Interleaving Delaunay Refinement and Optimization for Practical Isotropic Tetrahedron Mesh Generation

Jane Tournois
INRIA

Camille Wormser
ETH Zurich

Pierre Alliez
INRIA

Mathieu Desbrun
Caltech

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 conservative in number of Steiner point 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.

Keywords: Mesh generation, Delaunay refinement, optimization.

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 [Shewchuk 2002b], 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 isotropic tetrahedral meshes. Delaunay refinements and variational optimizations are interleaved in order to produce a discretization of the domain that meets a series of desirable 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 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 have been at the core of some of the fastest meshing techniques, as they provide a blazingly fast approach to meshing most of the domain. While smooth surface boundaries can be efficiently handled with guaranteed minimum dihedral angles [Labelle and Shewchuk 2007], the regularity of the mesh resulting from these methods (*i.e.*, the presence of preferred edge directions) can induce severe aliasing effects in simulation [Wendt et al. 2007].

Techniques combining Delaunay triangulation and refinement have received special attention due to their versatility and theoretical foun-

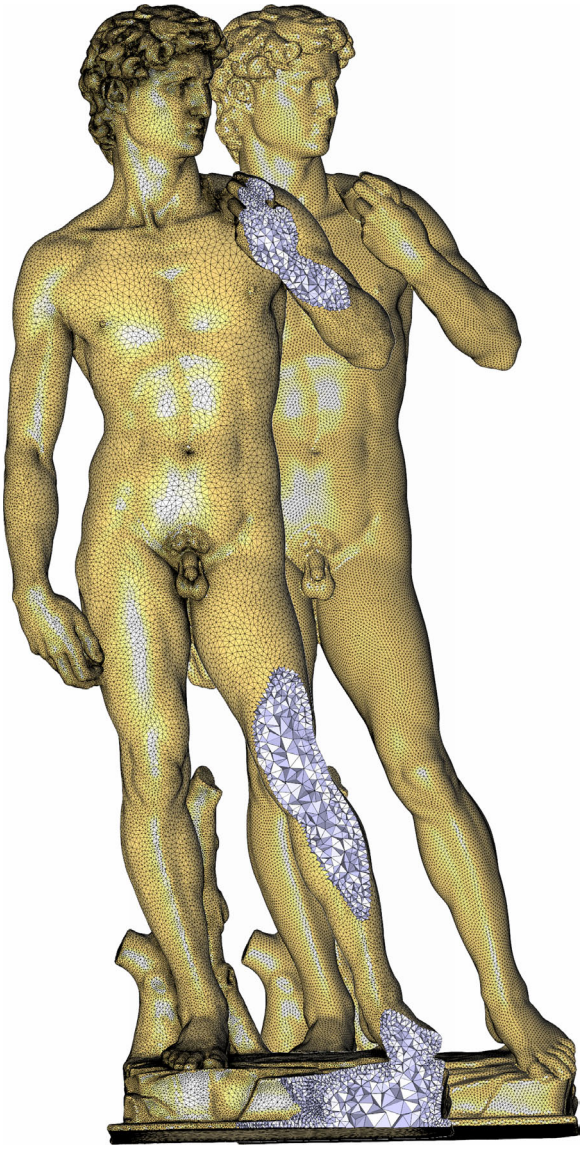


Figure 1: *Michelangelo's David. Our mesh generation algorithm produces high quality meshes through Delaunay refinements interleaved with optimization. The input PSC has 800K triangles; (right) a uniform sizing criterion generates a* 1*M vertices mesh; (left) approximation error and shape criteria alone generate a smaller graded mesh (*250*K vertices), while guaranteeing the same mesh quality and a better local approximation error.*

dations. They have been used initially in 2D [Chew 1989], then in 3D for polyhedral domains [Nave et al. 2002; Acar et al. 2007], for smooth surfaces [Chew 1993], for 3D domains bounded by smooth surfaces [Oudot et al. 2005; Boivin and Ollivier-Gooch 2002] and for 3D domains bounded by piecewise smooth surfaces [Rineau and Yvinec 2007; Cheng et al. 2007a; Cheng et al. 2007b]. They proceed by refining and filtering a 3D triangulation until a set of user-specified criteria is satisfied. Refinement 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 applied to cull simplices such that the triangulation restricted to the input domain tessellates the domain, and such that the boundary of this restricted triangulation approximates the domain boundary. This procedure becomes more delicate for piecewise-smooth inputs (a more general class of domains 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 another level of difficulty for Delaunay refinement. Refinement techniques are usually judged on the quality of the resulting mesh elements and on the sparsity of Steiner point insertion.

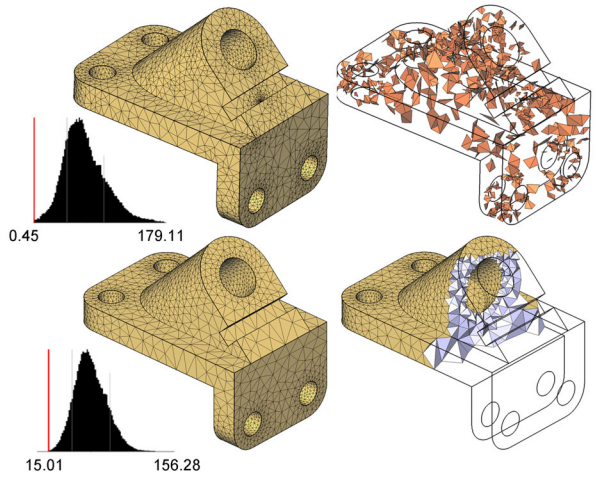


Figure 2: *Top: Mesh (5,499 vertices) generated by Delaunay refinement (shape and boundary approximation criteria activated). Notice the cluster in the middle of the armhole. Right image shows tetrahedra with dihedral angles smaller than 15 degrees. Bottom: Mesh (3,701 vertices) generated by interleaving Delaunay refinement and optimization so as to satisfy the same criteria. Distributions of dihedral angles are shown on the left.*

The quest for ever better quality meshes has sparked advances in mesh optimization through local vertex relocation to optimize a specific notion of mesh quality [Amenta et al. 1999; Persson and Strang 2004], topological operations [Cheng et al. 2000], or both [Freitag and Ollivier-Gooch 1997]. Further improvement of the mesh quality can be achieved by inserting additional vertices and/or incorporating a rollback mechanism to undo previous optimizations in order to guarantee a monotonic increase in mesh quality [Klingner and Shewchuk 2007]. 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 [Chen and Xu 2004; Chen 2004] 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 [Alliez et al. 2005], 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 were 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 [Rineau and Yvinec 2007]. Also, an estimate of the boundary local feature size (lfs) is required to derive a sizing function; however, there is currently no consensus on how to extend the notion of lfs to polyhedral domains. Finally, this method cannot handle arbitrary boundary meshes, requiring a *restricted* Delaunay triangulation instead.

1.2 Contributions

This paper combines the efficacy of Delaunay refinement methods with the isotropic quality induced by optimal Delaunay optimization techniques (extending the 2D approach of [Tournois et al. 2007]) 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 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 Fig. 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 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 select subsets of isolated Steiner points using the probabilistic multiple choice approach [Wu and Kobbelt 2002] to reduce the treatment 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, the method is versatile enough to serve as a general framework for isotropic tetrahedral meshing, as each step involved in the process can be adapted to special requirements.

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 shapes and sizes of elements through refinements, while passes of optimization further improve the shape of the elements. The high-level pseudo-code is as follows:

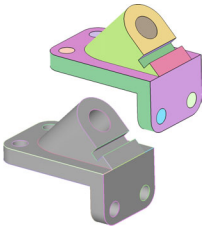
Algorithm 1 Mesh generation at a glance

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 Criteria $\{k_1, k_2, ..., k_n\}$ not all met **do**
  Refine through sparse vertex insertions (Section 2.4)
  Optimize mesh (Section 2.5)
end while
Perturb remaining slivers (Section 2.6)

2.1 Input

The input is a 3D domain Ω whose boundary $\partial\Omega$ is defined as a piecewise smooth complex (PSC). More specifically, our cur-

rent implementation takes as input a piecewise linear approximation of a PSC. This approximation 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 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*. 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. The sharp input creases subtending angles should not be too small (the theoretical bound is 90 degrees) to have a guarantee that Delaunay refinement steps will terminate (see [Rineau and Yvinec 2007]).



2.2 Parameters

The user selects a set of criteria that the final mesh must satisfy. These criteria, which accommodate the typical user requirements for mesh generation, are used to guide the refinement process as explained in Section 2.4. All of them but the first criterion are optional in our implementation:

- *Sizing:* a spatially-varying sizing function (or possibly a single value if constant) indicates the maximum mesh edge length allowed within the domain.
- *Approximation:* an approximation control function defines a local upper bound $\varepsilon_{\max}$ for the surface or crease approximation error. Similar to the mesh sizing function, it is defined either 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 if 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.

2.3 Initialization

A first mesh $\mathcal{M}$ of the domain is obtained 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 faces 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 [Rineau and Yvinec 2007]. 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 those defined by the user (typically, we relax the various input criteria parameters by a factor two), using our refinement procedure that we detail next.

2.4 Refinement

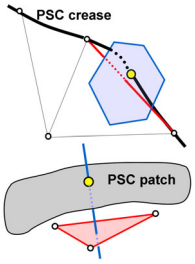
The refinement process is entirely driven by the user-defined criteria listed in Section 2.2. Each refinement step is designed to remove a set of *bad elements* (simplices not satisfying at least one of the given criteria) by inserting so-called *Steiner vertices* to $\mathcal{M}$. Unlike typical Delaunay refinement techniques that 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 (see Fig. 3).

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 $\partial\Omega$—namely, tetrahedra whose dual Voronoi vertex is located inside Ω, facets whose dual Voronoi edge intersects $\partial\Omega$ and edges whose dual Voronoi facet intersects an input crease. 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 a tet 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 $\partial\Omega$ 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*, a *feature* (e.g. corner), a *crease*, or a *boundary* (i.e. surface) 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* [Rineau and Yvinec 2007] 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.

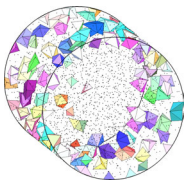
Steiner Vertices For each bad simplex, we define its associated Steiner point location. The associated Steiner point to a restricted *edge* is the farthest intersection point between its dual Voronoi face and the input creases. The associated Steiner point to a restricted *facet* is the farthest intersection point between its dual Voronoi edge and $\partial\Omega$. 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 enforce the topological disk condition.



To ensure termination of the refinement process, we further check

for *encroachment* [Shewchuk 2002a; Cheng et al. 2007a; Rineau and Yvinec 2007]. The candidate Steiner point p of a tetrahedron 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* on 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).

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 sets of conflict regions*. For each Steiner point p, we call the "conflict region" the tetrahedra that would be affected by its insertion as well as their adjacent tetrahedra: 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 regions, 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 number of bad tetrahedra 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. The inset shows an independent set on the mesh of a cylinder for which only the approximation criterion is not yet satisfied.



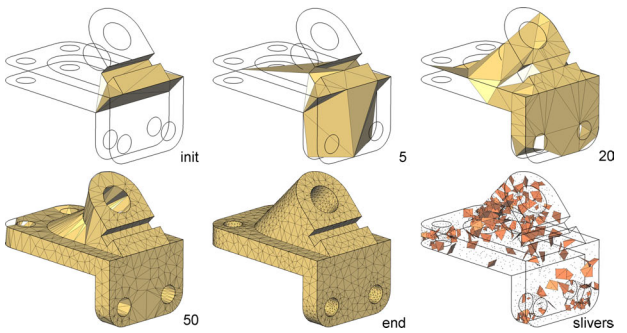


Figure 3: *Refinement steps without optimization. The mesh initialized with feature vertices; after a few batch refinement steps (from 5 to 50); the final refined mesh with shape and approximation criteria satisfied; and its 244 slivers (tetrahedra with dihedral angle smaller than 10 degrees).*

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 point 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 [Wu and Kobbelt 2002]) as follows. At each step, a small container

Algorithm 2 Construction of Independent Set of Conflict Regions

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

Set Independent Set *IS* to *nil*.
Collect all restricted tets in T_{bad}.
Collect all bad crease edges in E_{bad}, bad boundary facets in F_{bad}.
for each bad simplex s in E_{bad} and F_{bad} (from worst to best), **do**
  Let p be the Steiner point of s.
  Let U_c be the set of all tets in direct conflict with p's insertion.
  Let U_n be the set of all tets sharing a facet with a tet in U_c.
  if No tetrahedron of $U_c \cup U_n$ is in the Independent Set *IS*, **then**
    Insert conflict region $U_c \cup U_n$ in *IS* along with p.
  end if
end for
Let C_{mc} be a multiple-choice container of N_{mc} tets.
while There are non-conflicted tets **do**
  Fill up C_{mc} with random tets from T_{bad} which Steiner points' conflict regions $U_c \cup U_n$ do not intersect regions already in *IS*.
  Add Steiner point of C_{mc}'s worst tet & its conflict region to *IS*.
end while
Batch-insert all Steiner points stored in *IS* to mesh.
Update restricted Delaunay triangulation.

of N_{mc} "bad" tetrahedra ($N_{mc} = 20$ in our implementation) is filled with randomly selected non-conflicted tetrahedra. The worst tetrahedron in this container is then selected, and the container is updated with another random non-conflicted tetrahedron. As our goal is to only sparingly refine the mesh before further optimization, this multiple-choice approach significantly speeds up our refinement process while preserving its overall performance.

2.5 Optimization

Chen [Chen 2004] 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 each tetrahedron T_j. 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 the triangulation onto the paraboloid [Boissonnat et al. 2007]. Because of a result of function approximation theory [Shewchuk 2002b] 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.

The energy E_{ODT} can be reformulated [Alliez et al. 2005] as

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

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 derivation of this quadratic energy in $\mathbf{x}_i$ leads to the following *optimal position* $\mathbf{x}_i^\star$ of interior vertex $\mathbf{x}_i$ in its 1-ring [Chen 2004]:

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

The term $\nabla_{\mathbf{x}_i} |T_j|$ is the gradient of the volume of the tetrahedron 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 $\sum_{T_j \in \Omega_i} \nabla_{\mathbf{x}_i} |T_j| = 0$, thus it 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* T's circumcenter, denoted $\mathbf{c}_T$. Using Eq. (3) in this special case of a 1-ring in the shape of a tetrahedron $T = (\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_r, \mathbf{x}_s)$, and taking the point $\mathbf{x}_i$ to be located at $\mathbf{x}_p$, we get:

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

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

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

where $\mathbf{N}_{p,q,r}$ is the area-weighted normal of the face (p,q,r) pointing towards the inside of the tetrahedron, *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 [Chen 2004; Alliez et al. 2005] 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 $\mathbf{x}_i$ a vertex on the *boundary* of a 3D mesh (i.e., it does not have a full 1-ring $\mathcal{N}(\mathbf{x}_i)$ of restricted tetrahedra). For a given connectivity, the new position $\mathbf{x}_i^\star$ of $\mathbf{x}_i$ that extremizes the ODT energy is a bit more complicated, as some of the face terms F do not disappear:

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

where the boundary terms B are

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

The first part ($|T|\mathbf{c}_T$) is the weighted barycenter of the circumcenters divided by the total 1-ring volume just as before. At the boundary appears an extra term, a sum over boundary triangles (i,p,q) 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 assign a multiplicative weight λ to the supplementary term B without affecting the update rule for internal vertices, because the boundary terms cancel each other out for a full 1-ring:

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

We now use set this extra degree of freemdom λ so as to retain the *ODT circumsphere property* mentioned earlier, but now in the case of an incomplete 1-ring: if all neighbors of $\mathbf{x}_i$ are at the same distance from $\mathbf{x}_i$, we want $\mathbf{x}_i^\star = \mathbf{x}_i$: 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

$$\mathbf{x}_i^\star = \mathbf{x}_i - \frac{1}{2|\Omega_i|} \sum_{T_j \in \Omega_i} \left(\nabla_{\mathbf{x}_i} |T_j| \left[\sum_{\mathbf{x}_k \in T_j} ||\mathbf{x}_i - \mathbf{x}_k||^2 \right] + (1-\lambda) \sum_{p,q \neq i} \mathrm{F}(\mathbf{x}_i, \mathbf{x}_p, \mathbf{x}_q) \right). \tag{6}$$

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

$$\nabla_{\mathbf{x}_i} |T_j| = -\sum_{p,q \neq i} \frac{1}{3} \mathbf{N}_{i,p,q}.$$

On the one hand, $\nabla_{\mathbf{x}_i} |T_j|$ is weighted by $3R$ in (6). On the other hand, each $\frac{1}{3}\mathbf{N}_{i,p,q}$ is weighted by $2R$ in (6). Enforcing the circumsphere property on partial 1-rings at the boundary thus requires $(1-\lambda) = 3/2$, i.e., $\lambda = -1/2$. The optimal position for this variant (denoted NODT for *Natural ODT*) is now computed as

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

where the boundary terms B (if any) are

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

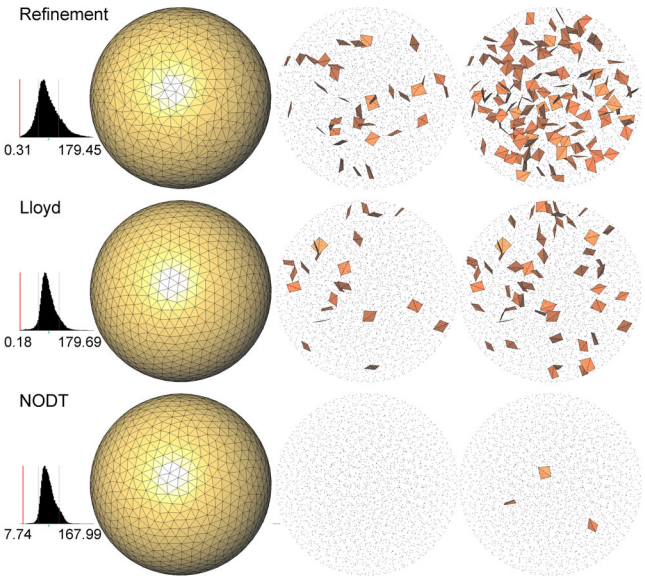


Figure 4: *Comparing Delaunay refinement and mesh optimization. Distributions of dihedral angles are shown to the left. Slivers are shown for a dihedral angle bound of respectively 5 (middle) and 10 degrees (right). Top: Delaunay Refinement alone (resp. 35 and 136 slivers). Middle: Optimized mesh with 100 Lloyd iterations (resp. 23 and 55 slivers). Bottom: Optimized mesh with 100 NODT iterations (resp. 0 and 3 slivers).*

Variable Sizing The optimization formula above is only valid for generating 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 the sizing function. Such measures are obtained by quadratures over the mesh elements. This sizing function [Antani et al. 2007] is guaranteed to be K-Lipschitz and is initialized with values computed 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 an update 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 to avoid the undoing of local refinements that a uniform sizing would produce.

Restriction and Projection In practice, as we want the mesh to interpolate the domain, each boundary vertex of the mesh should be on $\partial\Omega$. 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 [Alliez et al. 2005], we cannot guarantee a total absence of slivers (see Fig. 10). We thus perform a final phase of sliver removal. We implemented an explicit perturbation inspired by [Li 2000] which performed better than sliver exudation [Cheng et al. 2000] in all our experiments. This phase, described in [Tournois 2009], applies small perturbations to each vertex incident to slivers. We first try to move along the gradient of the squared circumradius of the sliver; if unsuccessful, the volume gradient is tried. If neither of these perturbations remove the sliver without adding new ones, random perturbations are applied repeatedly. Vertices located on sharp creases or the boundary are further reprojected onto their respective crease or boundary after perturbation. A relocation is validated if it both reduces the number of incident slivers and preserves the restricted triangulation locally. This process is iterated as long as there is a sliver in the mesh whose four vertices have not yet been perturbed, or until a maximum iteration count is reached. We also tried our perturbation directly after Delaunay refinement and Lloyd-based optimization; however, the presence of sliver chains (several slivers incident to each other) often resulted in significantly worse outputs.

2.7 Further Implementation Details

Our algorithm is implemented in C++ using the CGAL library. We use its 3D Delaunay triangulation as our core data structure. The input PSC is represented as a surface triangle mesh. 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 [Terdiman 2005]. 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

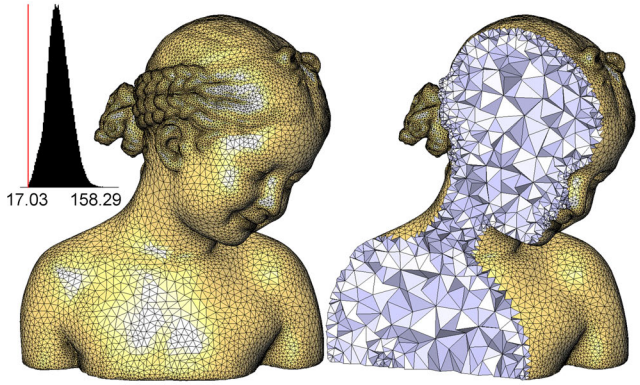


Figure 5: *Bimba. Mesh generated by interleaved refinement and optimization with $l_{max} = 0.1$, $\varepsilon_{max} = 0.0005$.*

(be it a test or an exhaustive enumeration) then calls intersection with AABBs during traversal, and intersection with PSC primitives (triangle or segments) at the leaves of the tree. 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. We also significantly 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 as excellent tetrahedra trades efficiency for the final mesh quality. Finally, each time the restricted Delaunay triangulation is updated, the circumcenters are cached to avoid recomputing them at each refinement and optimization step.

3 Results

To evaluate our approach, we tested the various steps of our algorithm separately, then together. Fig. 3 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 [Du et al. 1999], and our NODT in terms of number of slivers (*before* sliver removal for fairness) in Fig. 4. Fig. 10 shows an elephant mesh obtained by Delaunay refinement (top) as it gets optimized by our NODT routine (middle), then after sliver removal (bottom).

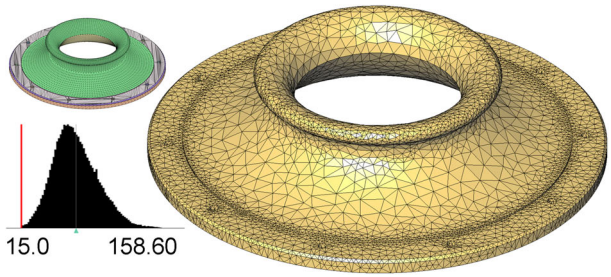


Figure 6: *Turbine. Mesh generated by interleaved refinement and optimization with $l_{max} = 0.1$, $\varepsilon_{max} = 0.001$. The inset shows the input PSC with all patches segmented. The mesh has 14K vertices and 51K tetrahedra, with all dihedral angles greater than 15 degrees.*

Fig. 5 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 dihedral angles are above 17 degrees. The input PSC has 400K vertices. We also tested our method on mechanical parts. Fig. 6 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 [Cheng et al. 2007a], TetGen [Si 2007] and GHS 3D [George 2004] in Fig. 7 in terms of both mesh quality and vertex count. Note that for TetGen and GHS 3D we provided as input the boundary of our optimized mesh for fair comparison. Fig. 9 shows the mesh of the Buddha model obtained by interleaved refinement and optimization, showcasing our method on a domain with a large range of local feature sizes.

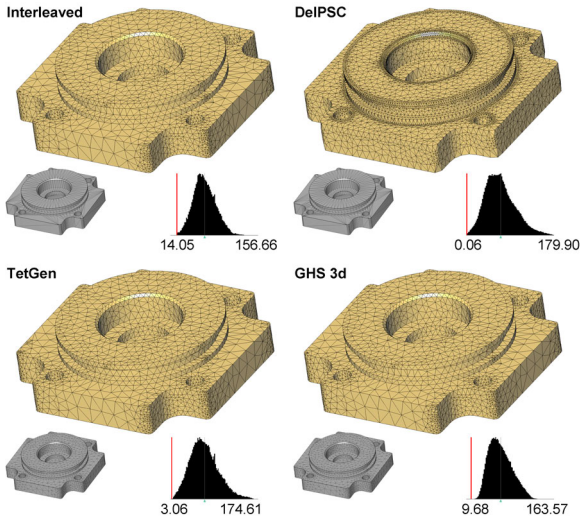


Figure 7: *Cover rear. Top left: mesh obtained by interleaved refinement and optimization with $l_{max} = 0.1$, $\varepsilon_{max} = 0.001$, $\sigma^f{}_{max} = 1.5$, $\sigma^t{}_{max} = 1.5$, with topology criterion activated; 4,050 vertices. Top right: mesh generated by DelPSC, same parameters; 15,157 vertices. Bottom: meshes generated by TetGen (left, 4,189 vertices) and GHS 3D (right, 6,641 vertices), same parameters and with the boundary of our optimized mesh taken as input.*

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 Fig. 8. In our experience (Table 1), computational times to obtain a mesh range from seconds for the sphere and nested-spheres models, to minutes for the anchor, turbine and bimba models (resp. 10, 15 and 23), to two hours for Michelangelo's David model.

Model	ε_{max}	Nb of tetrahedra	Time
Coverrear	0.001	22,091	3 min
Bimba(1)	0.001	52,866	10 min
Bimba(2)	0.0005	121,644	41 min
Buddha	0.001	161,378	20 min

Table 1: *Timings. The four meshes mentioned in this table were generated with $l_{max} = 0.1$, $\sigma^f{}_{max} = 2$, and $\sigma^t{}_{max} = 2$.*

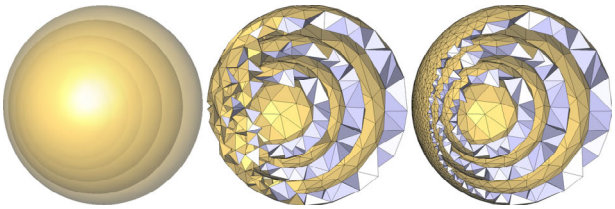


Figure 8: *Nested spheres. Left: input PSC. Middle: mesh generated by refinement with $l_{max} = 1$, $\varepsilon_{max} = 0.03$ and topology criterion not activated. Right: mesh further refined with same criteria but with topology activated.*

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 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 [Rineau and Yvinec 2007]). As future work, we plan on dealing with sharp creases subtending small angles through, e.g., the use of weights of a regular triangulation [Cheng et al. 2007a]. Finally, as the general framework of this algorithm is generic enough, we wish to extend it to other representations of the input, for example implicit surfaces or piecewise smooth parametric surfaces represented as NURBS patches.

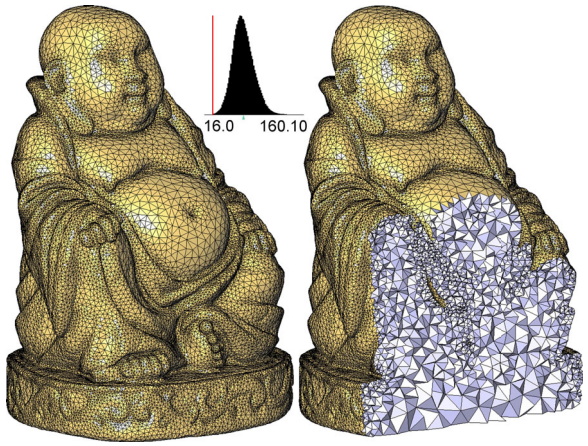


Figure 9: *Buddha. Obtained by interleaved refinement and optimization.*

Acknowledgments. The authors thank Mariette Yvinec for her constant support, Tamal Dey for his help with DelPSC, and Patrick Mullen for advice. Partial funding provided by the NSF (CCF-0811373, DMS-0453145, CMMI-0757106), the DOE (DE-FG02-04ER25657), and Pixar Animation Studios.

References

ACAR, U., HUDSON, B., MILLER, G., AND PHILLIPS, T. 2007. SVR: Practical Engineering of a Fast 3D Meshing Algorithm. In *Proc. of 16th Int. Meshing Roundtable*, 45–62.

ALLIEZ, P., COHEN-STEINER, D., YVINEC, M., AND DESBRUN, M. 2005. Variational Tetrahedral Meshing. In *Proc. of ACM SIGGRAPH 2005*, vol. 24(3), 617–625.

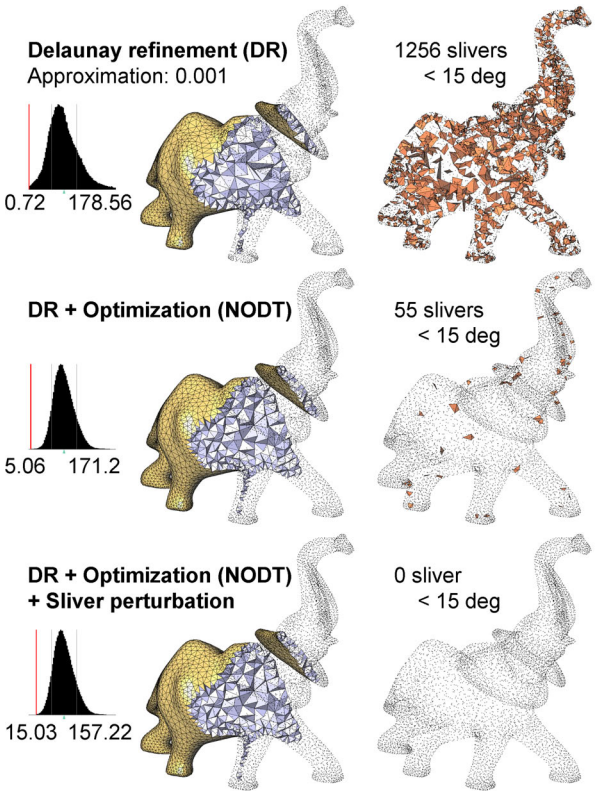


Figure 10: *Elephant. Top: mesh generated by Delaunay refinement with $l_{max} = 0.1$, $\varepsilon_{max} = 0.001$. Distribution of dihedral angles, and sliver tetrahedra with dihedral angles lower than 15 degrees are shown. Middle: output of Delaunay refinement (i.e., top row) optimized with our technique.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

AMENTA, N., BERN, M., AND EPPSTEIN, D. 1999. Optimal Point Placement for Mesh Smoothing. *J. of Algorithms 30*, 2, 302–322.

ANTANI, L., DELAGE, C., AND ALLIEZ, P. 2007. Mesh Sizing with Additively Weighted Voronoi Diagrams. In *Proc. of the 16th Int. Meshing Roundtable*, 335–346.

BOISSONNAT, J.-D., WORMSER, C., AND YVINEC, M. 2007. Curved Voronoi diagrams. In *Effective Comp. Geometry for Curves and Surfaces*, Springer, Ed. 67–116.

BOIVIN, C., AND OLLIVIER-GOOCH, C. 2002. Guaranteed-quality Triangular Mesh Generation for Domains with Curved Boundaries. *Int. J. Numer. Methods Eng. 55*, 1185–1213.

CHEN, L., AND XU, J. 2004. Optimal Delaunay triangulations. *J. of Comp. Mathematics 22*, 2, 299–308.

CHEN, L. 2004. Mesh Smoothing Schemes based on Optimal Delaunay Triangulations. In *Proc. of the 13th Int. Meshing Roundtable*, 109–120.

CHENG, S., DEY, T., EDELSBRUNNER, H., FACELLO, M., AND TENG, S. 2000. Sliver Exudation. *J. of the ACM 47*, 5, 883–904.

CHENG, S., DEY, T., AND LEVINE, J. 2007. A Practical Delaunay Meshing Algorithm for a Large Class of Domains. In *Proc. of the 16th Int. Meshing Roundtable*, 477–494.

CHENG, S., DEY, T., AND RAMOS, E. 2007. Delaunay Refinement for Piecewise Smooth Complexes. In *Proc. of the 18th Symp. on Discrete Algorithms*, 1096–1105.

CHEW, L. 1989. Guaranteed-Quality Triangular Meshes. Tech. rep., Dept. of Computer Science, Cornell Univ.

CHEW, L. 1993. Guaranteed-quality Mesh Generation for Curved Surfaces. In *Proc. of the 9th Symp. on Comp. Geometry*, ACM NY, USA, 274–280.

DU, Q., FABER, V., AND GUNZBURGER, M. 1999. Centroidal Voronoi tesselations. *SIAM Review 41*, 4, 637–676.

FREITAG, L., AND OLLIVIER-GOOCH, C. 1997. Tetrahedral Mesh Improvement using Swapping and Smoothing. *Int. J. for Num. Methods in Eng. 40*, 21, 3979–4002.

GEORGE, P.-L. 2004. Tetmesh-GHS3D, Tetrahedral Mesh Generator. *INRIA User's Manual, INRIA (Institut National de Recherche en Informatique et Automatique), France*.

KLINGNER, B., AND SHEWCHUK, J. 2007. Aggressive Tetrahedral Mesh Improvement. In *Proc. of the 16th Int. Meshing Roundtable*, 3–23.

LABELLE, F., AND SHEWCHUK, J. 2007. Isosurface Stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles. In *ACM Transactions on Graphics*, vol. 26(3), No. 57.

LI, X. 2000. *Sliver-free 3-Dimensional Delaunay Mesh Generation*. PhD thesis, University of Illinois at Urbana-Champaign.

NAVE, D., CHRISOCHOIDES, N., AND CHEW, L. 2002. Guaranteed Quality Parallel Delaunay Refinement for Restricted Polyhedral Domains. *Proc. of the 18th Symp. on Comp. Geometry*, 135–144.

OUDOT, S., RINEAU, L., AND YVINEC, M. 2005. Meshing Volumes Bounded by Smooth Surfaces. In *Proc. of the 14th Int. Meshing Roundtable*, 203–219.

PERSSON, P., AND STRANG, G. 2004. A Simple Mesh Generator in MATLAB. *SIAM Review*, 329–346.

RINEAU, L., AND YVINEC, M. 2007. Meshing 3D Domains Bounded by Piecewise Smooth Surfaces. In *Proc. of the 16th Int. Meshing Roundtable*, 443–460.

SHEWCHUK, J. 2002. Delaunay Refinement Algorithms for Triangular Mesh Generation. *Comp. Geometry: Theory and Applications 22*, 1-3, 21–74.

SHEWCHUK, J. 2002. What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. In *Proc. of the 11th Int. Meshing Roundtable*, 115–126.

SI, H., 2007. TETGEN, A Quality Tetrahedral Mesh Generator and 3-Dimensional Delaunay Triangulator. http://tetgen.berlios.de/.

TERDIMAN, P., 2005. OPCODE 3D Collision Detection library. http://www.codercorner.com/Opcode.htm.

TOURNOIS, J., ALLIEZ, P., AND DEVILLERS, O. 2007. Interleaving Delaunay Refinement and Optimization for 2D Triangle Mesh Generation. In *Proc. of the 16th Int. Meshing Roundtable*.

TOURNOIS, J. 2009. *Mesh Optimization*. PhD thesis, INRIA, Univ. de Nice Sophia Antipolis, France.

WENDT, J. D., BAXTER, W., OGUZ, I., AND LIN, M. C. 2007. Finite Volume Flow Simulations on Arbitrary Domains. *Graphical Models 69*, 1, 19 – 32.

WU, J., AND KOBBELT, L. 2002. Fast Mesh Decimation by Multiple-Choice Techniques. *Vision, Modeling, and Visualization 2002*, 241–249.