Interleaving Delaunay Refinement and Optimization for Practical Isotropic Tetrahedron Mesh Generation

Jane Tournois, Camille Wormser, Pierre Alliez, Mathieu Desbrun

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Interleaving Delaunay Refinement and Optimization for Practical Isotropic Tetrahedron Mesh Generation

Jane Tournois — Camille Wormser — Pierre Alliez — Mathieu Desbrun

N° 6826

Fevrier 2009

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--6826--FR+ENG

Interleaving Delaunay Refinement and Optimization for Practical Isotropic Tetrahedron Mesh Generation

Jane Tournois* , Camille Wormser† , Pierre Alliez‡ , Mathieu Desbrun§

Thème SYM — Systèmes symboliques

Projet Geometrica

Rapport de recherche n° 6826 — Fevrier 2009 — 24 pages

Abstract: We present a practical approach to isotropic tetrahedral meshing of 3D domains bounded by piecewise smooth surfaces. Building upon recent theoretical and practical advances, our algorithm interleaves Delaunay refinement and mesh optimization to generate quality meshes that satisfy a set of user-defined criteria. This interleaving is shown to be more sparing in Steiner points' insertions than refinement alone, and to produce higher quality meshes than optimization alone. A careful treatment of boundaries and their features is presented, offering a versatile framework for designing smoothly graded tetrahedral meshes of complex geometries.

Key-words: mesh generation, Delaunay refinement, optimization

* INRIA
† ETH Zurich
‡ INRIA
§ Caltech

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Raffinement et optimisation de Delaunay pour la génération de maillages tétraédriques isotropes

Résumé : Ce rapport présente une méthode de génération de maillages tétraédriques isotropes pour des domaines 3D bornés par des surfaces lisses par morceaux. Le principe consiste à entrelacer raffinement de Delaunay et optimisation de Delaunay pour maximiser la qualité des maillages tout en satisfaisant un ensemble de critères définis par l'utilisateur. Ce procédé est expérimentalement plus parcimonieux en nombre de points de Steiner que le raffinement seul, et produit des maillages de meilleure qualité que l'optimisation appliquée comme post-traitement. Un traitement particulier est réservé à la gestion des bords et des arêtes vives pour obtenir un cadre générique pour la génération de maillages.

Mots-clés : Génération de maillages, raffinement de Delaunay, optimisation

1 Introduction

Meshing a domain consists in defining a concise set of simple elements whose non-overlapping union best describes the domain and its boundaries, while satisfying a series of criteria on element shapes and sizes. Most ubiquitous in computer animation and computational sciences is the need for unstructured isotropic tetrahedral meshes: these versatile geometric representations are used in finite element and finite volume simulations of physical phenomena as varied as material deformation, heat transfer, and electromagnetic effects. As the accuracy and stability of such computational endeavors heavily depend on the shape of the worst element [She02b], mesh element quality is a priority when conceiving a mesh generation algorithm. In this paper we introduce a robust, hybrid meshing algorithm to generate high-quality, graded isotropic tetrahedron meshes. Delaunay refinements and variational optimizations are interleaved in order to produce a discretization of the domain that meets a series of desired geometric and topological criteria, while offering smooth gradation of the resulting well-shaped tetrahedra.

1.1 Background

Most previous work aimed at generating isotropic tetrahedral meshes were designed around one (or more) of four basic concepts: packing, regular lattices, refinement, and optimization. While packing methods (including advancing front approaches) were initially favored, their relatively high computational complexity and lack of theoretical guarantees have spawned the investigation of alternative methods. Regular lattices (be they red-green tessellations or octrees) have, recently, been at the core of some of the fastest meshing techniques, as they provide a blazingly fast approach to mesh most of the domain. While smooth surface boundaries can also be efficiently handled with guaranteed minimum dihedral angles [LS07], the regularity of the mesh resulting from these methods (*i.e.*, the presence of preferred edge directions) induces severe aliasing effects in simulation.

Techniques combining Delaunay triangulation and refinement have received special attention due to their versatility and theoretical foundations. They have been used initially in 2D [Che89], then in 3D for polyhedral domains [NCC02], for smooth surfaces [Che93], for 3D domains bounded by smooth surfaces [ORY05, BOG02] and for 3D domains bounded by piecewise smooth surfaces [RY07, CDL07, CDR07]. They proceed by refining and filtering a 3D triangulation until a set of user-specified criteria is satisfied. Refining is achieved through iterative insertion of Steiner points, either inside the domain or on the domain boundary, to meet the desired criteria. A filtering process is also iteratively applied to cull simplices so that the triangulation restricted to the input domain tessellates the domain and that the boundary of this restricted triangulation approximates the domain boundary. This procedure becomes more delicate for piecewise-smooth surface inputs (a more general class of shape where the boundary is a collection of smooth patches meeting at potentially sharp creases), as sharp creases require additional care. Non-smooth regions subtending small angles add yet another level of difficulty for Delaunay refinement. Refinement techniques are usually judged on the quality of the resulting mesh elements and the scarcity of Steiner point insertion.

The quest for ever better quality meshes has also stimulated significant advancements in mesh optimization, through local vertex relocation to optimize a specific notion of mesh quality [ABE99], topological operations [CDE+00], or both [FOG97]. Further improvement of the mesh quality can be achieved by inserting additional vertices, and/or incorporating a rollback mechanism to undo previous

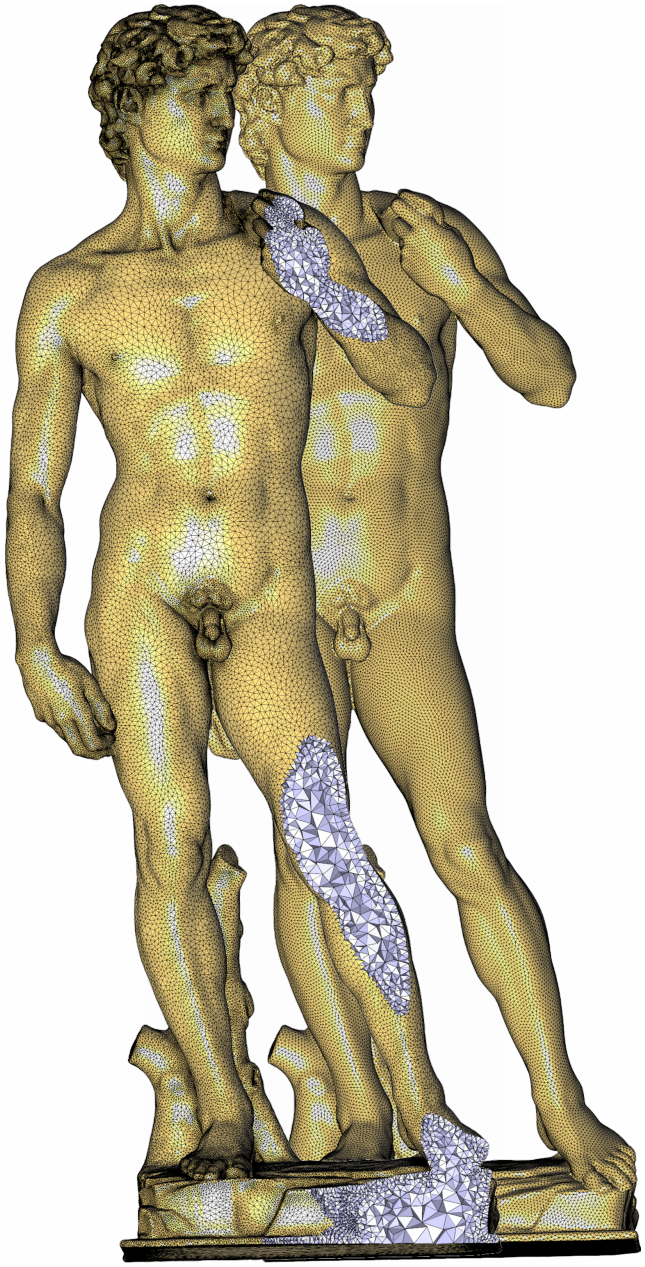


Figure 1: Michelangelo David. Our mesh generation algorithm produces high quality meshes through Delaunay refinements interleaved with Optimal Delaunay optimization. In this example, the input PSC has 800K triangles; on the right, illustrating scalability, a uniform sizing criterion generates a 1M vertices mesh; on the left, approximation error and shape criteria alone generate a smaller graded mesh (250K vertices), while guaranteeing the same quality of tetrahedra, and a better approximation error.

optimizations in order to guarantee a monotonous increase in mesh quality [KS07]. Among the large body of work in mesh optimization the *Optimal Delaunay Triangulation* approach (ODT for short) stands out, as it casts both geometric and topological mesh improvement as a single, unified functional optimization [CX04, Che04], that tries to minimize in $\mathbb{R}^4$ the volume between a paraboloid and the linear interpolation of the mesh vertices lifted onto the paraboloid. This approximation-theoretical method to obtain isotropic meshes was adapted for tetrahedral meshing of 3D domains [ACSYD05], mixed with a constrained Lloyd relaxation on the domain boundary. While this technique was shown to only produce nicely-shaped tetrahedra throughout the domain, slivers (i.e., nearly degenerate elements) could appear near the domain boundary, as the boundary vertices were guided by Lloyd relaxation and thus unaffected by the 3D optimization. Furthermore, this method lacks a number of useful features. First, the algorithm is not designed to satisfy the type of user-defined criteria commonly handled by Delaunay-based mesh generation techniques [RY07]. Also, an estimate of the boundary local feature size is required to derive a sizing function; such estimate turns out not to be always reliable depending on the tessellation of the input polyhedral domain boundary. Finally, this method cannot handle arbitrary boundary meshes as input, requiring a *restricted* Delaunay triangulation instead.

1.2 Contributions

In this paper, we combine the efficacy of Delaunay refinement methods with the isotropic quality induced by optimal Delaunay optimization techniques to provide a practical, high-quality meshing algorithm for domains bounded by piecewise smooth boundaries. This combination of techniques is motivated by the desire to maximize mesh quality while reducing mesh size. Delaunay refinement alone tends to generate overly complex meshes, with, *e.g.*, spurious clusters of vertices due both to the greedy nature of the algorithm and to the encroachment mechanisms; interleaving parsimonious refinement and mesh optimization instead turns out both to reduce the number of Steiner points and to improve the overall mesh quality (see Figure 2).

Unlike previous mesh optimization methods which either consider the boundary fixed or use boundary conditions incompatible with global mesh improvement, we introduce a consistent, unifying variational treatment applied to both interior and boundary nodes, improving the overall quality of the mesh. To speed up Delaunay refinement and make it parsimonious, we pick subsets of isolated Steiner points using the probabilistic multiple choice approach [WK02, VLV+04] to reduce the occurrence of short-lived primitives and provide independent refinements before each round of optimization. The practicality of our approach further stems from additional, distinctive features. First, we only rely on simple intersection tests to probe the domain boundary to make the approach as generic as possible with respect to the boundary surface representation. Second, we do not require a mesh sizing function as input and provide instead a dynamic sizing function which evolves throughout refinement until all user-specified criteria are satisfied. Finally, while we present a concrete implementation of our approach, the method is versatile enough to serve as a general framework for isotropic tetrahedron meshing, as each step involved in the process can be adapted to special requirements.

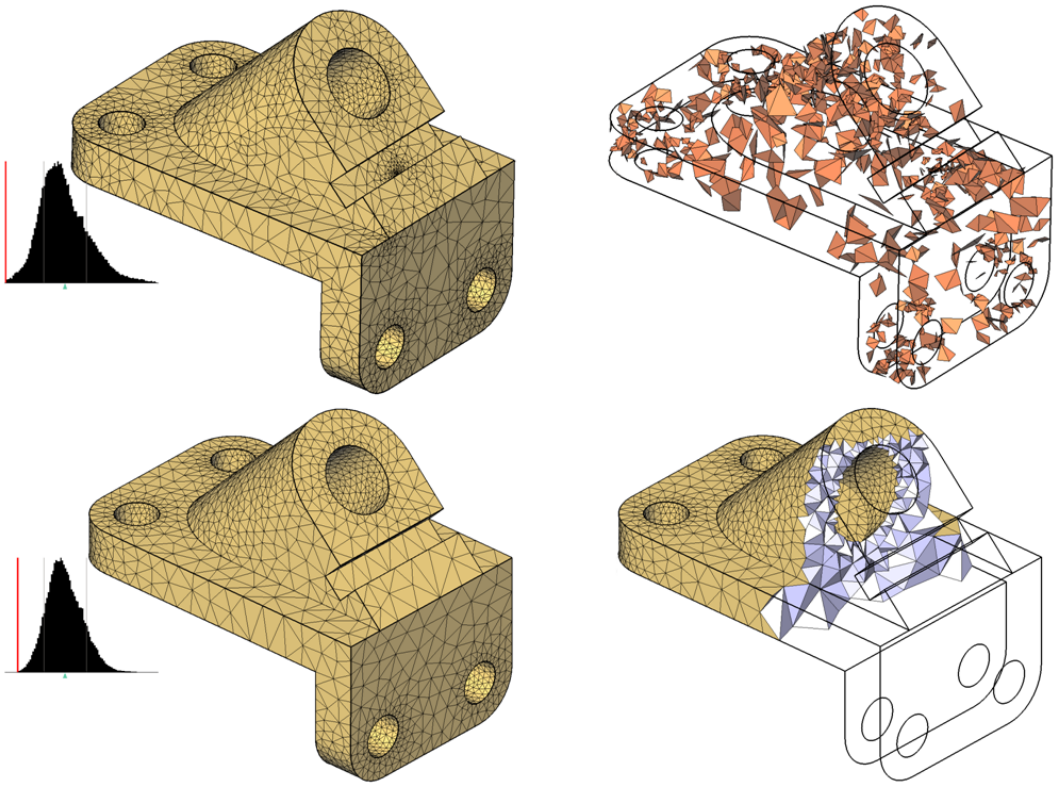


Figure 2: Top: Mesh generated by Delaunay refinement (shape and boundary approximation criteria activated). The mesh contains 5499 vertices, notice the cluster in the middle of the armhole. Right image depicts tetrahedra with dihedral angles lower than 15 degrees (all angles are in [0.45-179.1]). Bottom: Mesh generated by interleaving batches of Delaunay refinement and optimization so as to satisfy the same criteria. The mesh contains 3701 vertices and all dihedral angles are in [15.01-156.28]. Distribution of dihedral angles are shown on the left.

2 Algorithm

The algorithm we now detail interleaves refinement and optimization of an initial 3D Delaunay triangulation. Mesh simplices are gradually improved to meet user-defined criteria on boundary approximation and on the shape and size of elements through refinements, while passes of optimization further improve the shape of the elements. The high-level pseudo-code is as follows:

Algorithm 1 Mesh generation at a glance

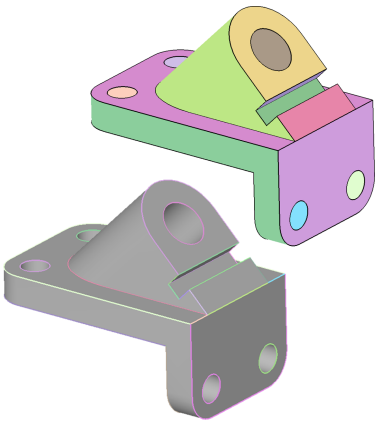
Require: *Domain* $\Omega \in \mathbb{R}^3$ *(Section 2.1)*
and a set $\{k_1, k_2, ..., k_n\}$ *of user-defined criteria (Section 2.2).*
Initialize coarse mesh $\mathcal{M}$ (Section 2.3)
while Refine through sparse vertex insertions (Section 2.4) **do**
Optimize mesh (Section 2.5)
end while
Perturb remaining slivers (Section 2.6)

The refinement procedure inserts Steiner points so as to satisfy the criteria $\{k_1, k_2, ..., k_n\}$. It returns true if at least one Steiner point has been inserted and false otherwise, so that the mesh is further optimized only when refined. This way, the algorithm benefits from the same guarantee of common Delaunay refinement algorithms.

2.1 Input

The input is a 3D domain Ω whose boundary is defined as a piecewise smooth complex (PSC). More specifically, our current implementation takes as input a piecewise linear approximation of a PSC.

The latter is provided as a triangle surface mesh, watertight, and forming a 2-manifold with no self-intersection. In addition, we assume that sharp edges as well as feature vertices of this mesh are tagged. Dart (resp., corner) vertices are easily deduced from tagged sharp edges as they are incident to one (resp., three or more) sharp edges. Tip and cusp vertices, which are incident respectively to zero and two sharp edges, cannot be derived solely using the sharp edge tags and hence must be specified by the user. By chaining sharp edges together, we obtain a set of polylines that we will refer to as *creases* from now on. A crease may either connect two feature vertices, or form a cycle. All creases are enumerated, and each sharp edge of the input surface mesh is marked with the index of its associated crease. Finally, we identify and enumerate surface *patches* as connected components of the boundary, bounded (or not) by sharp creases. Each face of the input surface mesh is marked with the index of its associated patch as depicted in the inset.



2.2 Parameters

The user can also input a number of desired criteria that the final mesh must satisfy. These criteria will be used to guide the refinement process as explained in Section 2.4. Our meshing framework can handle five types of criteria:

- *Sizing:* a spatially-varying sizing function (or possibly a single value if constant) indicates the maximum mesh edge length desired within the domain.
- *Approximation:* an approximation control function defines a local upper bound for the surface or crease approximation error, $\epsilon_{\max}$. Similarly to the mesh sizing function, it can be defined as a single value if the function is constant over the boundary, or as a spatially-varying scalar function.
- *Shape:* two global element shape quality bounds are defined as the maximum circumradius to shortest edge ratio allowed in the final mesh. We denote by ${\sigma^f}_{\max}$ and ${\sigma^t}_{\max}$ these bounds for facets and tetrahedra, respectively.
- *Topology:* a Boolean flag determines whether the topology of the input PSC should be preserved, *i.e.*, if the vertices of each restricted facet must belong to the same patch, and the vertices of each restricted edge must belong to the same crease.
- *Manifold:* a Boolean flag determines whether the final mesh boundary should be a two-manifold surface.

These criteria accommodate the typical user requirements for mesh generation. Note that they are all optional in our implementation, except for the *sizing* field.

2.3 Initialization

A first mesh $\mathcal{M}$ of the domain is obtained as follows. We begin by inserting in $\mathcal{M}$ all *feature vertices* (corners and such) of the input surface mesh. These vertices remain untouched throughout the mesh generation procedure. We also add the eight corners of a large bounding box of the input domain, in order not to have to deal with infinite Voronoi cells in later stages. Finally, we ensure that each surface patch and each crease have received the minimal number of sample points to seed the refinement process by adding more vertices if necessary, as in [RY07]. The mesh $\mathcal{M}$ is defined to be the Delaunay mesh of all these vertices. Finally, we refine this initial mesh with respect to looser criteria than the ones defined by the user (typically, we halve the various input criteria parameters), using our refinement procedure that we detail next.

2.4 Refinement

The refinement process is entirely driven by the user-defined criteria listed in Section 2.2. Each refinement step is designed to remove a set of *bad elements* (simplices that do not satisfy at least one of the given criteria) by inserting so-called *Steiner vertices* to $\mathcal{M}$. Unlike typical Delaunay refinement techniques which insert one Steiner point at a time, we proceed in batches of refinement, inserting a sparse subset of all the candidate Steiner points per batch.

2.4.1 Definition of Bad Elements

The simplices considered for refinement are the so-called *restricted simplices*, that is, the ones considered as inside the domain Ω or on the domain boundary $\partial\Omega$—namely, edges whose dual Voronoi facet intersects an input crease, facets whose dual Voronoi edge intersects the domain

boundary, and tetrahedra whose dual Voronoi vertex is located inside the domain. We consider one of these restricted elements bad if it violates one of the following criteria.

Size. A restricted edge is considered bad if it is longer than the sizing function evaluated at its midpoint. A restricted facet or tetrahedron is considered bad if at least one of its edges is badly sized.

Approximation error. A restricted edge e is considered bad if the distance from its midpoint to the farthest intersection point between its dual Voronoi face and an input crease is larger than the local approximation bound. Similarly, a restricted facet f is considered bad if the distance from f's circumcenter to the farthest intersection point between its dual Voronoi edge and the domain boundary is larger than the approximation bound.

Shape. A restricted facet (resp., tetrahedron) is considered bad if the ratio of its circumradius to shortest edge is higher than the user-specified bound $\sigma^f_{\max}$ (resp., $\sigma^t_{\max}$).

Topology. A restricted edge (resp., facet) is considered as not capturing the proper topology if its two (resp., three) vertices do not belong to the same input crease (resp., surface patch). If the topology criterion is activated, we store for each vertex v of the mesh its location with respect to the input PSC. That is, each vertex is tagged either as an *interior* vertex, a *feature* vertex, a *crease* vertex, or a *boundary* vertex. In the last two cases, the index of the feature (crease or surface patch) is stored too.

In addition to these types of bad elements, we add an extra one to enforce the *topological disk condition* [RY07] as it is an important indicator of topological conformity of the mesh to the input domain. For a vertex v tagged as boundary (*i.e.*, on an input surface patch), the topological disk condition is satisfied iff the boundary facets incident to v form a topological 2-disk. If v belongs to an input crease, its incident restricted edges (edges whose dual Voronoi facet intersects input creases) have to form a topological 1-disk. We thus mark every boundary vertex of the mesh whose topological disk condition is not satisfied as bad as well.

2.4.2 Steiner Vertices

For each bad simplex, we define its associate Steiner point location. The associated Steiner point to a restricted *edge* is the farthest intersection point between its dual Voronoi face and the input creases. The associated Steiner point to a restricted *facet* is the farthest intersection point between its dual Voronoi edge and the input domain boundary. The associated Steiner point to a restricted tetrahedron is its circumcenter. Finally, for each boundary vertex of the mesh whose topological disk condition is not satisfied, we define its associated Steiner point to be the Steiner point of the facet (resp., crease edge) incident to v that realizes the largest approximation error: its insertion will help enforcing the topological disk condition.

To ensure termination of the refinement process, we further check for *encroachment* [Rup95, She98, She02a, CDL07, RY07]. The Steiner point p of a tetrahedron, candidate for insertion, is said to *encroach* a boundary facet f if it is inside its restricted Delaunay ball (centered at f's Steiner point and passing through the vertices of f). Similarly, the Steiner point of a facet is said to *encroach* a crease edge if it is inside its restricted Delaunay ball (centered at its Steiner point and passing through its endpoints). In these two cases of encroachment, we alter the position of the associated Steiner

point, replacing it by the Steiner point of the encroached primitive (and recursing the encroachment check).

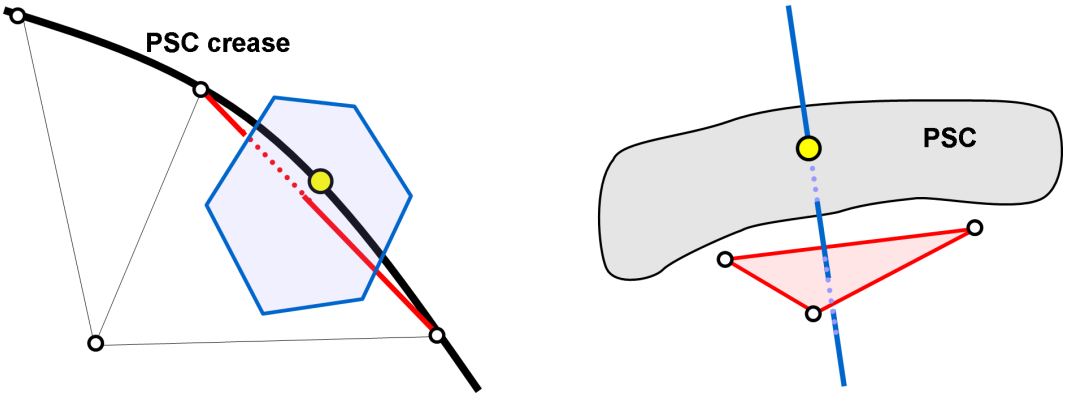
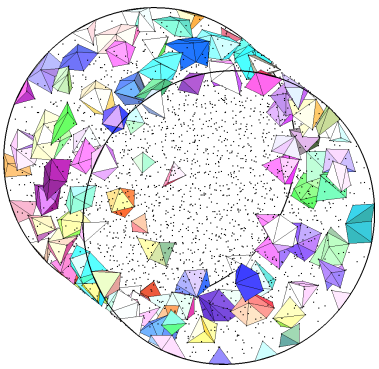


Figure 3: Left: Steiner point (yellow) of a restricted edge (red) computed as the furthest intersection point of its dual Voronoi facet (blue) with the input PSC creases. Right: Steiner point (yellow) of a restricted facet (red) computed as the furthest intersection point of its dual Voronoi edge (blue) with the input PSC surface.

2.4.3 Independent Set Refinement

To help define a good subset of Steiner points to add in batch, we introduce the notion of *conflict regions* and *independent set of conflict regions*. For each Steiner point p, we call "conflict region" the tets that would be affected by its insertion as well as their adjacent tets, since these elements are likely to be destroyed by the insertion of p. We call "an independent set" of conflict regions a set that does not contain overlapping conflict regions, so that none of the insertions of these selected Steiner points would influence each other.



We construct such an independent set of conflict regions as described in Algorithm 2: we iteratively select Steiner points *in order of increasing dimension* of their associated simplices. That is, first crease edges are collected and sorted from worst to best. As many crease-edge Steiner points as possible are inserted into the set, along with their conflict region, while making sure there is no overlap of conflict regions. Second, we similarly treat boundary facets. Finally, tetrahedra are handled; however, as there can be a large amount of bad tets during the meshing process, the same process of sorting elements before choosing them would be too costly. We therefore process bad tetrahedra through a more efficient Multiple-Choice approach

as explained next, and this is done iteratively until no Steiner point can be inserted to the independent set without overlapping the regions already inserted. Inset shows an independent set on the mesh of a cylinder for which only the approximation criterion is not yet satisfied.

Multiple-Choice Selection of Tetrahedra Although many Delaunay refinement algorithms use modifiable priority queues to store all bad simplices of the mesh $\mathcal{M}$, most queue elements are short-lived as each Steiner position insertion affects its surrounding. In fact, our experiments consistently showed that the computational burden spent maintaining the global priority queue of all bad simplices is overly high compared to the number of primitives actually refined. We thus depart from the usual refinement strategy by using a multiple choice approach (proposed for mesh decimation in [WK02, VLV$^+$04]) as follows. At each step, a small container of N_{mc}"bad" tetrahedra (N_{mc}= 20 in our implementation), randomly updated among the non-conflicted tetrahedra, is used to select the worse tetrahedron. As our goal is to only sparingly refine the mesh before further optimization, this multiple-choice approach significantly speeds up our refinement process.

Algorithm 2 Independent Set of Conflict Regions Construction

Require: *A PSC as input domain,*
a coarse initialization of the mesh,
and a set $K = \{k_i\}_i$ of criteria to be met.

Collect all restricted (inside) tetrahedra.
Collect all bad crease edges in E_{bad} and all bad boundary facets in F_{bad} w.r.t. K.
for each bad simplex s in E_{bad} and F_{bad} (from worst to best), **do**
  Let p the Steiner point of s.
  Compute the union U_c of tetrahedra that are in direct conflict with the insertion of p.
  Compute the union U_n of tetrahedra sharing a facet with one of the conflicted tetrahedra in U_c.
  if No tetrahedron of $U_c \cup U_n$ is in the Independent Set, **then**
    Insert the conflict region $U_c \cup U_n$ in Independent Set, along with p.
  end if
end for
Let N_{mc}be the size of C_{mc}a container of tetrahedra.
while There are non-conflicted tetrahedra **do**
  Fill $C_{mc}$with N_{mc}tetrahedra chosen by multiple-choice, whose Steiner point conflict zones do not intersect the tetrahedra in independent set.
  Insert worst tetrahedron of C_{mc}'s Steiner point, along with its conflict tetrahedra, in independent set.
end while
Batch-insert all Steiner points of independent set to mesh.
Update restricted Delaunay triangulation.

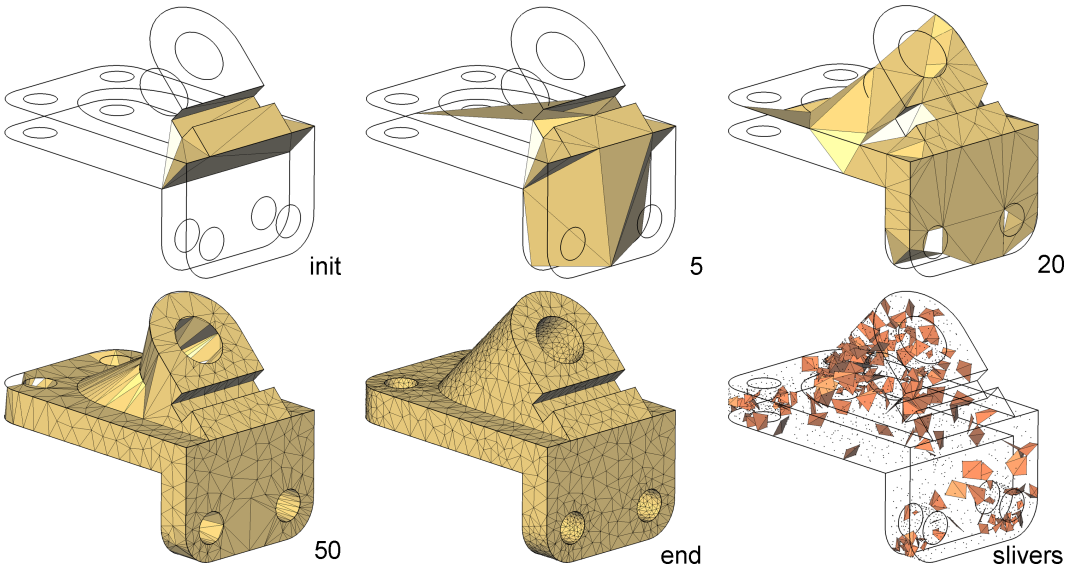


Figure 4: Refinement steps. From left to right and top to bottom: The mesh initialized with feature vertices; after a few batch refinement steps (from 5 to 50); the final refined mesh with shape and approximation criteria satisfied; and its 244 slivers (tetrahedra with dihedral angle smaller than 10 degrees).

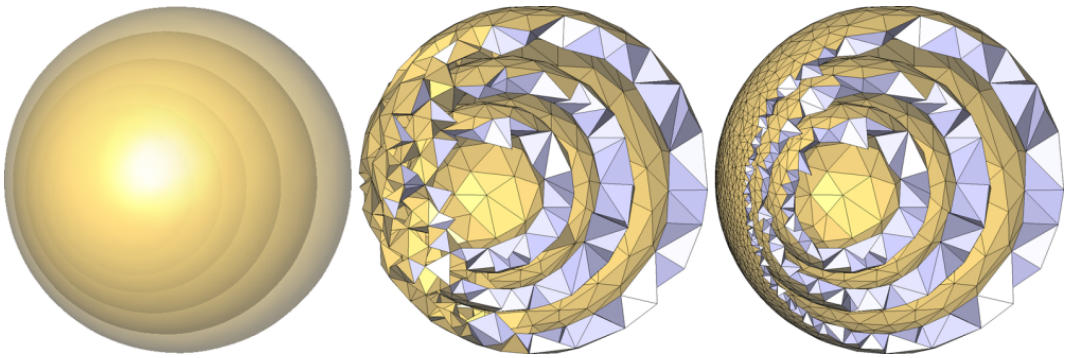


Figure 5: Nested spheres. Left: input PSC. Middle: mesh generated by refinement with $l_{max} = 1$, $\epsilon_{max} = 0.03$ and topology criterion not activated. Right: mesh further refined with same criteria but with topology activated.

2.5 Optimization

Chen [Che04] defines an *Optimal Delaunay Triangulation (ODT)* as the minimizer of the energy

$$E_{\text{ODT}} = ||f_{\text{PWL}} - f||_{\mathcal{L}^1} = \sum_j \int_{T_j} |f_{\text{PWL}} - f|,$$

where $f(\mathbf{x}) = ||\mathbf{x}||^2$ and f_{PWL} is the linear function interpolating the values of f at the vertices of the tets T_j's of the triangulation. This energy has a simple geometric interpretation: it is the volume between the 4D paraboloid (defined by f and its inscribed piecewise linear approximation f_{PWL} through lifting off the triangulation onto the paraboloid [BWY07]. Because of a result of function approximation theory stating that the best interpolating approximation of a function is achieved when the elements' size and orientation match the Hessian of the function, an ODT is thus isotropic (see Fig 6).

By integrating this function over each tet and summing all the contributions, one gets:

$$E_{\text{ODT}} = \frac{1}{4} \sum_{x_i \in T_j} {\mathbf{x}_i}^2 |\Omega_i| - \int_{\mathcal{M}} \mathbf{x}^2 d\mathbf{x}, \tag{1}$$

where $|\Omega_i|$ is the volume of the 1-ring neighborhood of vertex $\mathbf{x}_i$. Noting that the last term is constant given a fixed boundary $\partial\mathcal{M}$, a simple derivation of this quadratic energy in $\mathbf{x}_i$ leads to the following *optimal position* $\mathbf{x}_i^\star$ of the interior vertex $\mathbf{x}_i$ in its 1-ring [Che04]:

$$\mathbf{x}_i^\star = -\frac{1}{2\,|\Omega_i|} \sum_{T_j \in \Omega_i} \left(\nabla_{\mathbf{x}_i} |T_j| \left[\sum_{\substack{\mathbf{x}_k \in T_j \\ \mathbf{x}_k \neq \mathbf{x}_i}} ||\mathbf{x}_k||^2 \right] \right). \tag{2}$$

The term $\nabla_{\mathbf{x}_i} |T_j|$ is the gradient of the volume of the tet T_j with respect to $\mathbf{x}_i$. Replacing the paraboloid function $f(\mathbf{x}) = ||\mathbf{x}||^2$ by the translated function $f(\mathbf{x}) = ||\mathbf{x} - \mathbf{x}_i||^2$, does not change the interpolation error, leading to the same optimal position. We thus get the following equivalent expression used to update a vertex position :

$$\mathbf{x}_i^\star = \mathbf{x}_i - \frac{1}{2\,|\Omega_i|} \sum_{T_j \in \Omega_i} \left(\nabla_{\mathbf{x}_i} |T_j| \left[\sum_{\mathbf{x}_k \in T_j} ||\mathbf{x}_i - \mathbf{x}_k||^2 \right] \right). \tag{3}$$

We also know that $\sum_{T_j \in \Omega_i} \nabla_{\mathbf{x}_i} |T_j| = 0$, it thus follows that when all $||\mathbf{x}_i - \mathbf{x}_k||^2$ are equal, $\mathbf{x}_i^\star = \mathbf{x}_i$. In other words, when the neighbors of $\mathbf{x}_i$ lie on a sphere with center $\mathbf{c}$, $\mathbf{x}_i^\star = \mathbf{c}$; we call this property the *ODT circumsphere property*.

As a special case of this property, the optimal position of a vertex that has only four neighbors is exactly *at* $\mathbf{c}_T$. Using Eq. (3) in this special case of a 1-ring in the shape of a tet $T = (\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_r, \mathbf{x}_s)$,

and taking the point $\mathbf{x}_i$ to be located at $\mathbf{x}_p$, we get:

$$\mathbf{c}_T = \mathbf{x}_p - \frac{1}{2\,|T|}\Bigg[\nabla_{\mathbf{x}_p}|T|\,\Big[\sum_{\mathbf{x}_k\in T}||\mathbf{x}_p-\mathbf{x}_k||^2\Big] + \mathrm{F}(\mathbf{x}_p,\mathbf{x}_q,\mathbf{x}_r) + \mathrm{F}(\mathbf{x}_p,\mathbf{x}_q,\mathbf{x}_s) + \mathrm{F}(\mathbf{x}_p,\mathbf{x}_r,\mathbf{x}_s)\Bigg] \tag{4}$$

where the extra terms on the rhs only depend on each face of the tet (because, as we took $\mathbf{x}_i$ to be at $\mathbf{x}_p$, all but one of the tets inside T are degenerate and become *faces* of T). More precisely, these terms are explicitly given as:

$$\mathrm{F}(\mathbf{x}_p,\mathbf{x}_q,\mathbf{x}_r) = +\frac{1}{3}\Big[||\mathbf{x}_p-\mathbf{x}_q||^2 + ||\mathbf{x}_p-\mathbf{x}_r||^2\Big]\mathbf{N}_{p,q,r}$$

where $\mathbf{N}_{p,q,r}$ is the area-weighted normal of the face (p,q,r) pointing towards the inside of the tet, *i.e.*, $\mathbf{N}_{p,q,r} = |(p,q,r)|\mathbf{n}_{p,q,r}$.. Now, go back to Eq. 3 for an arbitrary 1-ring centered on $\mathbf{x}_p$, and note that the term in parenthesis appears as is (for $p \equiv i$) in Eq. 4. Substitute this term by the circumcenter and all the other terms that Eq. 4 contains. All the face terms F cancel each other out, thus simplifying the expression to:

$$\mathbf{x}_i^\star = \frac{1}{|\Omega_i|}\sum_{T_j\in\Omega_i}|T_j|\mathbf{c}_j. \tag{5}$$

Natural ODT for Boundary Vertices While [Che04, ACSYD05] do not involve the boundary vertices in the minimization of the ODT energy, we propose an extension that changes the update of boundary vertices during optimization so as to further reduce the total energy, thus providing a boundary extension to the original ODT mesh smoothing procedure. Denote by $\mathbf{x}_p$ a vertex on the *boundary* of a 3D mesh (i.e., it does not have a full 1-ring $\mathcal{N}(\mathbf{x}_p)$ of restricted (inside) tetrahedra. For a given connectivity, the new position $\mathbf{x}_p^\star$ of $\mathbf{x}_p$ that extremizes the ODT energy is a bit more complicated, as some of the face terms F do *not* disappear:

$$\mathbf{x}_p^\star = \left[\left(\sum_{T\in\mathcal{N}(\mathbf{x}_p)}|T|\mathbf{c}_T\right) + B\right] \Big/ \sum_{T\in\mathcal{N}(\mathbf{x}_p)}|T|,$$

where the boundary terms B are

$$B = \frac{1}{6}\left(\sum_{(p,q,r)\in\partial\mathcal{M}}\mathbf{N}_{p,q,r}\left[||\mathbf{x}_p-\mathbf{x}_q||^2 + ||\mathbf{x}_p-\mathbf{x}_r||^2\right]\right).$$

(Again, by $\mathbf{N}_{p,q,r}$, we mean the vector that is along the normal of the face (pointing toward the inside of the domain), with a magnitude equal to the area of the triangle.) The first part ($|T|\mathbf{c}_T$) is the weighted barycenter of the circumcenters and is divided by the total 1-ring volume just as before.

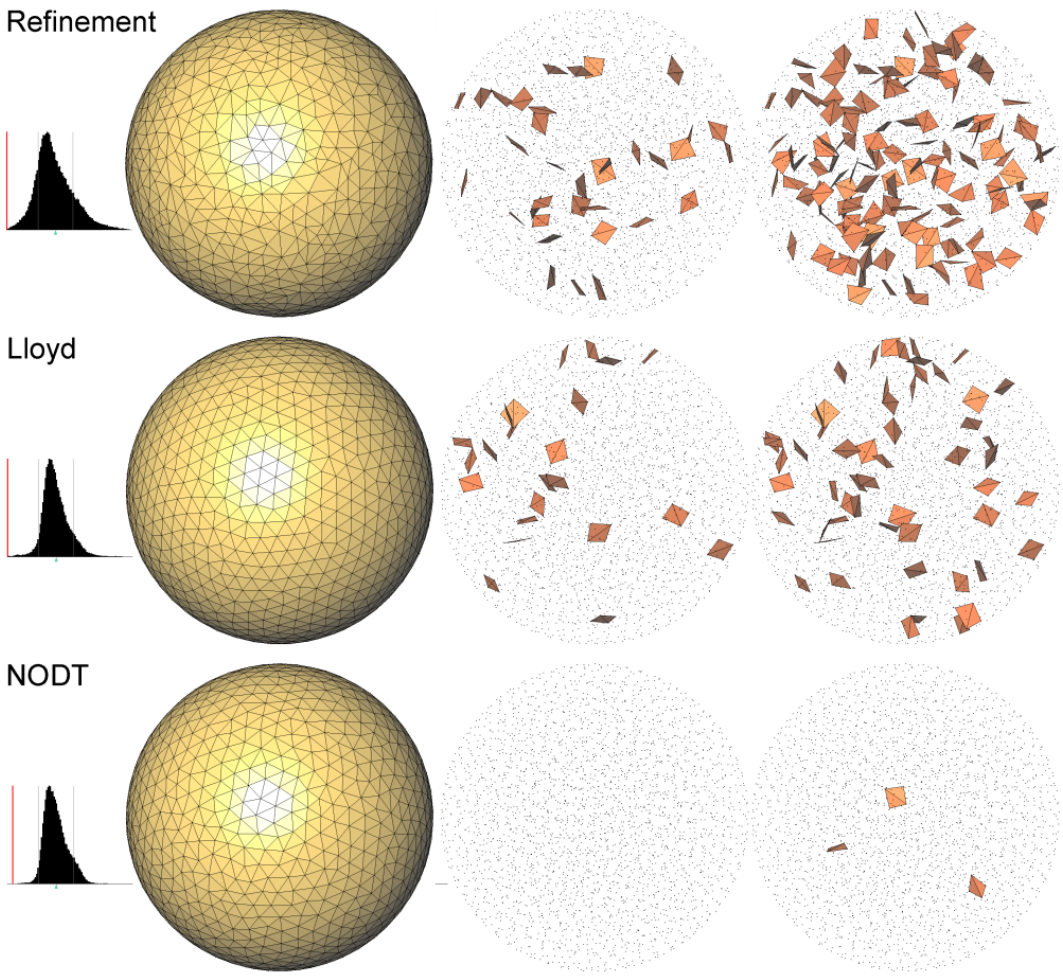


Figure 6: Comparing Delaunay refinement and mesh optimization. Distributions of dihedral angles are shown to the left. Slivers are shown for a dihedral angle bound of respectively 5 (middle) and 10 degrees (right). Top: Delaunay Refinement alone (resp. 35 and 136 slivers). Middle: Optimized mesh with 100 Lloyd iterations (resp. 23 and 55 slivers). Bottom: Optimized mesh with 100 NODT iterations (resp. 0 and 3 slivers).

Note however that there is an extra term: a sum on triangles (p, q, r) that are on the boundary of the domain, involving the squared length of the "spokes" of the triangle 1-ring.

This formula, applied as is, shrinks the domain as it obviously decreases the total energy. However, as seen previously, we can add a multiplicative weight to the supplementary terms B without changing the update rule in the case of a full 1-ring, because they cancel each other out anyway:

$$\mathbf{x}_p^\star = \left[\left(\sum_{T\in\mathcal{N}(\mathbf{x}_p)} |T|\mathbf{c}_T\right) + \lambda B\right] / \sum_{T\in\mathcal{N}(\mathbf{x}_p)} |T|.$$

We now use this flexibility to choose λ so that we retain the *ODT circumsphere property* mentioned earlier, but now in the case of an incomplete 1-ring: if all neighbors of $\mathbf{x}_p$ are at the same distance from $\mathbf{x}_p$, we want $\mathbf{x}_p^\star = \mathbf{x}_p$: we will thus obtain a formula valid for both the complete 1-ring and incomplete 1-ring cases, while preserving the ODT circumsphere property. We have

$$\mathbf{x}_i^\star = \mathbf{x}_i - \frac{1}{2|\Omega_i|}\sum_{T_j\in\Omega_i}\left(\nabla_{\mathbf{x}_i}|T_j|\Big[\sum_{\mathbf{x}_k\in T_j}||\mathbf{x}_i-\mathbf{x}_k||^2\Big] + (1-\lambda)\sum_{r,s}\mathrm{F}(\mathbf{x}_i,\mathbf{x}_r,\mathbf{x}_s)\right). \tag{6}$$

Consider the case where all $||\mathbf{x}_i - \mathbf{x}_k||^2$ are equal to some constant R. We want $\mathbf{x}_i^\star = \mathbf{x}_i$ and we know, from the divergence theorem applied on the 1-ring of the boundary vertex, that

$$\nabla_{\mathbf{x}_i}|T_j| + \sum_{r,s}\frac{1}{3}\mathbf{N}_{p,q,r} = 0.$$

On the one hand, $\nabla_{\mathbf{x}_i}|T_j|$ is weighted by $3R$ in (6). On the other hand, each $\frac{1}{3}\mathbf{N}_{p,q,r}$ is weighted by $2R$ in (6). It follows that we need $(1-\lambda) = 3/2$ for each term

$$\left(\nabla_{\mathbf{x}_i}|T_j|\Big[\sum_{\mathbf{x}_k\in T_j}||\mathbf{x}_i-\mathbf{x}_k||^2\Big] + (1-\lambda)\sum_{r,s}\mathrm{F}(\mathbf{x}_i,\mathbf{x}_r,\mathbf{x}_s)\right)$$

to vanish.

Hence, for $\lambda = -1/2$, the ODT circumcenter property is still enforced on the boundary. In particular, the optimal position for this method (denoted NODT for *Natural ODT*) is computed as

$$\mathbf{x}_p^\star = \left[\left(\sum_{T\in\mathcal{N}(\mathbf{x}_p)} |T|\mathbf{c}_T\right) - \frac{1}{2}B\right] / \sum_{T\in\mathcal{N}(\mathbf{x}_p)} |T|,$$

where the boundary terms B (if any) are

$$B = \frac{1}{6}\left(\sum_{(p,q,r)\in\partial\mathcal{M}} \mathbf{N}_{p,q,r}\left[||\mathbf{x}_p - \mathbf{x}_q||^2 + ||\mathbf{x}_p - \mathbf{x}_r||^2\right]\right).$$

Variable Sizing The optimization formula above is only valid to generate uniform isotropic meshes. To account for a variable mesh sizing, we update a dynamic mesh sizing function after each batch of refinement, and replace all measures in above formulas (lengths, areas, volumes) by measures in the metric of this sizing function. Such measures are obtained by quadratures over the mesh elements. The dynamic sizing function [ADA07] is guaranteed to be K-Lipschitz and is obtained by averaging the lengths of the mesh edges incident to all mesh vertices. Intuitively, the refinement is in charge of discovering the local feature size of the domain boundary. One of the user-defined criteria triggers a local refinement of the mesh, which induces the updating of the sizing function which, in turn, imposes further refinements to maintain the K-grading of the mesh. The optimization part of the algorithm then takes the current sizing function as input, so that it does not smooth out the increased density brought by the refinement steps to satisfy the user-defined criteria.

Restriction and Projection In practice, as we want the mesh to interpolate the domain, each boundary vertex of the mesh should be on the boundary $\partial\Omega$ of the PSC domain. To enforce this property, the new location $\mathbf{x}_p^\star$ of $\mathbf{x}_p$ is projected onto $\partial\Omega$. Two cases are distinguished: $\mathbf{x}_p^\star$ can belong to a surface patch, or to a sharp feature of the mesh. If at least one of the incident edges to $\mathbf{x}_p$ is a *crease edge* (i.e., its dual Voronoi facet intersects a PSC crease), then we project $\mathbf{x}_p^\star$ onto the closest crease. Similarly, if at least one of the incident facets to $\mathbf{x}_p$ is a boundary facet (i.e., its dual Voronoi edge intersects the PSC), we project $\mathbf{x}_p^\star$ onto the closest facet of the input PSC.

2.6 Sliver Removal

While our NODT boundary treatment significantly reduces the number of slivers compared to the results reported in [ACSYD05], we cannot guarantee a total absence of slivers (see Figure 11). We thus perform a final phase of sliver removal. We implemented an explicit perturbation inspired by [Li00], which performed better and faster than sliver exudation [CDE$^+$00] in our experiments. This method applies repeatedly a small perturbation of each vertex incident to slivers. Vertices located on sharp creases or boundary are then reprojected onto their respective crease or boundary. This relocation is validated if it both reduces its number of incident slivers and preserves the restricted triangulation locally. This process is iterated a number of times (max. 100 in our implementation) for each vertex incident to one or more slivers, and interrupted earlier if all incident slivers are removed. Note that for completeness we have also applied such perturbation after Delaunay refinement and Lloyd-based optimization. In these meshes many slivers remain due to the presence of chains of slivers, i.e., several slivers incident to each other.

2.7 Further Implementation Details

Our algorithm is implemented both in C++ 32 bits and C++ 64 bits for large models, using the *Computational Geometry Algorithms Library* CGAL. We use its 3D regular (weighted Delaunay) triangulation as our core data structure. The input PSC is represented as a surface triangle mesh with attributes.

One crucial component for reaching good timings is the efficient update of the restricted triangulation and Steiner points: this requires many intersection tests between rays and Voronoi edges and the input domain boundary, as well as intersections between Voronoi faces and the input sharp creases. We have implemented a collision detection library based on the principles used in OPCODE [Ter05]. Two hierarchies of axis-aligned-bounding-boxes (AABBs) are created right after loading the input PSC: one for the PSC triangle facets and one for the PSC segment sharp creases. Each intersection query (be it a test or an exhaustive enumeration) then calls intersection with AABBs during traversal, and intersection with PSC primitives (triangle or crease segments) at the leaves of the tree (see [dB05]). In addition, the same AABB trees are used for projecting the optimized boundary vertices onto the domain boundary or creases. The trees are this time queried with 3D balls whose radius decreases during the tree traversal. In our tests, these hierarchies of AABBs result in the fastest intersection and projection routines on average. Finally, we also speed up the NODT procedure through a locking process. We lock up (i.e., deactivate the optimization of) all mesh vertices which are incident to only excellent restricted tetrahedra. A tetrahedron is defined as excellent when all its dihedral angles are within a user-specified interval (typically [45-95]). Only the vertices newly inserted during refinement or relocated during optimization are allowed to unlock their incident vertices. Consequently, entire parts of the mesh which do not need to be improved either by refinement or by optimization are skipped throughout the refinement/optimization alternation. Tuning the interval bounds which qualify excellent tetrahedra is our mean to trade efficiency for the final mesh quality.

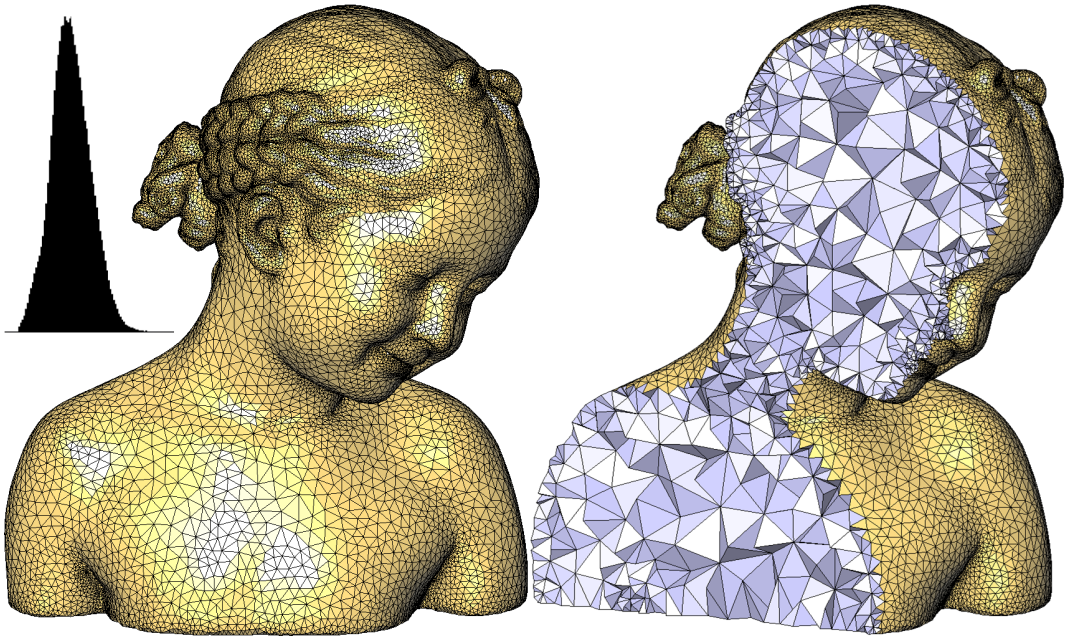


Figure 7: Bimba. Mesh generated by interleaved refinement and optimization with $l_{max} = 0.1$, $\epsilon_{max} = 0.0005$.

3 Results

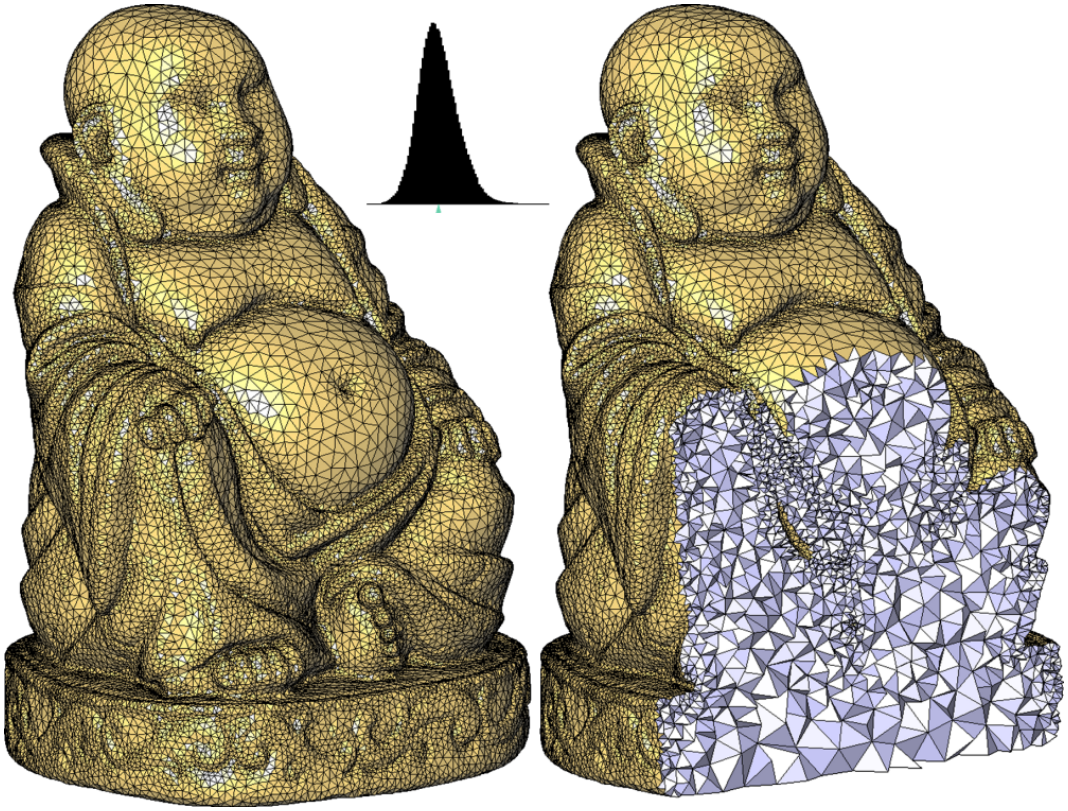


Figure 8: Buddha. Mesh obtained by interleaved refinement and optimization. All dihedral angles are above 15 degrees.

To evaluate our approach, we tested the various steps of our algorithm separately, then together. Figure 4 shows our refinement routine when no optimization step is performed. Notice that the resulting mesh lacks gradation, as typical for Delaunay refinement methods. We compare results of Delaunay refinement, Lloyd relaxation [DFG99], and our NODT in terms of number of slivers (*before* sliver removal for fairness) in Figure 6. Figure 11 shows the mesh of an elephant model obtained by Delaunay refinement (top) as it gets optimized by our NODT routine (middle), then after sliver removal (bottom).

Figure 7 shows the mesh of the bimba model obtained by interleaved refinement and optimization with approximation and element quality criteria activated (the sizing criterion $l_{max} = 0.1$ is not significant as the input PSC fits into a unit bounding box). The mesh contains 43K vertices and all

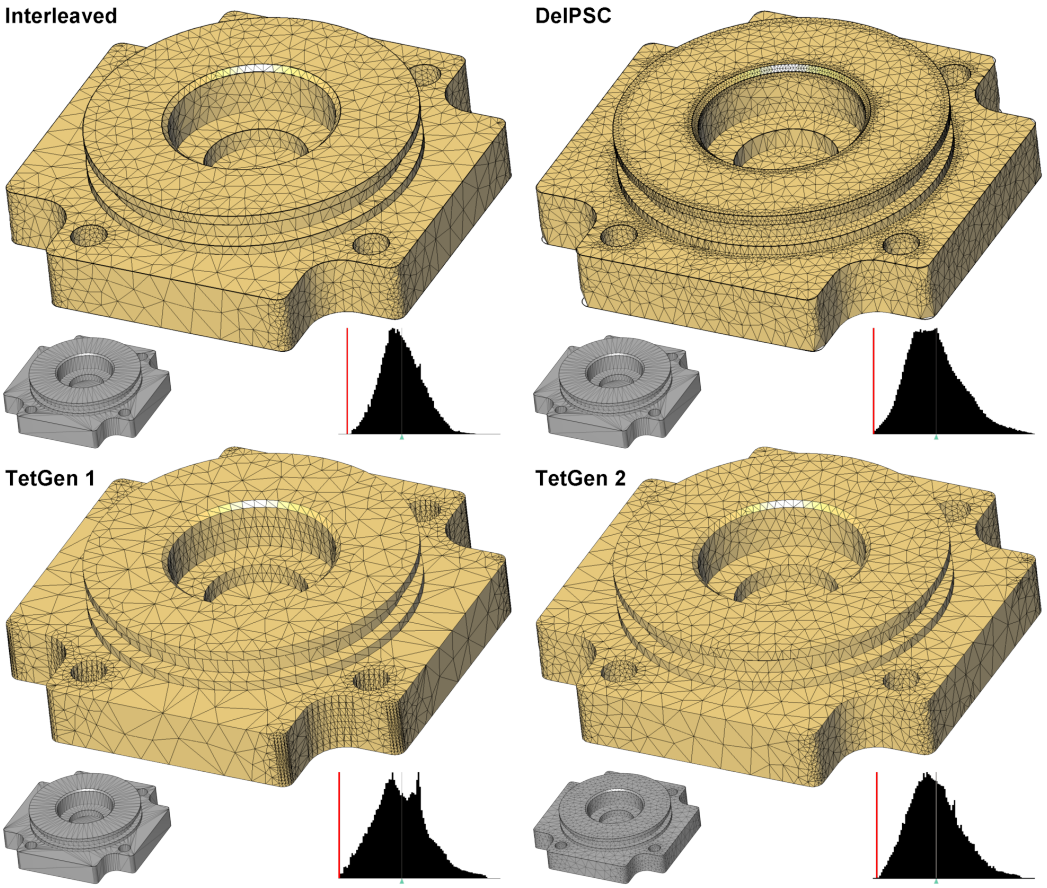


Figure 9: Cover rear. Top left: mesh obtained by interleaved refinement and optimization with $l_max = 0.1$, $\epsilon_{max} = 0.001$, $\sigma^f_{\max} = 1.5$, $\sigma^t_{\max} = 1.5$ and the topology criterion activated. It contains 4,050 vertices and all dihedral angles are above 12.0 degrees. Top right: mesh generated by DelPSC, with the same parameters. It contains 15,157 vertices and all dihedral angles are above 0.1 degree. Bottom left: mesh generated by TetGen, with the same parameters and input. It contains 6,966 vertices and all dihedral angles are above 0.2 degree. Bottom right: mesh generated by TetGen, with the same parameters and the boundary of our optimized mesh taken as input. It contains 4,189 vertices and all dihedral angles are above 3.0 degrees.

dihedral angles are above 15 degrees. The input PSC has 400K vertices. We also tested our method on mechanical parts. Figure 10 shows the mesh of a turbine generated by our interleaved algorithm. The mesh contains significantly fewer vertices (13%) than Delaunay refinement alone.

We also compared our technique to DelPSC [CDL07] and TetGen [Si] in Figure 9. Our interleaved technique improves both over the mesh quality and complexity. For TetGen we provided as input both the input PSC (which is then refined) and the boundary of our optimized mesh for fair comparison. Figure 8 shows the mesh of the buddha model obtained by interleaved refinement and optimization. This example illustrates the mesh of a domain boundary with larger range of feature size.

Activating the topology criterion enforces that each restricted facet has its three vertices on the same PSC patch, and that each restricted edge has its two vertices on the same PSC crease. The mesh can thus be refined beyond the specified approximation criterion until all surface sheets are separated, as illustrated by Figure 5. In our experience, computational times to obtain a mesh range from seconds for the sphere and nested spheres models to hours for the Michelangelo David models through minutes for the anchor, turbine and bimba models (resp. 10, 15 and 23).

4 Conclusion

The algorithm presented in this paper introduces a new mesh generation framework, based on the idea of interleaving refinement and optimization. Guided by user-defined criteria such as size, shape, and approximation error of mesh elements, refinement steps are parsimoniously applied batch-wise through the insertion of independent sets. Optimization steps are performed through a variant of

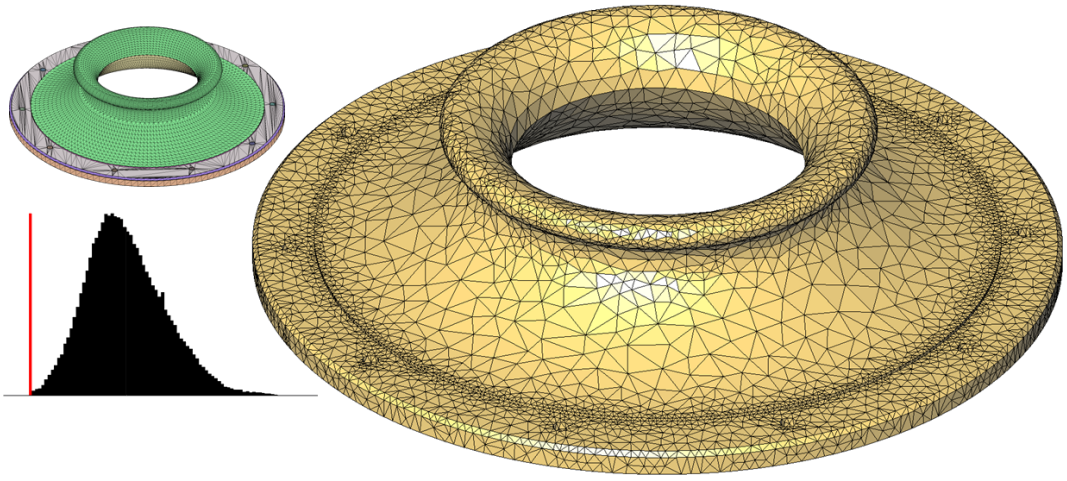


Figure 10: Turbine. Mesh generated by interleaved refinement and optimization with $l_{max} = 0.1$, $\epsilon_{max} = 0.001$. The inset shows the input PSC with all patches segmented. The mesh has 14K vertices and 51K tetrahedra, with all dihedral angles greater than 15 degrees.

Chen's ODT that handles boundary as well as spatially-varying mesh sizing. The main limitation of our algorithm is that it does *not* handle sharp input creases subtending small angles (the theoretical bound on input angles is 90°, see [RY07]).

As future work, we plan on dealing with small angles through the use of weights of a regular triangulation [CDL07]. Finally, and as the general framework of this algorithm is generic enough, we can imagine extending it to other representations of the input, as for example implicit surfaces or piecewise smooth parametric surfaces represented as NURBS patches. The latter would require efficient intersection computations using, e.g., the SINTEF Spline Library.

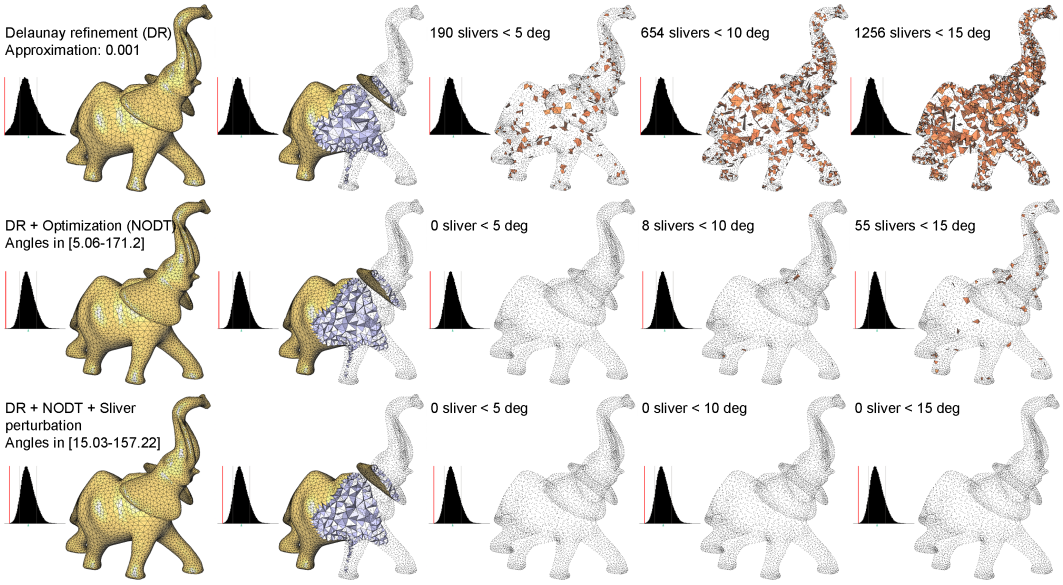


Figure 11: Elephant. Top: mesh generated by Delaunay refinement with $l_{max} = 0.1$, $\epsilon_{max} = 0.001$. Distribution of dihedral angles, and sliver tetrahedra with dihedral angles lower than 5, 10 and 15 degrees are shown. Middle: output of Delaunay refinement (i.e., top row) optimized with our technique. Bottom: optimized mesh (middle row) after sliver perturbation.

References

[ABE99] N. Amenta, M. Bern, and D. Eppstein. Optimal point placement for mesh smoothing. *Journal of Algorithms*, 30(2):302–322, 1999.

[ACSYD05] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. In *Proc. of ACM SIGGRAPH 2005*, volume 24(3), pages 617–625, 2005.

[ADA07] L. Antani, C. Delage, and P. Alliez. Mesh sizing with additively weighted voronoi diagrams. In *Proc. of the 16th Int. Meshing Roundtable*, pages 335–346, 2007.

[BOG02] C. Boivin and C. Ollivier-Gooch. Guaranteed-quality triangular mesh generation for domains with curved boundaries. *Int. J. Numer. Methods Eng*, 55:1185–1213, 2002.

[BWY07] J.D. Boissonnat, C. Wormser, and M. Yvinec. Curved voronoi diagrams. In Springer, editor, *Effective Comp. Geometry for Curves and Surfaces*, pages 67–116. 2007.

[CDE+00] S.W. Cheng, T.K. Dey, H. Edelsbrunner, M.A. Facello, and S.H. Teng. Sliver exudation. *Journal of the ACM*, 47(5):883–904, 2000.

[CDL07] S.W. Cheng, T.K. Dey, and J. Levine. A practical delaunay meshing algorithm for a large class of domains. In *Proc. of the 16th Int. Meshing Roundtable*, pages 477–494, 2007.

[CDR07] S.W. Cheng, T.K. Dey, and E.A. Ramos. Delaunay refinement for piecewise smooth complexes. In *Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1096–1105, 2007.

[Che89] L.P. Chew. Guaranteed-quality triangular meshes. Technical Report 89-983, Department of Computer Science, Cornell University, 1989.

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

[Che04] L. Chen. Mesh smoothing schemes based on optimal delaunay triangulations. In *13th International Meshing Roundtable*, pages 109–120, 2004.

[CX04] L. Chen and J. Xu. Optimal delaunay triangulations. *Journal of Computational Mathematics*, 22(2):299–308, 2004.

[dB05] G. Van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *Graphics Tools: The Jgt Editors' Choice*, 2005.

[DFG99] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tesselations. *SIAM Review*, 41(4):637–676, 1999.

[FOG97] L.A. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *In. Jour. for Num. Methods in Eng.*, 40(21):3979–4002, 1997.

[KS07] B. Klingner and J.R. Shewchuk. Aggressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, pages 3–23. 2007.

[Li00] Xiangyang Li. *Sliver-free Three Dimensional Delaunay Mesh Generation*. PhD thesis, Computer Science, University of Illinois at Urbana-Champaign, 2000.

[LS07] F. Labelle and J.R. Shewchuk. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. In *International Conference on Computer Graphics and Interactive Techniques*. ACM Press New York, NY, USA, 2007.

[NCC02] D. Nave, N. Chrisochoides, and L.P. Chew. Guaranteed quality parallel delaunay refinement for restricted polyhedral domains. *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 135–144, 2002.

[ORY05] S. Oudot, L. Rineau, and M. Yvinec. Meshing volumes bounded by smooth surfaces. In *Proc. of the 14th Int. Meshing Roundtable*, pages 203–219, 2005.

[Rup95] J. Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.

[RY07] L. Rineau and M. Yvinec. Meshing 3d domains bounded by piecewise smooth surfaces. In *Proc. of the 16th Int. Meshing Roundtable*, pages 443–460, 2007.

[She98] J.R. Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *Proc. of the Fourteenth Annual Symp. on Comp. Geometry*, pages 86–95, 1998.

[She02a] J.R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1-3):21–74, 2002.

[She02b] J.R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *Proc. of the 11th Int. Meshing Roundtable*, pages 115–126, 2002.

[Si] Hang Si. TETGEN, a quality tetrahedral mesh generator and three-dimensional delaunay triangulator. http://tetgen.berlios.de/.

[Ter05] Pierre Terdiman. OPCODE 3D collision detection library, 2005. http://www.codercorner.com/Opcode.htm.

[VLV+04] A.W. Vieira, T. Lewiner, L. Velho, H. Lopes, and G. Tavares. Stellar mesh simplification using probabilistic optimization. *Computer Graphics Forum*, 23(4):825–838, 2004.

[WK02] J. Wu and L. Kobbelt. Fast mesh decimation by multiple-choice techniques. *Vision, Modeling, and Visualization 2002*, pages 241–249, 2002.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399