and show its link to inconsistent simplices.

Let τ be a facet of some surface star and $p \in V \setminus \tau$. We call $U = (\tau, p)$ a (γ_0, M)-cospherical configuration iff there exists two metrics N and N' such that

1. $\gamma(M, N) \leq \gamma_0$, $\gamma(M, N') \leq \gamma_0$ and $\gamma(N, N') \leq \gamma_0$;
2. p is included in the interior of the surface N'-circumball $\mathcal{B}_{N'}(\tau)$ and not in the interior of the surface N-circumball $\mathcal{B}_N(\tau)$.

If M is clear from the context, we simply say that U is a γ_0-*cospherical* configuration and if both M and γ_0 are understood, we say that U is a *quasi-cospherical* configuration.

See Figure 2 for an illustration where S is the plane.

The following lemma ensures that, when all simplices in the stars have small distorsion, inconsistencies only result from the occurence of quasi-cospherical configurations.

Lemma 3.1 (Inconsistencies and quasi-cosphericities) *Let τ be an inconsistent facet that appears in star S_v but not in star S_w where v and w are two vertices of τ. If $\gamma(\tau) < \gamma_0$, then there exists a site $q \in S_w$ such that the configuration (τ, q) is (γ_0, M_v)-cospherical.*

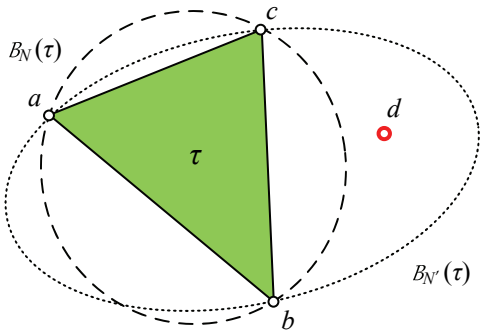

Figure 2: The configuration (τ, d) is quasi-cospherical because point d is outside the N-circumball $\mathcal{B}_N(\tau)$ but inside the N'-circumball $\mathcal{B}_{N'}(\tau)$.

Proof Take $N = M_v$ and $N' = M_w$. Because the distortion of τ is less than γ_0, we have $\gamma(v, w) \leq \gamma_0$, and therefore the distorsions $\gamma(M_v, N)$, $\gamma(M_v, N')$ and $\gamma(N, N')$ are all less than γ_0. Assume first that all the vertices of τ appear in S_w. Then if $\tau \notin S_w$, the surface M_w-circumball $\mathcal{B}_w(\tau)$ of τ contains some sites of S_w in its interior. Consider now the case where some vertex v' of τ does not appear in S_w. This means that $v'w$ is not an edge of $\mathrm{Del}_w(V)$. Hence, any surface M_w-ball circumscribing edge $v'w$, and in particular $\mathcal{B}_w(\tau)$, contains some sites of S_w. Thus, in both cases, there is some site q in S_w such that q is included in the interior of $\mathcal{B}_w(\tau)$ and not in the interior of $\mathcal{B}_v(\tau)$. It follows from Definition 3.3 that (τ, q) is a (γ_0, M_v)-cospherical configuration. □

4 Meshing Algorithm

4.1 Algorithm Outline

Let S be a given surface we want to mesh. The algorithm constructs the set of sites V by inserting new sites on S in a greedy way. While there remain inconsistent or bad facets in the surface star set, the algorithm selects one such facet τ and kills τ by inserting a new site on S called the *refinement point* of τ. Inserting a new point is performed by the following `Insert` procedure that maintains the star sets $T(V)$ and $S(V)$. We say that a tetrahedron σ of a star T_w *conflicts* with a point p if p belongs to the M_w-circumball of p.

`Insert`(p)

1. insert p in each star T_w that contains a simplex in conflict with p and update the surface star S_w;
2. create the new stars T_p and S_p.

The refinement algorithm consists in applying the following rules with a priority order : Rule (i) is applied only if no Rule (j), $j < i$, can be applied. The algorithm ends when no rule applies anymore. The algorithm relies on procedure `Insert` to insert a new point in the data structures, and on procedure `Pick_valid` to select the location of the new site (see the next section).

If τ is a facet in some star S_v, we write $B_v(\tau)$ for the surface M_v-circumball of τ, and $c_v(\tau)$ and $r_v(\tau)$ for the center and the radius of this ball. The quality of facet τ with respect to metric M_v is characterized by the *M_v-radius-edge ratio*, $\rho_v(\tau)$, which measures the ratio between the M_v-circumradius of τ and the M_v-length of the edge of τ of shortest M-length.

The refinement rules depend on the constants α, ρ_0 and γ_o.

1. **Sizing field - Distortion :**
   If $\exists v \in V$ and $\tau \in S_v$ such that $r_v(\tau) \geq \alpha\, \mathrm{sf}(c_v(\tau), \gamma_0)$,
   then `Insert`$(c_v(\tau))$;

2. **Shape - Radius-edge ratio :**
   If $\exists v \in V$ and $\tau \in S_v$ such that $\rho_v(\tau) > \rho_0$,
   then `Insert` (`Pick_valid`(τ, M_v));

3. **Inconsistency :**
   If $\exists v \in V, \tau \in S_v$ such that τ is inconsistent,
   then `Insert` (`Pick_valid`(τ, M_v));

Assume that the algorithm terminates. Then, no rule applies anymore. In particular rule (3) does not apply which means that the star sets $S(V)$ includes no inconsistent simplex. All surface stars are consistent and can be stitched together to form a triangulated surface $\mathcal{T}$ with the property that the star of any vertex v in $\mathcal{T}$ is Delaunay for metric M_v. Hence, $\mathcal{T}$ is a locally uniform anisotropic mesh of surface S. The next step is therefore to establish that the algorithm terminates (Section 4.3).

4.2 Picking Region and Hitting Sets

The simplest idea to kill a bad triangle τ in some surface star S_v is to insert a new site at the center of the surface M_v-circumball of τ. However, this simple strategy may lead to endless occurrences of quasi-cosphericities and inconsistencies. We present in this section an alternative strategy, inspired by the strategy introduced by Li and Teng [LT01, Li03] to avoid slivers in isotropic meshes and also used in [BWY08]. The basic idea is to relax the choice of the refinement point of a bad facet. Instead of using the center of the surface circumball, the refinement point is picked from a small region around this center, and carefully chosen so as to avoid the formation of new quasi-cosphericities.

Let $\delta < 1$ be a constant called the *picking ratio*. The *M_v-picking region* of a facet τ in star S_v, denoted by$P_v(\tau)$, is the intersection of the M_v-ball $\mathcal{B}_v(c_v(\tau), \delta r_v(\tau))$ with the surface S.

In fact, it is not possible, when choosing a refinement point in the picking region $P_v(\tau)$ of a triangle τ of S_v to avoid the formation of any new quasi-cospherical configurations. The `Pick_valid` procedure will only avoid the creation of *small* quasi-cospherical configurations where the meaning of *small* depends on an additional parameter $\beta \geq 1$ and is made precise below.

Assume we are choosing a new site p in the M_v-picking region of some triangle τ in surface star S_v. A subset σ of three sites of the current set V is said to *hit* p if the configuration $U = (\sigma, p)$ is both :

1. (γ_0, M)-cospherical for some metric M such that $\gamma(M_p, M) \leq \gamma_0$,

2. *small*, meaning that the smallest *M-circumradius* of all the triangles with vertices in U is less than $\beta r_v(\tau)$.

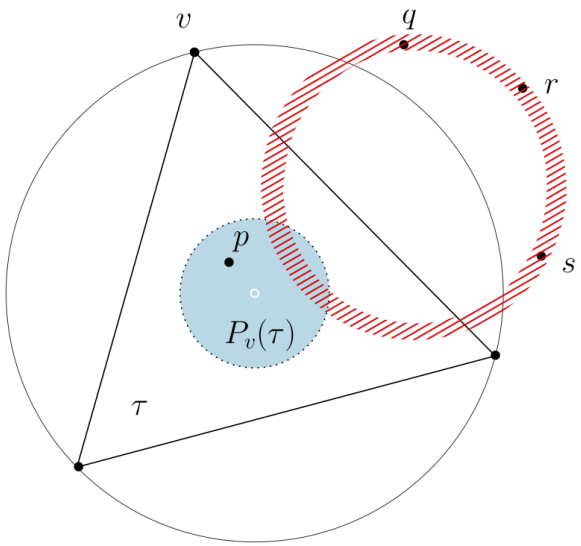

Figure 3: The picking region of triangle τ. Set $\sigma = \{q, r, s\}$ is a hitting set for the picking region $P_v(\tau)$. It defines a forbidden region, shown with red dashes, to be avoided by the refinement point p of τ.

A subset σ of three sites of V that hits some point p in the picking region $P_v(\tau)$ is called a *hitting set* of $P_v(\tau)$. We associate to each hitting set σ a so-called *forbidden region* that consists of the points of $P_v(\tau)$ hit by σ. A point of the picking region $P_v(\tau)$ is called a *valid refinement point* when it does not belong to any forbidden region. See Figure 3.

Note that the definition of a valid refinement point depends on the constants δ and β that respectively defines the size of the picking regions and bounds the minimum size of new quasi-cospherical configurations. It also depends on the constant γ_0 that defines quasi-cosphericity. The next lemma is fundamental for our algorithm.

Lemma 4.1 (Picking lemma) *For any positive values of the parameters $\alpha \leq 1$, $\beta \geq 1$, $\delta < 1$ and ρ_0, it is possible to choose a distortion parameter γ_0 close enough to 1, so that the valid refinement points form a subregion of positive area inside any picking region considered by the algorithm.*

The proof, only sketched here, is given in [BWY11]. It uses a volume argument based on the two following facts. 1. The number of hitting sets of a picking region is bounded. 2. For given values of parameters α, β, δ and ρ_0, the volume of the forbidden region associated to a hitting set tends to zero when γ_0 tends to 1.

To find a valid refinement point in the M_v-picking region $P_v(\tau)$ of some bad k-simplex τ, the insertion algorithm calls the following `Pick_valid` procedure that randomly selects a point in the picking region $P_v(\tau)$ until a valid refinement point is found. This procedure depends on the constants γ_o, δ and β, to be fixed later in Section 4.3.

`Pick_valid`(τ, M_v) :

1. Pick randomly a point p in the picking region $P_v(\tau)$

2. If there exists a subset of three sites in V that hits p,
   then discard p and go back to 1.

3. Return p.

Lemma 4.1 ensures that the picking process will succeed with a positive probability at each trial. Observe also that `Pick_valid` takes care of small configurations only. Hence, checking if a point of some picking region is valid requires only to consider sites that are close to the picking region. Furthermore, the number of triplets we need to consider as possible hitting sets is bounded by a constant (see the proof of Lemma 4.1 in [BWY11]).

4.3 Termination of the Algorithm and Quality of the Mesh

We now prove that, under some conditions on the parameters ρ_0, γ_0, δ and β, the number of inserted vertices is bounded, which implies that the algorithm terminates. Let us define the *intersite distance* between two sites p and q as $d(p,q) = \min(d_p(p,q), d_q(p,q))$, and the insertion radius $r(p)$ of a site p as the minimum intersite distance between p and the current set of sites, i.e. the set of sites that have been inserted before p. Recall that $\Gamma = \sup_{x,y \in S} \gamma(x,y)$ is finite since S is compact. The proofs of the following lemmas, available in [BWY11], are only sketched here.

Lemma 4.2 (Insertion radius lemma) *If the two conditions*

$$\frac{(1-\delta)\rho_0}{\Gamma\gamma_0^2} \geq 2 \quad \text{and} \quad \frac{(1-\delta)\beta}{\Gamma\gamma_0^2(1+\delta)} \geq 2$$

hold, then there exists a constant C such that the insertion radius $r(p)$ of any vertex p of the mesh is at least $C\,\mathrm{sf}(p)$.

The proof uses the fact that the new point is inserted close to the center of the surface circumball of a facet in a surface star, and therefore relatively far from the current set of sites. The precise proof is a case analysis that studies the three rules of the algorithm. From this lemma, we can bound from below the intersite distance between any two points p and q in the final mesh. We consider whether p or q has been inserted last. Lemma 4.2 and the continuity property (2) of sf then allow to prove the following lemma.

Lemma 4.3 (Smallest intersite distance) *If the conditions of Lemma 4.2 are satisfied, the smallest intersite distance between a site p and all the other vertices of the final mesh is at least :*

$$d(p,q) \geq \frac{C}{(1+C)\Gamma}\,\mathrm{sf}(p).$$

It follows from the lemma that the M_p-balls $B_p(p, l(p))$, where $l(p) = \frac{C}{2(1+C)\Gamma}\,\mathrm{sf}(p)$, have disjoint interiors. A simple volume calculation then provides an upper bound on the number of vertices of the final mesh.

Lemma 4.4 (Number of inserted sites) *If the conditions of Lemma 4.2 are satisfied, the number n of sites inserted by the algorithm is at most*

$$n \leq \frac{\gamma_0^2}{\pi}\left(1 + \frac{2\Gamma(1+C)}{C}\right)^2 \int_S \frac{d_M p}{\mathrm{sf}(p)^2}.$$

When the algorithm terminates, the surface stars are consistent and can be stitched so as to make a triangulated surface mesh $\hat{S}$. Each triangular facet of $\hat{S}$ is Delaunay for the metric of any of its three vertices, conforms to the sizing field sf (Rule 1) and has a radius-edge ratio smaller than ρ_0 (Rule 2) when measured using the metric of any of its vertices. The radius-edge ratio being a fair quality measure for triangles [She02], the last property ensures that the facets of the output triangulated surface are well-shaped with respect to the local anisotropy.

In addition, we can ensure that the topology of $\hat{S}$ is the same as the topology of S. This can be done by choosing a sizing field that is sufficiently small with respect to the local feature size of the surface. This condition, together with the bound on the radius-edge ratio of the facets, ensures that $\hat{S}$ is homeomorphic (and even ambient isotopic) to S. See for example [ACDL02, BO05].

5 Implementation

The algorithm can handle different kinds of surfaces provided that we are given an oracle able to :

(1) compute a few initial points on the surface,

(2) detect and compute the intersections between a given line segment and the surface.

A small set of initial points is enough. We create the initial stars and put the bad facets of the surface stars in a global queue containing all the facets to be refined. The refinement process can then start. Our implementation uses the CGAL library [CGA].

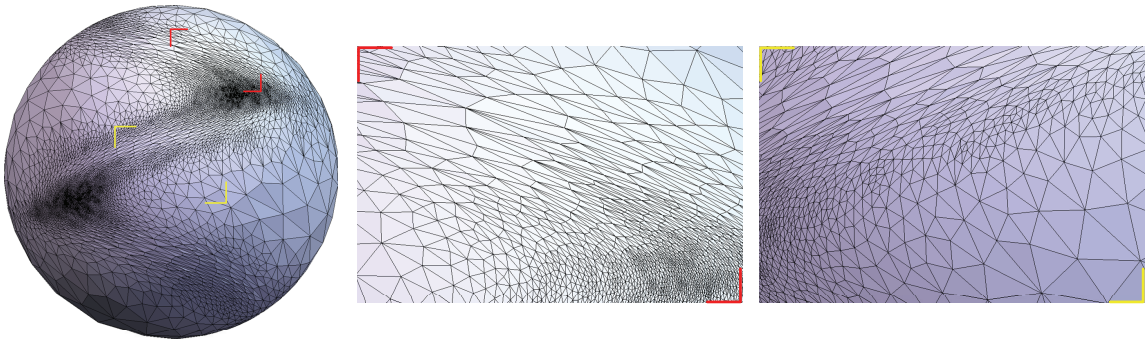

Figure 4: A unit sphere endowed with a metric field derived from the sine-shaped shock function $E(x, y, z) = \tanh(\frac{1}{\lambda}(2x - \sin(5y))) + x^3 + xy^2$. $\lambda = 0.6$, $\rho_0 = 3.0$, $r_0 = 0.1$, $\gamma_0 = 1.4$, $\beta = 2.5$ and $\delta = 0.3$. The mesh has 18323 vertices and 36642 facets.

5.1 The algorithm parameters

Our anisotropic surface mesh generation algorithm depends on five parameters : α, β, δ, ρ_0 and γ_0. A priori, the parameters values should be chosen to satisfy the conditions of Lemma 4.2 and γ_0 should be set close enough to 1 to ensure the existence of valid refinement points in any picking region (Lemma 4.1). Note that, if the conditions of Lemma 4.2 are satisfied for a given choice of α, β, δ, ρ_0 and γ_0, they are also satisfied for the same values of α, β, δ, ρ_0 and a lower γ_0.

Conditions in Lemmas 4.1 and 4.2 are sufficient to ensure the termination of the meshing algorithm. However they are not necessary. These conditions arise from our proof of termination and our will to keep the proof simple makes them over-restrictive. In particular, the conditions in Lemma 2 depend on a global distortion bound Γ. This makes the proof easier but a more thorough proof would involve local distortion bounds. In practice, in our experiments we do not try to evaluate a global distortion bound, nor to respect the conditions of Lemmas 4.1 and 4.2. In fact, the meshing algorithm runs smoothly and yields good meshes for a quite large range of parameter values.

Parameter α should be chosen ≤ 1. This parameter is needed to prove the picking lemma. In practice, this parameter is not critical and has been set to 1 in all our experiments.

Parameters β and δ, which define the picking region and the validity of a refinement point, have little influence on the quality of the final mesh. Those parameters have to be set in such a way that procedure `Pick_valid`. finds a valid refinement point in a reasonable amount of time. By definition, we must have $\delta < 1$, and the second condition in Lemma 4.2 suggests to choose $\beta > 2$. In all experiments reported later in this paper, we use $\beta = 2.5$ and $\delta = 0.3$. Further experiments showed that varying the value of β in the range $[2, 5]$ and the value of δ in the range $[0.1, 0.5]$ has basically no effect on the results (number of vertices, computing time, and quality of the output mesh). More information on these experiments are given in the appendix.

Parameter ρ_0, on one hand, should be as small as possible to ensure that the simplices of the final mesh are well-shaped according to the metric field. In the isotropic case, the Delaunay refinement algorithm for smooth surface [Che93] is known to terminate for any value $\rho_0 > 1$. The first condition in Lemma 4.2 suggests to take $\rho_0 > 2$. In all our experiments, we use $\rho_0 = 3$.

The size of the simplices in the final mesh mostly depends on the parameter γ_0 and on the sizing field . As defined above, the sizing field takes into account the variation of the metric (distortion radius) and the geometry of the surface (local feature size). Estimating the local feature size is a notoriously difficult problem, for which existing solutions can be used [BO05, ADA08]. In this paper, the emphasis is put on the metric variation and, in our experiments, we assume the local feature size to be uniformly bounded from below by some parameter r_0 that we use to specify the desired size of the mesh simplices. To take into account the metric variation, we replace the distortion radius by a direct and simpler control on the distortion of the simplices. Specifically, Rule 1 is replaced in practice by the following rule :

1. **Sizing field - Distortion :**
   If $\exists v \in V$ and $\tau \in S_v$ such that
   either $r_v(\tau) \geq r_0$ or $\gamma(\tau) \geq \gamma_0$,
   then `Insert`$(c_v(\tau))$;

Parameter γ_0 has to be greater than 1 and close enough to 1. The influence of the value of parameter γ_0 is complex. On one hand, choosing for γ_0 a value very close to one triggers many applications of the *Sizing field - Distorsion* rule and leaves only a few inconsistencies to be removed by the *Inconsistency* rule. Such a choice also keeps the area of the forbidden regions small, which eases picking valid refinement points. On the other hand, such a choice may lead to an overly dense mesh. Raising the value of γ_0 results in a larger number of inconsistencies to be removed by the *Inconsistency* rule, an increasing number of calls to procedure `Pick_valid`, which therefore slows down the meshing process. In all our experiments, γ_0 is chosen between 1.3 and 1.8. Also, for efficiency reasons, we approximate $\gamma(\tau)$ by the maximum distortion between two vertices of τ. The influence of parameter γ_0 on the resulting mesh density and algorithm efficiency is shown in

Figure 7.

In summary, our implementation depends on parameters β, δ, ρ_0, γ_0 and r_0. In all our experiments, β, δ, ρ_0 have been set to fixed values.

5.2 Computing the Metric Field

The algorithm requires to compute the metric field at any refinement point on S. It can either be prescribed by the user or approximate the curvature tensor of the surface at the refinement point. We detail below several cases to illustrate the versatility of our approach.

5.2.1 Three-Dimensional Scalar Fields

Three-dimensional scalar fields are common in physical or mathematical applications. If E denotes such a scalar field defined in $\mathbb{R}^3$ and S a surface, we may want to mesh S so as to provide the best approximation of E on S. The triangles should then be elongated orthogonally to the gradient ∇E. We define the metric field as follows.

$$F_M = U \cdot \text{diag}\{1/(1 + \varphi \, \|\nabla E\|), 1, 1\} \cdot U^t$$

where $U = [\frac{\nabla E}{\|\nabla E\|}, U_1, U_2]$, U_1 and U_2 being two arbitrary unit vectors that form an orthogonal frame with ∇E. If the gradient is zero, the metric field is isotropic. Parameter φ controls how the facets are stretched with respect to the norm of the gradient.

Figure 4 presents an anisotropic Delaunay mesh of a sphere endowed with a sine-shaped shock scalar field. The triangles are elongated orthogonally to the gradient direction and the density depends both on the norm of the gradient and on the changing rate of the gradient.

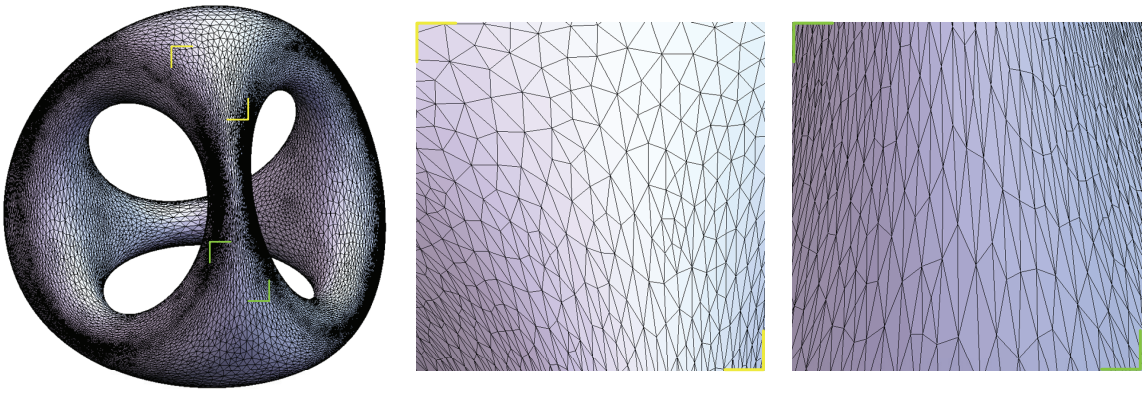

Figure 5: The "chair" implicit surface with a curvature-related metric field. The equation of the chair is $f(x,y,z) = \left(x^2+y^2+z^2-ak^2\right)^2 - b\left((z-k)^2-2x^2\right)\left((z+k)^2-2y^2\right)$. In this example, $a = 0.8$, $b = 0.4$, $k = 1.0$, $\rho_0 = 3.0$, $r_0 = 0.1$, $\gamma_0 = 1.3$, $\beta = 2.5$ and $\delta = 0.3$. The mesh has 24164 vertices and 48336 facets.

5.2.2 Curvature Tensor on an Implicit Surface

Let S be an implicit surface defined by $f(x, y, z) = 0$. At each point p of S, we denote by N the normal vector $-\nabla f / \|\nabla f\|$ and by H the Hessian of f. The principal curvatures $c_{\max}$, $c_{\min}$ are the non-zero eigenvalues of the matrix

$$C \quad = \quad P \cdot H' \cdot P \;\text{ where }\; H' = H / \|\nabla f\|\,, \;\; P = I_3 - N \cdot N^t,$$

see e.g. [Hug03]. Let $U_{\max}$, $U_{\min}$ and N be the normalized eigenvectors of C, and let $U = [U_{\max}, U_{\min}, N]$. The metric at point p is defined by $M_p = F_p^t F_p$, where:

$$\begin{aligned} F_p &= U \cdot \Delta \cdot U^t, \\ \text{with } \Delta &= \text{diag}\{e_{\max}, e_{\min}, e_{\text{n}}\}, \\ e_{\max} &= \max\{\epsilon, \|c_{\max}\|\}, \\ e_{\min} &= \max\{\epsilon, \|c_{\min}\|\}. \end{aligned}$$

Here, ϵ is a small positive constant that prevents the eigenvalues of the metric to vanish when $c_{\max}$ and/or $c_{\min}$ vanish, e.g., at planar or parabolic points. e_{n} is a global constant to be discussed in the next subsection, and $\Delta = \text{diag}\{e_{\max}, e_{\min}, e_{\text{n}}\}$ stands for a diagonal matrix whose entries are $\{e_{\max}, e_{\min}, e_{\text{n}}\}$.

Figure 5 shows an anisotropic Delaunay mesh of the "chair" implicit surface.

5.2.3 Curvature Tensor on Polyhedral Surfaces

To remesh a polyhedral surface S that may be an approximation of a smooth surface $\tilde{S}$, we can estimate the curvature tensor of $\tilde{S}$ from S. This is done in two steps.

1. **Curvature estimation at the vertices of S**
   We first estimate the curvature tensor at each vertex $v \in S$ as follows. We compute the Euclidean k-nearest vertices of v, and then apply the jet-fitting algorithm of [CP05] to estimate

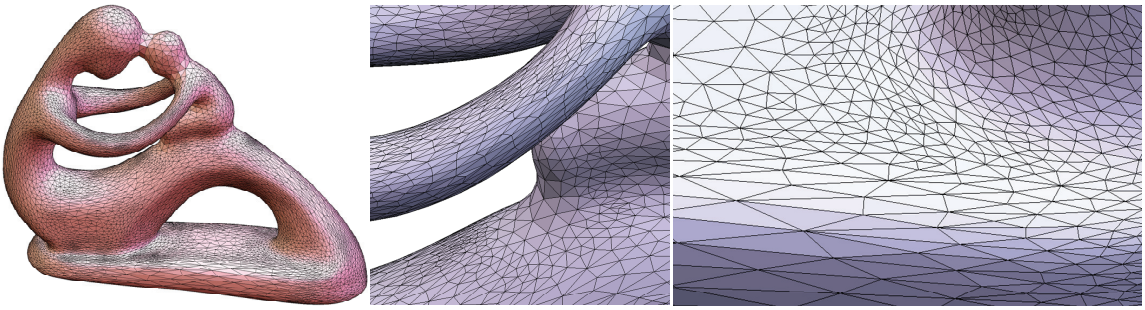

Figure 6: Result on the polyhedral surface "fertility", a model from AIM@SHAPE repository. We took $r_0 = 0.1$, $\gamma_0 = 1.5$. The mesh has 12,480 vertices and 24,972 facets.

   the two principal curvatures $c_{\max}$ and $c_{\min}$, the corresponding unit vectors $U_{\max}$ and $U_{\min}$, and the normal vector N at v. From the estimated curvatures, we compute $e_{\max}$ and $e_{\min}$ as above. We choose e_{n} as explained in subsection 5.4 and store at each vertex v a resulting estimated metric tensor $M(v) = U\Delta^2 U^t$, with $U = [U_{\max}, U_{\min}, N]$ and $\Delta = \text{diag}\{e_{\max}, e_{\min}, e_{\text{n}}\}$.

2. **Blending metric tensors**
   To estimate the metric tensor at any prescribed query point on S, we smoothly blend the metric tensors computed in the previous step using the following formula :

$$\begin{aligned} M_p &= \sum_{v \in S} \omega_p(v) M(v), \quad \text{with} \\ \omega_p(v) &= \mu_p(v) \Big/ \sum_{v \in S} \mu_p(v), \quad \mu_p(v) = e^{-\|p-v\|^2/r_B^2} \end{aligned}$$

   where r_B is a parameter that controls the locality of the blending. By default, r_B can be computed by averaging the distance to the k-nearest points of each vertex in S.

   M_p is a positive combination of symmetric tensors and therefore has no negative eigenvalues. This property can be proved by contradiction. Assume that M_p has some negative eigenvalues. Then there exists a vector x that satisfies $x^t M_p x = \sum_{v \in S} \omega_p(v) x^t M(v) x < 0$. Since all $M(v)$ are symmetric tensors, $x^t M(v) x \geq 0$ and, since according to the definition, $\omega_p(v) \geq 0$, we have $x^t M_p x \geq 0$, which provides a contradiction.

Figure 6 shows an anisotropic Delaunay mesh of a polyhedral surface conforming to the estimated curvature metric field.

5.3 Computing Surface Stars

Our algorithm needs to maintain the surface star S_v of each vertex v in the mesh. As described above, S_v can be extracted from the star T_v of v in $\mathrm{Del}_v(V)$. Yet, maintaining stars T_v is not a fully satisfactory solution. Indeed, since the vertices of the mesh all lie on the surface, most of the tetrahedra in stars T_v spread through space and their diameters do not decrease significantly when the sample density increases. As a consequence, the algorithm is not local and the computing time is not optimized. To remedy this problem, we add a few vertices close to the medial axis of the surface and maintain for each vertex v a triangulation T'_v that does not necessarily include the star T_v but nevertheless has the same restricted star S_v as T_v. The vertices close to the medial axis are computed from the initial points following Amenta and Bern's poles method [AB99]. The triangulation T'_v is defined by the following insertion rule: a site p is inserted in the triangulation T'_v of a site v only if p belongs to the so-called conflict zone of S_v. The *conflict zone* of a surface star S_v, denoted by Z_v, is the union of the surface M_v-circumballs $\mathcal{B}_v(\tau)$ for all the facets τ of S_v. The correctness of this variant of the algorithm follows from the fact that the *conflict zone* of star S_v can only shrink when inserting a new site p, and from the fact that the surface star S_v has to be updated iff p belongs to Z_v.

When the sample density increases, the mesh facets become small and have small surface circumballs. At each insertion of a new site p, we use the conflict zones as filters to quickly report the vertices v whose small triangulations T'_v have to be updated and the vertices w that have to be inserted in the triangulation T'_p created for the new site.

To each conflict zone Z_v, we associate its (Euclidean) axis-parallel bounding box that we simply call the bounding box of v and denote by B_v. We maintain in a data structure $\mathcal{Q}$ the bounding boxes of the current set of vertices V and we maintain V in a kd-tree $\mathcal{K}\mathrm{d}(V)$. When inserting a new site p, we first query the data structure $\mathcal{Q}$ and report all the boxes that contain p. If p belongs

to some box B_v, we further check whether p belongs to the conflict zone Z_v and, in the affirmative, we insert p in the triangulations T'_v. We then build the triangulation T'_p of the new site. T'_p is initialized as the star of p in the 3D Delaunay triangulation of p and of all the vertices v whose triangulations T'_v have been updated by inserting p. Then, we compute a bounding box for the current conflict zone Z_p, and query the kd-tree $\mathcal{K}\mathrm{d}(V)$ to compute all the vertices in this box. Each vertex in this box is inserted in T'_p if it belongs to the current conflict zone Z_p.

5.4 Sliver Prevention

When the sampling is dense enough, quasi-cospherical configurations are tetrahedra with well-shaped facets (because of Rule 2) that have small volumes and are almost tangent to the surface. We call such tetrahedra *slivers*. We can reduce the number of slivers by enlarging the eigenvalue of the metric in the direction normal to the surface (eigenvalue e_{n} defined in the preceding subsection). Indeed, the volume of those tetrahedra in the new metric will be increased, hence reducing the number of slivers and inconsistencies. Observe that this modification of the metric does not significantly affect the shape of the facets of the mesh since those facets are almost tangent to the surface. Experimental evidence indicates that e_{n} should be of the same order of magnitude as the global maximum of $c_{\max}$.

6 Results

6.1 Performance and Complexity

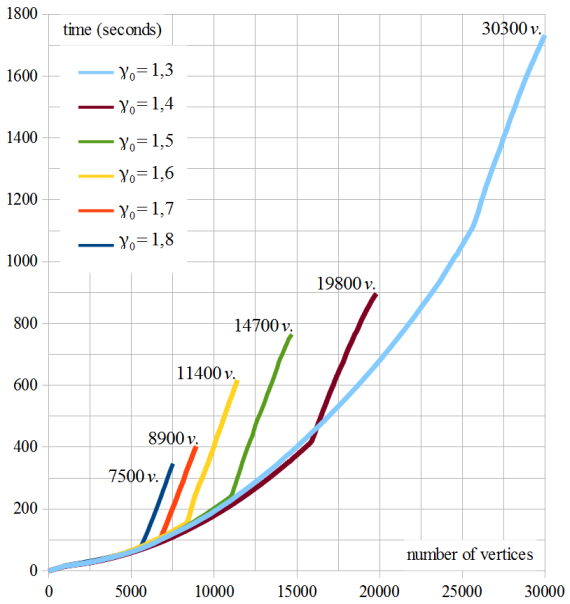

Figure 7: Computing time to mesh an implicit torus with $r_0 = 0.1$ and different values of γ_0 between 1.3 and 1.8.

Our experiments reveal that our algorithm is roughly linear behavior with respect to the number of vertices of the final mesh. As is well-known, for most point sets and insertion orders, inserting a new point in a Delaunay triangulation takes a constant time. Moreover, our use of the conflict zones (Section 5.3) ensures that a constant number of stars are visited when inserting a new vertex and that a constant number of vertices are considered when creating the star of the new vertex.

Figure 7 shows computation times for meshing a torus defined by an implicit function. All meshes are generated with $r_0 = 0.1$ and for different values of γ_0. Each curve in Figure 7 consists roughly of two linear portions with different slopes. This corresponds to the fact that, in a first stage, only Rule 1 is applied. During the second stage, Rules 2 and 3 and the `Pick_valid` function are activated, which slows down the selection of new points.

6.2 Mesh Quality

Our meshing algorithm provides surface meshes of guaranteed quality since the algorithm controls the radius-edge ratio of each facet, as measured by the metric of any facet vertex. Figure 8 shows the histogram of the squared radius-edge ratios of the facets. For each facet, three radius-edge ratios were computed, one for each of the metrics of the vertices of the facet.

The quality of the meshes produced by our algorithm can still be improved by locally optimizing the positions of the vertices. This is done by adapting the optimization method of Alliez et al. [ACSYD05]. Each vertex v is moved to the barycenter of the circumcenters of the facets in star S_v, weighted by their respective areas. In our anisotropic setting, the circumcenters are M_v-circumcenters, the weights are the M_v-areas of the facets of S_v, and the barycenter b_v is an M_v-barycenter. In order to preserve the locally uniform property of the anisotropic mesh, the motion should not introduce inconsistent, badly-shaped nor over-distortioned facet. Hence, instead of directly moving v to b_v, we test new positions on the line segment (v, b_v) and check that the planned motion does not trigger the application of one of the Rules (1)-(3). We first test $v' = b_v$. If v' is a valid refinement point, we move v to v', otherwise, we set $v' = (v + v')/2$ while $d_v(v, v') > h$. Each star is optimized in turn. In our implementation, the stars are sorted by decreasing values of $d_v(v, b_v)$ and maintained in a priority queue. The first element is optimized first. The algorithm terminates when no more star can be optimized. Figure 8 shows that the distribution of radius-edge ratios is improved after the optimization step. The additional computing time required for this optimization is negligible.

6.3 Accuracy-Size Trade-off

Using an anisotropic metric field derived from the curvature tensor leads to surface meshes that enhance the trade-off between the accuracy of the approximation and the size of the mesh.

To measure the accuracy of a mesh, we use the mean error which we compute by uniformly sampling the mesh and averaging the Euclidean distance between each sample point on the mesh and its projection on the surface.

Table 6.3 provides a comparison between isotropic meshes and curvature adapted anisotropic meshes of tori. Each torus has a small radius $r = 1$, while the big radius R takes different values. In both the isotropic and the anisotropic cases, the meshes have been obtained by a refinement process where the size criterion (r_0) has been replaced by a bound on the quality of the approximation over a facet. Hence, both meshes are guaranteed to approximate the surface up to the same error bound. The bound on the approximation error is 0.001. Each column of the table gives the number of

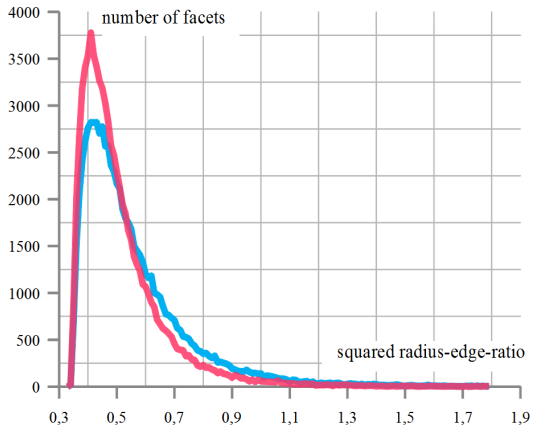

Figure 8: The distributions of the squared radius-edge ratios of the mesh facets, respectively before (blue curve) and after (red curve) the optimization step. The radius-edge ratio of each mesh facet is computed three times, using each time the metric attached to one of the three vertices. Model : "fertility", $r_0 = 1.0$, $\gamma_0 = 1.5$.

Comparing the number of vertices between anisotropic and isotropic surface meshes with the same accuracy.

	$r = 1$ $R = 10$	$r = 1$ $R = 50$	$r = 1$ $R = 200$
Number of vertices anisotropic meshes	2314	1217	898
Number of vertices isotropic meshes	6310	6120	6040
ratio κ	36%	19,9%	14,9%

Number of vertices and timings to mesh a torus with different values of β and δ.

number of vertices time	$\beta = 2$	$\beta = 3$	$\beta = 4$	$\beta = 5$
$\delta = 0.1$	5687 511.5 s	5652 197.2 s	5660 254.3 s	5643 209.1 s
$\delta = 0.3$	5227 93.5 s	5104 101.5 s	5130 121.2 s	5065 114.7 s
$\delta = 0.5$	4926 72.9 s	4979 77.4 s	4903 81.0 s	4973 72.4 s

vertices in the isotropic mesh of a piece of the torus, the number of vertices in the anisotropic mesh of the same piece of the torus, and the ratio κ between these two numbers. Anisotropic meshes have been computed using the following parameter values $\rho_0 = 3.0$, $\gamma_0 = 1.8$, $\beta = 2.5$, $\delta = 0.3$ and $r_0 = 1.0$. Figure 9 shows a piece of the mesh of the torus with $R = 50$ that was generated by our algorithm and highlights the high anisotropy ratio that can be reached.

As can be seen, the number of vertices N_a of the anisotropic mesh can be much smaller than the number of vertices N_i of the corresponding isotropic mesh with the same accuracy. The table also highlights the fact that the ratio between N_a and N_i decreases with the ratio r/R.

7 Conclusion

We have introduced a novel approach to generate anisotropic meshes of surfaces. Given a surface S endowed with a metric tensor that varies over S, our algorithm produces a mesh that approximates S and whose elements conform to the metric field. The resulting mesh provides a better approximation of the surface (if one takes as metric tensor the curvature tensor of S) or a better approximation of a function f defined over S (if we take as metric tensor the hessian of f). Our algorithm is simple and comes with theoretical guarantees without relying on heuristics. As demonstrated by our experiments, it can handle complex shapes and metric fields. The algorithm is generic in the sense that it can handle various types of surfaces such as implicit surfaces or triangulated surfaces. Although we have not presented such results, it could be easily extended to mesh isosurfaces in 3D images.

As future work, we intend to consider surfaces with sharp features following the approach of [DL09]. We also intend to extend our method to evolving surfaces [LT01], and to 3D domains bounded by surfaces and equipped with a 3D metric field, combining the results of this paper with [BWY08].

More on parameters β and δ.

We provide here a few additional experimental results, to support our claim that parameters β and δ have little influence on the quality of the final mesh.

Table 7 shows computing times and numbers of vertices needed to mesh a torus ($r = 1, R = 10$) with $r_0 = 0.1, \gamma_0 = 1.5$ and different values of β (2,3,4,5) and δ (0.1, 0.3, 0.5). As can be seen, changing the value of β has basically no effect. The situation is different with parameter δ. As expected, choosing a small δ induces small *picking regions* and makes the finding of a

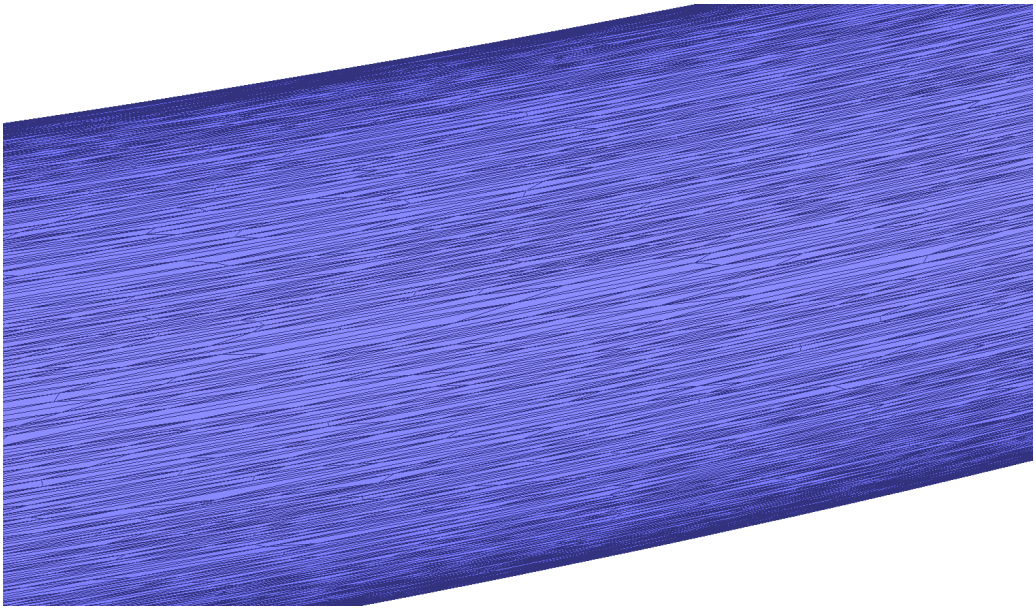

Figure 9: Anisotropic Delaunay mesh of a piece of torus with big radius 50 and small radius 1. We took $\rho_0 = 3.0$, $\gamma_0 = 1.8$, $r_0 = 1.0$, $\beta = 2.5$, $\delta = 0.3$. The approximation error is smaller than 0.001.

valid refinement point harder. Hence, the picking process fails more often, leading to more vertex insertion trials and to an increased computing time.

Figure 10 shows the quality of the meshes obtained in the above experiment. Specifically, the figure shows the distribution of the squared radius-edge ratios of the mesh facets as in Fig.8. We have selected three histograms to highlight the fact that the overall quality of the mesh is not impacted by varying β nor δ. The first histogram corresponds to our default values ($\delta = 0.3, \beta = 2$). The two others correspond to one default value and one "large" value, specifically ($\delta = 0.3, \beta = 5$), and ($\beta = 2, \delta = 0.5$).

Acknowledgments

This work has been partially supported by the Agence Nationale de la Recherche (ANR) under project GIGA (Geometric Inference and Geometric Approximation) and by the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions). The second author was partially supported by Chinese 973 Program (2010CB328001), Chinese 863 Program (2012AA040902) and the NSFC (61035002, 61272235).

References

[AB99] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, 1999.

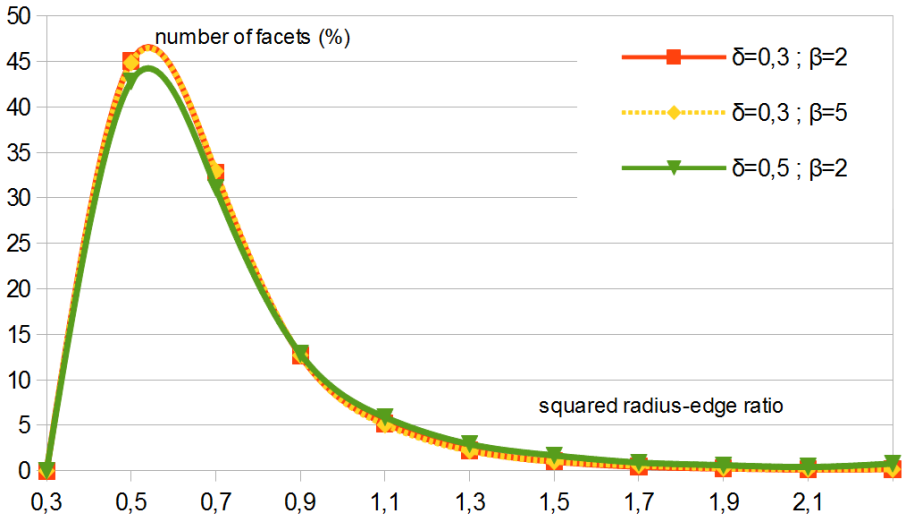

Figure 10: Squared radius-edge ratios in meshes obtained with different values of β and δ.

[ACDL02] N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Intl. Journal on Computational Geometry & Applications*, 12:125–141, 2002.

[ACSD+03] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics*, 22:485–493, 2003. SIGGRAPH '2003 Conference Proc.

[ACSYD05] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Transactions on Graphics*, 24:617–625, 2005. SIGGRAPH '2005 Conference Proc.

[ADA08] L. Antani, C. Delage, and P. Alliez. Mesh sizing with additively weighted Voronoi diagrams. In *16th International Meshing Roundtable*, pages 335–346. Springer, 2008.

[AF05] S. Azernikov and A. Fischer. Anisotropic meshing of implicit surfaces. In *Proc. of the Int. Conf. on Shape Modeling and Applications (SMI'05)*, 2005.

[BG10] Jean-Daniel Boissonnat and Arijit Ghosh. Manifold reconstruction using Tangential Delaunay Complexes. In *Proc. of the 26th Symposium on Computational Geometry*, page 200, Snowbird, États-Unis, June 2010. Full version in http://hal.archives-ouvertes.fr/inria-00440337/.

[BGH+97] Houman Borouchaki, Paul Louis George, Frédéric Hecht, Patrick Laug, and Eric Saltel. Delaunay mesh generation governed by metric specifications. part I algorithms. *Finite Elem. Anal. Des.*, 25(1-2):61–83, 1997.

[BH96] F.J. Bossen and P.S. Heckbert. A pliant method for anisotropic mesh generation. In *5th International Meshing Roundtable*, October 1996.

[BO05] Jean-Daniel Boissonnat and Steve Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67:405–451, 2005.

[BWY08] J.-D. Boissonnat, C. Wormser, and M. Yvinec. Locally uniform anisotropic meshing. In *Proc. of the 24th Symposium on Computational Geometry*, pages pages 270–277. ACM Press, 2008.

[BWY11] Jean-Daniel Boissonnat, Camille Wormser, and Mariette Yvinec. Anisotropic Delaunay Mesh Generation. Technical Report RR-7712, INRIA, August 2011.

[CDRW06] Siu-Wing Cheng, Tamal K. Dey, Edgar A. Ramos, and Rephael Wenger. Anisotropic surface meshing. In *In SODA'06 : Proc. of the 17th annual ACM-SIAM symposium on Discrete algorithm*, pages 202–211, New York, NY, USA, 2006. ACM.

[CG11] Guillermo D. Canas and DSteven J. Gortler. Orphan-free anisotropic Voronoi diagrams. *Discrete and Computational Geometry*, 46(3), 2011.

[CGA] CGAL. Computational Geometry Algorithms Library. http://www.cgal.org.

[Che93] L. Paul Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of the ninth annual symposium on Computational geometry*, SCG '93, pages 274–280, New York, NY, USA, 1993. ACM.

[CP05] Frédéric Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005.

[DL09] T. K. Dey and J. A. Levine. Delaunay meshing of piecewise smooth complexes without expensive predicates. *Algorithms*, 2(4):1327–1349, 2009.

[DS89] E. F. D'Azevedo and R. B. Simpson. On optimal interpolation triangle incidences. *SIAM J. Sci. Statist. Comput.*, 10(6):1063–1075, 1989.

[DW05] Q. Du and D. Wang. Anisotropic centroidal voronoi tessellations and their applications. *SIAM Journal on Scientific Computing*, 26(3):737–761, 2005.

[HG99] P. S. Heckbert and M. Garland. Optimal triangulation and quadric-based surface simplification. *Computational Geometry*, 14:49–65, 1999.

[Hug03] J.F. Hughes. Differential geometry of implicit surfaces in 3-space-a primer. Report, Department of Computer Science Brown University Provuidence, Rhode Island, 2003.

[JCNH06] X. Jiao, A. Colombi, X. Ni, and John C. Hart. Anisotropic mesh adaptation for evolving triangulated surfaces. In *15th International Meshing Roundtable*, pages 173–190, September 2006.

[Li03] Xiang-Yang Li. Generating well-shaped d-dimensional delaunay meshes. *Theor. Comput. Sci.*, 296(1):145–165, 2003.

[LL00] Greg Leibon and David Letscher. Delaunay triangulations and voronoi diagrams for riemannian manifolds. In *Proc. of the 16th Symposium on Computational Geometry*, pages 341–349, 2000.

[LL10] B. Lévy and Y. Liu. Lp centroidal voronoi tessellation and its applications. *ACM Transactions on Graphics*, 29(4):119, 2010.

[LS03] Francois Labelle and Jonathan Richard Shewchuk. Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *SCG' 03 : Proc. of the 19th Symposium on Computational Geometry*, pages 191–200, New York, NY, USA, 2003. ACM Press.

[LT01] Xiang-Yang Li and Shang-Hua Teng. Generating well-shaped delaunay meshed in 3d. In *SODA '01: Proc. of the 12th annual ACM-SIAM symposium on Discrete algorithms*, pages 28–37. Society for Industrial and Applied Mathematics, 2001.

[LTÜ99] Xiang-Yang Li, Shang-Hua Teng, and Alper Üngör. Biting ellipses to generate anisotropic mesh. In *8th International Meshing Roundtable*, October 1999.

[Mir10] J-M. Mirebeau. Optimal meshes for finite elements of arbitrary order. *Constructive approximation*, 32(2):339–383, 2010.

[Sch08] Jessica Schoen. Robust, guaranteed-quality anisotropic mesh generation. Master's thesis, University of California at Berkeley, 2008.

[She02] Jonathan Richard Shewchuk. What is a good linear finite element? Interpolation, conditioning, anisotropy, and quality measures. In *http://www.cs.cmu.edu/~jrs/jrspapers.html*, Manuscript 2002.

[She05] J. R. Shewchuk. Star splaying: an algorithm for repairing Delaunay triangulations and convex hulls. In *SCG '05: Proc. of the 21st Symposium on Computational Geometry*, pages 237–246, New York, NY, USA, 2005. ACM.