Variational Tetrahedral Meshing

Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, Mathieu Desbrun

▶ **To cite this version:**

Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, Mathieu Desbrun. Variational Tetrahedral Meshing. ACM Transactions on Graphics, 2005, 10.1145/1186822.1073238 . inria-00226418

HAL Id: inria-00226418
https://inria.hal.science/inria-00226418

Submitted on 30 Jan 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

published in ACM Transactions on Graphics, 24:617-625, 2005. SIGGRAPH'2005 Conference Proceedings

Variational Tetrahedral Meshing

Pierre Alliez
INRIA

David Cohen-Steiner
INRIA

Mariette Yvinec
INRIA

Mathieu Desbrun
Caltech

Abstract

In this paper, a novel Delaunay-based variational approach to isotropic tetrahedral meshing is presented. To achieve both robustness and efficiency, we minimize a simple mesh-dependent energy through global updates of both vertex positions *and* connectivity. As this energy is known to be the $\mathcal{L}^1$ distance between an isotropic quadratic function and its linear interpolation on the mesh, our minimization procedure generates well-shaped tetrahedra. Mesh design is controlled through a gradation smoothness parameter and selection of the desired number of vertices. We provide the foundations of our approach by explaining both the underlying variational principle and its geometric interpretation. We demonstrate the quality of the resulting meshes through a series of examples.

Keywords: Isotropic meshing, Delaunay mesh, sizing field, slivers.

1 Introduction

Three-dimensional simplicial mesh generation aims at tiling a bounded 3D domain with tetrahedra so that any two of them are either disjoint or sharing a lower dimensional face. Such a discretization of space is required for most physically-based simulation techniques: realistic simulation of deformable objects in computer graphics, as well as more general numerical solvers for partial differential equations in computational science, need a discrete domain to apply finite-element or finite-volume methods. Most applications have specific requirements on the size and shape of simplices in the mesh. *Isotropic meshing* is desirable in the common case where nearly-regular tetrahedra (nearly-equal edge lengths) are preferred.

Creating high quality tetrahedral meshes is a difficult task for a variety of reasons. First, the mere size of the resulting meshes requires robust, disciplined data structures and algorithms. There are also basic mathematical difficulties which make tetrahedral meshing significantly harder than its 2D counterpart: the most isotropic 3D simplex, the regular tetrahedron, does *not* tile 3D space (let alone specific domains), while the equilateral triangle does tile the plane; unlike the 2D case, even well-spaced vertices can create degenerate 3D elements such as slivers (see Fig. 2). Dealing with boundaries is also fundamentally more difficult in 3D: while it always exists a 2D triangulation conforming to any set of non intersecting constraints, this is no longer true in 3D [Shewchuk 1998a]. All these facts render both the development of algorithms and suitable error analysis for the optimal 3D meshing problem very challenging. Given that one can often observe in applications that the worst element in the domain dictates accuracy and/or efficiency [Shewchuk 2002a], it is clear that great care is required to design the underlying meshes and ensure that they meet the desired quality standards.

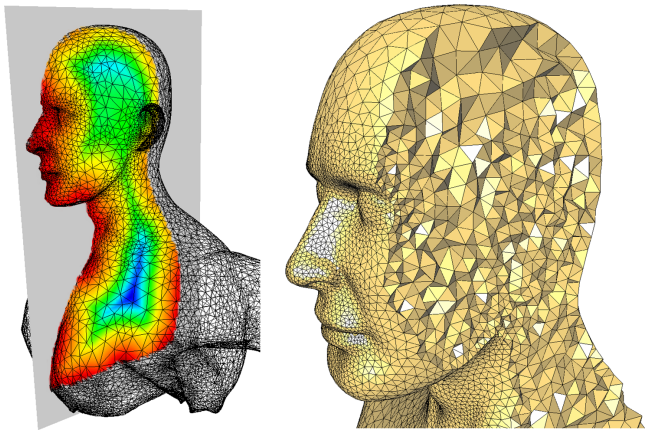


Figure 1: *Variational Tetrahedral Meshing: Given the boundary of a domain (here, a human torso), we automatically compute the local feature size of this boundary as well as an interior sizing field (left, cross-section), before constructing a mesh with a prescribed number of vertices (here $65K$) and a smooth gradation conforming to the sizing field (right, cutaway view). The resulting tetrahedra are all well-shaped (i.e., nearly regular).*

1.1 Previous Work & Nomenclature

The meshing community has extensively studied a number of techniques over the last 20 years. We do not aim at covering all previous work since comprehensive surveys are available [Carey 1997; Owen 1998; Frey and George 2000; Teng et al. 2000; Eppstein 2001]. To motivate our work we briefly review both the usual nomenclature and the main difficulties involved in isotropic tetrahedral mesh generation. Throughout *tet* will be the abbreviation for tetrahedron.

Proper mesh generation requires a number of successive stages, which are governed by a number of key factors:

- **Shape Quality Measures**: Element shape/size requirements are typically application-dependent. Consequently, an extraordinarily large number of quality measures has been proposed, ranging from minimum or maximum bounds on dihedral or solid angles, to more complex geometric ratios. We recommend [Shewchuk 2002a] for a clear exposition of both the history behind these measures and their relation to (1) the conditioning of finite element stiffness matrices and (2) the accuracy of linear interpolation of functions and their gradients. Among the most popular quality measures of a tet are the radius and radius-edge ratios. The latter measures the ratio between the circumsphere radius and the shortest edge length. It is not a *fair* measure since it does not approach zero for a class of degenerate tets called *slivers* (slivers result when four tet vertices are close to a great circle of a sphere and spaced roughly equally along this circle, see Fig. 2). The radius ratio, which takes the quotient of inscribed and circumscribed sphere radii (times three for normalization purposes), *is* a good measure for any kind of degeneracy.
- **Sizing requirement**: Accuracy and efficiency of numerical solvers depend on the local size of tets. Consequently, a *sizing field*, prescribing the ideal local edge length as a function of space, must be added. Obvious choices include the constant field for a uniform mesh, and a priori or a posteriori error estimators for sim-

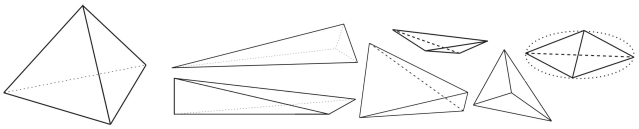


Figure 2: *Tet shapes: the regular tet (leftmost) is well shaped, unlike the other tets displayed: each represents a type of degeneracy. The rightmost one with 4 near-cocircular vertices is usually referred to as a* sliver.

ulations. To avoid bad dihedral angles in the simplices one typically requires the sizing field to vary smoothly [Ruppert 1993].

- **Boundary Requirements**: Some approaches aim at *conforming* to (i.e., matching exactly) the domain boundary by adding Steiner points if necessary [Cohen-Steiner et al. 2002; Krysl and Ortiz 2001; Cheng and Poon 2003]. Others require of the mesh boundary to only *approximate* the domain boundary. The latter allows for higher tet quality since the boundary is not required to match the input surface. In particular the latter is important when the initial input is a low quality surface triangulation.
- **Strategy**: Existing meshing techniques can be roughly classified by the general strategy they employ:
  - *Advancing front*: Starting from the boundary of the domain, new vertices are added by a local heuristic to ensure that the generated tets have acceptable shapes and sizes and conform to the desired sizing field. Global optimization steps can also be performed sporadically to improve the mesh quality further. A number of variants exist, such as sphere or bubble packing [Li et al. 2000], which provide better tet shape and size control albeit adding a significant computational overhead.
  - *Octree-based methods*: An octree is first refined until each of its leaves is either strictly inside or strictly outside of a finely voxelized version of the domain. Proper connections of the interior leaves through, for instance, a red-green strategy [Molino et al. 2003] then ensure a good initial mesh of the domain, usually improved through optimization or physically-based relaxation in particular to better approximate the domain boundary. Other similar methods offer bounds of worst dihedral angles even without a relaxation stage [Mitchell and Vavasis 2000]. Unfortunately, octree-based meshes have preferred edge directions, which may be detrimental to subsequent use in simulation.
  - *Delaunay approaches*: For a given set of sample points in $3D$, its *Delaunay triangulation* has the canonical property of minimizing the maximum radius of the minimum containment sphere. This property is very useful in approximation theory: this radius provides an upper bound on the $\mathcal{L}^\infty$ difference between any function f and its *piecewise linear approximant*, assuming f has bounded second derivatives. Thus a Delaunay triangulation provides good control over the worst interpolation error inside a domain. Consequently a large body of work in numerical analysis provides error estimates for a variety of applications using these meshes. Because of these as well as many other optimality properties, mesh generation relying on Delaunay triangulation such as Delaunay refinement [Ruppert 1993; Shewchuk 1998b; Shewchuk 2002b; Cheng et al. 2004], unit mesh [Borouchaki et al. 1997a; Borouchaki et al. 1997b], or centroidal Voronoi tessellations [Du and Wang 2003] have flourished in the meshing and Computational Geometry communities. Delaunay refinement methods offer some theoretical guarantees on the resulting meshes: they provide bounds on the radius-edge ratio, and are shown to be asymptotically optimal with respect to the number of elements in the mesh. Delaunay refinement, however, can generate slivers; some attempts have been made to handle the sliver problem within Delaunay refinement [Cheng et al. 1999; Cheng and Dey 2002; Li and Teng 2001]. Unfortunately the theoretical guarantees are quite poor, and the mesh either is *no longer Delaunay* but a regular (weighted Delaunay) triangulation, or comes with degraded bounds on the radius-edge ratio.
  - *Mesh Optimization Techniques*: Even if fast and robust Delaunay triangulators are available, the previous strategies can require substantial implementation effort to make them robust to arbitrary input domains. A large number of practical meshing techniques instead employ local optimization methods which move vertices adjacent to poorly-shaped tets to improve mesh quality. Coupled with local face swapping between adjacent tets as well as tet insertions and deletions, these strategies can result in nice final meshes [Freitag and Ollivier-Gooch 1996; Cutler et al. 2004]. Unfortunately, these optimizations often use highly non-convex functionals and get easily stuck in local minima.

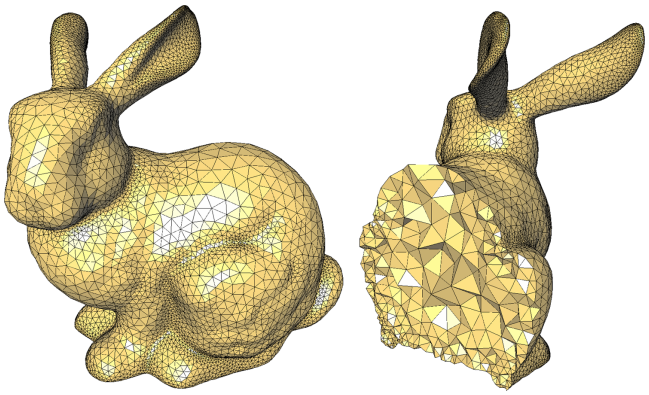


Figure 3: *Stanford bunny: meshing the interior of the bunny with adapted tets (smaller near the boundary, larger inside, and smooth gradation ($K = 1$) in between). The cutaway views show the well-shapedness of the mesh elements inside the domain; notice also the quality of the boundary mesh.*

From this brief overview we see that meshing has been approached with two very different emphases: theory and practice. Theoretical methods, most commonly using iterative Delaunay refinement approaches, come with quality guarantees that are often not suited to further use in practical applications: the presence of fairly degenerated tets are a serious problem for many numerical methods. Alternatively, optimization methods provide viable solutions with relatively little implementation effort, and the quality obtained is satisfactory for a class of applications. Alas, their ad-hoc nature does not warrant high-quality meshes. When seeking high quality meshes, a method combining optimization with solid theoretical foundations would provide the best of both worlds, promising meshes of a quality that none of the existing approaches could obtain by themselves.

1.2 Approach and Contributions

In this paper, we present a Delaunay-based optimization technique, that we call *Variational Tetrahedral Meshing*, to efficiently mesh a bounded 3D domain Ω of arbitrary topology or number of connected components. The domain boundary $\partial\Omega$ is assumed to be a manifold, watertight and intersection-free triangular mesh. Drawing on recent work on surface approximation [Cohen-Steiner et al. 2004] and Optimal Delaunay Triangulations [Chen and Xu 2004], we propose a simple minimization procedure that alternates global 3D Delaunay triangulation and local vertex relocation to *consistently and efficiently minimize a global energy* over the domain. It results in a robust meshing technique that generates high quality isotropic meshes in terms of radius ratios, as well as angles. A notable feature of the method is that it removes slivers inside the domain. To provide a flexible meshing tool, we also introduce an automatic sizing field construction that guarantees an arbitrary smooth gradation of the mesh together with faithful approximation of the domain boundary. Equipped with these tools, the user has full control over the mesh design, and can require a specific number of vertices for the final mesh. We demonstrate the versatility and robustness of our method through a series of results and comparisons; we also give details on the current limitations.

2 Variational Approach to Meshing

Variational approaches (that is, methods relying on energy minimization) have been advocated as a powerful and robust tool in meshing both in graphics for triangle [Hoppe et al. 1993; Cohen-Steiner et al. 2004] and tet [Molino et al. 2003; Cutler et al. 2004] meshes and in mechanical engineering for volumetric meshes [Freitag and Ollivier-Gooch 1996; Du and Wang 2003]. These methods basically define (often highly) non-convex energies that they minimize through vertex displacements and/or connectivity changes in the current mesh. Our method also falls into this broad category. However, in contrast to earlier work, we use a simple quadratic energy (which we analyze) and allow for global changes in mesh connectivity during energy minimization. We will point out both the theoretical and practical consequences of such a strategy. We begin by motivating our choice of energy.

2.1 Consistent Energy Minimization

A few of these variational methods [Cohen-Steiner et al. 2004; Du and Wang 2003] have an attractive theoretical property resulting in remarkable results: vertex positions *and* connectivity updates are performed alternately to *minimize the same quadratic energy*. This specificity has rich consequences. First, each update can be done *optimally* due to the simplicity of the energy used. Second, assuming convexity of the boundary, the energy decreases monotonically, implying eventual convergence. Lastly, since both optimization steps minimize the same energy, their final meshes have a concrete, variational nature: the resulting meshes are (quasi) minimizers of a "quality" functional.

Centroidal Voronoi Tessellations Du and Wang [2003] propose to generate meshes that are *dual* to optimal Voronoi diagrams. These diagrams are achieved by minimizing the quadratic energy:

$$E_{\text{CVT}} = \sum_{i=1..N} \int_{V_i} ||\mathbf{x} - \mathbf{x}_i||^2 d\mathbf{x} \qquad (1)$$

where the $\mathbf{x}_i$ are vertex positions and V_i a local cell associated with each $\mathbf{x}_i$; the union of these cells forms a partition of the domain Ω. Du and Wang used Lloyd relaxation [Lloyd 1957] to robustly minimize this energy: for a given set of vertices, compute their Voronoi diagram (restricted to the domain Ω) since it *is* the energetically optimal partition for the current vertex positions. In a second phase, the partition is held fixed and vertex positions $\mathbf{x}_i$ are optimized. Even though these steps of *partitioning* and *vertex position* optimization are quite different in character, each of them decreases the same energy. Du and Wang explain how a mesh that minimizes this energy has each vertex *at the centroid* of its own Voronoi cell: hence the name *Centroidal Voronoi Tessellation* (CVT). Aside from the theoretical properties of CVTs, Du and Wang also note the superior results they get in comparison to conventional Laplacian smoothing (a widespread technique in graphics due to its simplicity, but for which the associated energy only relies on edge length, not on spatial distribution).

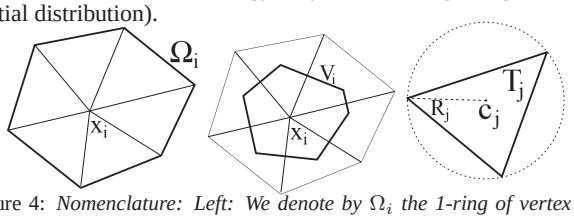


Figure 4: *Nomenclature: Left: We denote by Ω_i the 1-ring of vertex $\mathbf{x}_i$. Middle: V_i is the Voronoi cell of vertex $\mathbf{x}_i$. Right: The center of the circumcircle of triangle T_j, is denoted $\mathbf{c}_j$, while its radius is denoted R_j.*

¿From the analysis of E_{CVT} it is well known that its minimization corresponds to minimizing the volume between a paraboloid $f(\mathbf{x}) = ||\mathbf{x}||^2$ and an *underlaid, circumscribing piecewise linear approximant* $f^{\text{dual}}_{\text{PWL}}$, which is formed by planar patches tangent to the paraboloid (see Fig. 5(a)):

$$E_{\text{CVT}} = ||f - f^{\text{dual}}_{\text{PWL}}||_{\mathcal{L}^1}.$$

In 2D, this approach will lead to isotropically sampled meshes since it has been shown that any $\mathcal{L}^p$ optimal approximation of a smooth function asymptotically tends to align and shape its elements according to the eigenvectors and eigenvalues of its Hessian [Shewchuk 2002b]: since the Hessian of the quadratic function $f(\mathbf{x}) = ||\mathbf{x}||^2$ is isotropic, the resulting meshes must have nearly hexagonal Voronoi cells, i.e., nearly equilateral triangles in the dual Delaunay mesh.

Unfortunately, and despite Du's proposal [2003] to use CVTs for tet meshing, there exists no proof of such a dual property in 3D. Our own tests show that using Du's suggestion for tet meshing gives rise to numerous degenerate *sliver* tets (see Fig. 2). We can attribute the slivers to the fact that E_{CVT} tends to optimizes the compactness of the dual Voronoi cells, but not the compactness of simplices in the primal Delaunay triangulation: therefore, the presence of a sliver is *not* penalized by this energy. In other words, this variational approach ensures that the vertices in the domain are well spaced (i.e., isotropic point sampling—see [Hardin and Saff 2004] for an overview of this interesting problem); sadly, having well-spaced vertices guarantees nothing in terms of the quality of the actual 3D mesh [Eppstein 2001].

Optimal Delaunay Triangulations Recently, Chen [2004] proposed an approach "dual" to the above in the context of mesh optimization. He used the following energy:

$$E_{\text{ODT}} = ||f - f^{\text{primal}}_{\text{PWL}}||_{\mathcal{L}^1},$$

i.e., the volume between a paraboloid and an *overlaid, circumscribing piecewise linear approximant* $f^{\text{primal}}_{\text{PWL}}$ formed by a linear interpolation of points on the paraboloid (see Fig. 5(b)). Chen made the observation that changing the energy from E_{CVT} to E_{ODT} amounts to only a slight change in Eq. (1), turning it into:

$$E_{\text{ODT}} = \frac{1}{n+1} \sum_{i=1..N} \int_{\Omega_i} ||\mathbf{x} - \mathbf{x}_i||^2 d\mathbf{x}. \qquad (2)$$

The integral is now taken over each *1-ring* region Ω_i (also called the star of the vertex $\mathbf{x}_i$, see Fig. 4). Notice that these regions overlap. These quadratic energies differ quite significantly: Chen's E_{ODT} energy measures a quality of the *simplicial mesh*, not of its dual. It is thus more prone to generate well-shaped primal elements, while E_{CVT} was maximizing the compactness of the dual Voronoi cells.

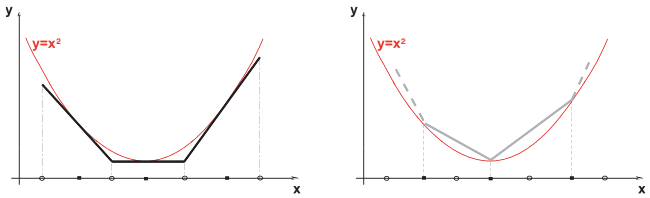


Figure 5: *PWL approximations: A paraboloid can be approximated by an underlaid circumscribed PWL function (left), or by an overlaid one (right).*

Although no formal guarantee on the resulting meshes is given in [Chen and Xu 2004], the 2D results presented are of high quality. The smoothing technique presented in [Chen 2004] updates the mesh connectivity through only-local edge flips when an inverted triangle is detected. Unfortunately, this local connectivity optimization in the 2D triangle case does not carry over to 3D: there is no theorem proving that an arbitrary mesh is only a few flips away from the optimal connectivity.

2.2 Our Variational Approach

We propose an algorithm to consistently minimize the primal energy E_{ODT}. This is achieved not just through a *smoothing* procedure (as suggested in [Chen 2004]), but through a full-blown minimization procedure for both vertex positions *and* connectivity.

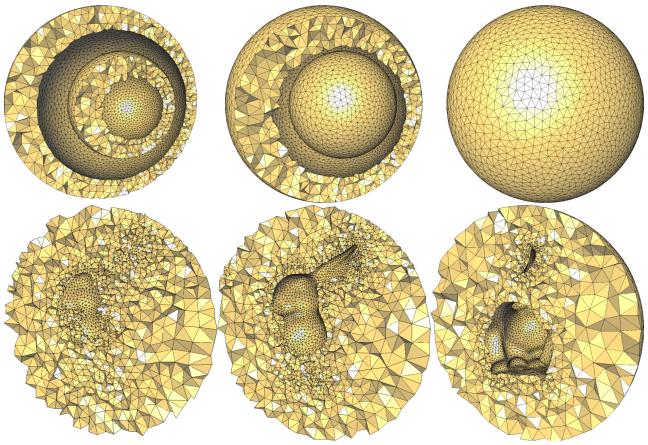


Figure 6: *Complex Topology: As stressed in this paper, our approach can as well mesh complex domains with arbitrary genus. Four nested spheres define a multi-layer object (top); a bunny immersed in a sphere (bottom).*

Optimizing Connectivity Connectivity optimization is easily achieved: for a given set of vertex positions $\mathbf{x}_i$, its Delaunay triangulation is again (remarkably!) *the* optimal connectivity which minimizes E_{ODT} (as shown in [Chen and Xu 2004]) just as it is optimal for E_{CVT}. Therefore, we compute the (global) Delaunay connectivity systematically, guaranteeing optimality of the connectivity at each iteration.

Optimizing Vertex Positions One can show that for a given mesh $\mathcal{M}$ with vertices $\mathbf{x}_i$'s, E_{ODT} can be written as:

$$E_{\text{ODT}} = \frac{1}{4}\sum_i {\mathbf{x}_i}^2 |\Omega_i| - \int_{\mathcal{M}} \mathbf{x}^2 d\mathbf{x}, \tag{3}$$

where $|\Omega_i|$ is the measure (volume in 3D) of the 1-ring neighborhood of vertex $\mathbf{x}_i$ (Appendix B gives a short proof). 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:

$$\mathbf{x}_i^\star = -\frac{1}{2\,|\Omega_i|}\sum_{T_j\in\Omega_i}\left(\nabla_{\mathbf{x}_i}|T_j|\left[\sum_{\substack{\mathbf{x}_k\,\in\,T_j\\ \mathbf{x}_k\,\neq\,\mathbf{x}_i}} ||\mathbf{x}_k||^2\right]\right). \tag{4}$$

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 function $f(\mathbf{x}) = ||\mathbf{x}||^2$ by the translated function $f(\mathbf{x}) = ||\mathbf{x}-\mathbf{x}_i||^2$, which has the same interpolation error and thus leads to the same optimal position, we get the following equivalent expression used to update a vertex position :

$$\mathbf{x}_i^\star = \mathbf{x}_i - \frac{1}{2\,|\Omega_i|}\sum_{T_j\in\Omega_i}\left(\nabla_{\mathbf{x}_i}|T_j|\left[\sum_{\mathbf{x}_k\in T_j} ||\mathbf{x}_i-\mathbf{x}_k||^2\right]\right). \tag{5}$$

Geometric and Physical Interpretations As shown in the Appendix C, we can express the latter optimality condition in more obvious geometric terms, to further our understanding of the benefits of this variational approach:

$$\mathbf{x}_i^\star = \frac{1}{|\Omega_i|}\sum_{T_j\in\Omega_i}|T_j|\mathbf{c}_j. \tag{6}$$

where $\mathbf{c}_j$ is the circumcenter of tet T_j (see Fig. 4). This last expression shows that, although we move each vertex to a local average, the optimal placement heavily depends on the local distribution. For instance, if all the 1-ring neighbors are on a common sphere, the optimal position will be the sphere center. In fact, as evidenced by Eq. 4, this optimal location depends *only* on the 1-ring neighbors, not on the current location. Note also the similarities with the generalized barycentric coordinates in the Voronoi polytope proposed in [Warren et al. 2004].

If we further transform the energy (see Appendix B), we get:

$$E_{\text{ODT}} = \sum_j \left(|T_j|R_j^2 - \int_{T_j}||\mathbf{x}-\mathbf{c}_j||^2 d\mathbf{x}\right) = 2\sum_j |M_{S_j} - M_{T_j}| \tag{7}$$

where M_{T_j} is the sum of the principal moments (i.e., the trace of the inertia tensor) of the tet T_j w.r.t. the circumcenter $\mathbf{c}_j$, while M_{S_j} is the same quantity for the *circumshell* S_j of equivalent mass (i.e., a shell in the shape of the circumsphere, with the same mass as tet T_j). Minimizing this energy amounts to *make the average moment of inertia of each tet match the one of its circumshell of equivalent mass*. Even Eq. (6) can be re-expressed in terms of these circumshells: a vertex is moved at the *barycenter* of its neighboring circumshells, which is reminiscent of the CVT property, this time on the primal mesh. We believe that these series of observations provide further insights on this simple quadratic energy, and how it relates to the well-shapedness of the resulting tets. It also provides a straightforward generalization to graded meshing as explained next.

2.3 Extension to Graded Meshes

Since the previous expressions only apply to uniform meshing, we extend the optimality condition next to allow for more flexible meshing capacities. For this purpose, we will make use of a *sizing field* μ as a roadmap to the desired tet sizes within the domain.

Generalized Optimality Conditions Eq. (6) gives us a straightforward means to extend the previous approach to create *graded* meshes. One can simply define a *mass density* in space, and use it in computing the inertia tensors. This density should agree with the sizing field, i.e., the locally-desired size of a tet. To simplify the computations, we use a one-point approximation of the sizing field μ in a tet and define the mass density as being $1/\mu^3$ since the local volume of a tet should be roughly the cube of the ideal edge size. Thus, we modify the optimality condition of a vertex as follows:

$$\mathbf{x}_i^\star = \frac{1}{\sum_{T_k\in\Omega_i}\frac{|T_k|}{\mu^3(\mathbf{g}_k)}}\sum_{T_j\in\Omega_i}\frac{|T_j|}{\mu^3(\mathbf{g}_j)}\mathbf{c}_j. \tag{8}$$

where $\mathbf{g}_k$ is the centroid of T_k. A formal integration of the sizing field within each tet would provide more precision. However it would also significantly affect the computational cost without a drastic change in the results thanks to our choice of sizing field (described next). Finally, we keep the Delaunay triangulation (of the new point positions) as the optimal connectivity.

Automatic Design of Sizing Field The sizing field can be virtually any function tuned to the specific application needs. We also want to provide a default sizing field construction for robustly generating a large spectrum of mesh types. Notice that the sizing field is *relative*; it describes the inhomogeneity of the desired edge length. The actual edge length will be proportional to this relative value, with the proportiondepending on the prescribed vertex budget. (Alternatively the user may want to use as many vertices as needed to produce a specific edge size.)

Because we aim at an isotropic approximation of the input domain boundary, the sizing field on the boundary should be a function of the local absolute maximum curvature. Since we also aim at approximating the domain topology, we need to make sure that the boundary approximation error will never exceed the local "thickness" of the domain: for instance, a dumbbell shape should have small tets in its bottleneck. Therefore we propose to build our sizing field on the notion of *local feature size* (*lfs*) introduced by Amenta and Bern [1998] and widely used in the field of shape reconstruction: it corresponds to the combination of curvature *and* thickness as we require. To define the local feature size, one first introduces

the medial axis $Sk(\Omega)$, of the domain (its intuitive skeleton) which is the locus of all the centers of maximal balls included in either Ω or its complement. Note that this skeleton has already been identified in the meshing community as playing a central role in sizing [Quadros et al. 2004]. Then the local feature size *lfs*$(\mathbf{x})$ at a point $\mathbf{x}$ of $\delta\Omega$ is defined as the distance $d(\mathbf{x}, Sk(\Omega))$ from $\mathbf{x}$ to the medial axis (where $d(.,.)$ is the Euclidean distance function).

Given the local feature size on the boundary, we need a canonical and controllable way to extrapolate this function to the interior. We face two conflicting constraints: a desire to minimize the number of total tets by forcing the inside of the object to have the largest tets possible, and the need to bound the mesh gradation (i.e., how fast tet sizes vary within a neighborhood) to maintain good shape quality [Ruppert 1993]. We propose to recast the problem of finding an ideal sizing field to finding the *maximal K-Lipschitz function* that does not exceed *lfs*$(\mathbf{x})$ on $\delta\Omega$. The parameter K will control the gradation (0 being the uniform case) of the resulting field. As we prove in the Appendix A, the function:

$$\mu(\mathbf{x}) = \inf_{\mathbf{s}\in\partial\Omega} [K\ d(\mathbf{s},\mathbf{x}) + \mathit{lfs}(\mathbf{s})] \tag{9}$$

satisfies these requirements. Consequently, we used it in all the examples shown in this paper. Now that the theoretical aspects of our approach have been addressed, we describe in the next section the details of a concrete implementation of these ideas.

3 Algorithm

In this section, we go through the details of each step of the following pseudo-code which summarizes our approach:

```
Read the input boundary mesh ∂Ω
Setup Data Structure & Preprocessing
Compute sizing field μ
Generate initial sites x_i inside Ω
Do
    Construct Delaunay triangulation({x_i})
    Move sites x_i to their optimal positions x_i*
Until (convergence or stopping criterion)
Extract interior mesh
```

For efficiency as well as robustness, we opted to use the Computational Geometry Algorithms Library (CGAL [Fabri et al. 2000]) for the input mesh data structure, as well as for the 3D Delaunay triangulation using robust arithmetics.

3.1 Input Domain Boundary

Our algorithm takes as input an intersection free closed surface triangle mesh defining the domain boundary $\partial\Omega$. We have no restriction on the topology of the domain Ω: it may contain multiple connected components, or have multiple voids, or both (see Fig. 6).

3.2 Setup & Preprocessing

The vertices of the input surface mesh $\partial\Omega$ are inserted in a 3D Delaunay triangulation, to create what we call the *control mesh*. This control mesh is used by our algorithm to estimate the local feature size of $\partial\Omega$ as well as to answer inside/outside queries. For efficient inside/outside queries, we require that the control mesh contains all triangle facets of the input boundary $\partial\Omega$, guaranteeing that it is the *restricted Delaunay triangulation* of the input vertices [Cohen-Steiner et al. 2002]. This allows us to tag the corresponding faces of the control mesh and in turn its tets with inside/outside tags. To achieve these goals, $\partial\Omega$ is originally either enriched or remeshed using an isotropic surface meshing algorithm [Boissonnat and Oudot 2005].

Discrete Skeleton Once the control mesh has been generated, we extract its *poles* by selecting a subset of Voronoi vertices (i.e., circumcenters of tets) in the following manner. For each Delaunay vertex v we first select as a pole the farthest circumcenter from all incident tets. The vector formed by v and this pole is considered as the local normal estimate. We deduce a local tangent plane estimate in v, and two half-spaces bounded by this plane. Next we search in the half-space that does not contain the pole the farthest Voronoi vertex incident to the site. If it exists it is added as a pole too. We refer the reader to [Amenta and Bern 1998] for a more detailed description of this simple procedure. By definition, and assuming a sufficiently dense set of points sampled on the boundary, the resulting set of all these poles is a discrete approximation of the *skeleton* (or medial axis) of the domain boundary $\partial\Omega$ (see Fig. 7).

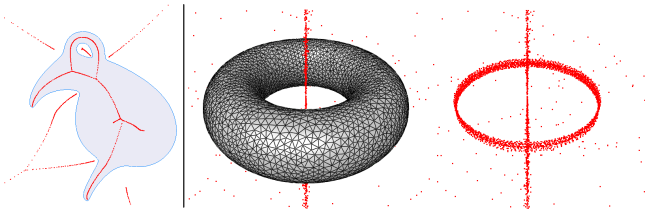


Figure 7: *Poles: The set of all poles (depicted as red dots) represents a discrete approximation of the* skeleton *of a 2D (left) or 3D (right) shape.*

Local Feature Size At each vertex of the boundary $\partial\Omega$, we approximate its local feature size *lfs* by measuring the distance to its closest pole (Fig. 8). To improve the efficiency of these queries (the set of poles is a dense point set for a complex boundary), we create a static kD-tree search data structure from the poles.

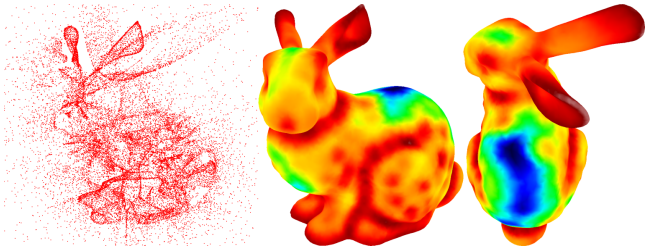


Figure 8: *Left: poles extracted from the Bunny model. Middle, right: the distance from each input vertex to the set of poles is the approximated* local feature size, *capturing both local thickness and curvature of the shape.*

Boundary Supersampling The input boundary is initially sampled with a large number of *quadrature samples* (used later on to find a good approximation of the surface). More precisely, three sets of quadrature samples are generated: on the boundary itself, on its sharp creases, and on its corners (for piecewise smooth domains—see, e.g., Fig. 16); this will allow us to both approximate the boundary and fit its features. Each quadrature sample stores its position $\mathbf{x}$ as well as a quadrature value, incorporating the area ds (for surface quadrature samples) or length dl (for feature quadrature samples) it covers. These values are set to $ds/\mu(x)^4$ and $dl/\mu(x)^3$ respectively, to conform to our mass density field [Du and Wang 2003]. Each corner sample is given an infinite density to guarantee that a vertex will be assigned there.

3.3 Fast Marching Construction of Sizing Field

Recall that a parameter K is used to adjust the sizing field according to the desired mesh gradation (see Section 2.3, and Figure 9 for illustration). We store the sizing field on a uniform grid bounding the domain. Each node of the grid must store the local sizing field value μ and an additional bit to specify whether this grid node lies inside or outside the domain as these grid nodes will be used to efficiently generate initial positions for the vertices of the mesh. However, computing μ in the interior of Ω would require the evaluation of a minimum over *each* vertex of $\partial\Omega$. To provide a faster grid initialization we use a fast marching method on the uniform grid using the 6-neighborhood incidence relationship beginning with the grid cells that intersect $\partial\Omega$. We define a *candidate cell* $\mathbf{x}$ as a cell for which we have stored a temporary *buddy cell*, denoted hereafter $y(\mathbf{x})$. The latter is astride of the boundary, and has the property that $K\ d(\mathbf{x}, y(\mathbf{x})) + \mathit{lfs}(y(\mathbf{x}))$ is the current known minimum value of

the sizing field $\mu(\mathbf{x})$. The candidate cells are maintained in a priority queue ordered by their current estimated value of μ. This queue is initialized with all grid cells that are neighbors of a grid cell intersecting $\partial\Omega$. At each step of the marching process we pop the candidate cell with minimum μ value out of the queue, set its final sizing field value to μ, and push other possible adjacent candidates in the queue with the same buddy cell. This fast marching method will thus propagate values of μ from the initial boundary to the inside of the domain. Note that this could introduce an approximation in the evaluation of the sizing field, since the boundary cell $c(\mathbf{x})$ such that $K\ d(\mathbf{x}, c(\mathbf{x})) + lfs(c(\mathbf{x}))$ is globally minimal might not be among the buddy cells of $\mathbf{x}$'s neighbors. We argue that the error is negligible. The reason is that the set of points $\mathbf{p}$ that have a same buddy cell $\mathbf{y}$ is star-shaped around $\mathbf{y}$. Indeed, on the line segment from $\mathbf{p}$ to $y(\mathbf{p})$, the function $\lambda(\mathbf{s}) = K\ d(\mathbf{s}, y(\mathbf{p})) + lfs(y(\mathbf{p}))$ decreases with speed K; as μ is K-Lipschitz, we have that $\lambda \leq \mu$, therefore $\lambda = \mu$ since μ is the minimum over all $\mathbf{y} \in \partial\Omega$, and finally $y(\mathbf{s}) = y(\mathbf{p})$. Hence, the first grid cell $\mathbf{q}$ met by the ray $\mathbf{p} - y(\mathbf{p})$ is most likely such that $y(\mathbf{q}) = y(\mathbf{p})$. One then has $\mu(\mathbf{q}) \leq \mu(\mathbf{p})$; thus, $\mu(\mathbf{q})$ must have been already computed by the time $\mathbf{p}$ is taken care of. This simple procedure enables an efficient and robust initialization of our sizing field grid.

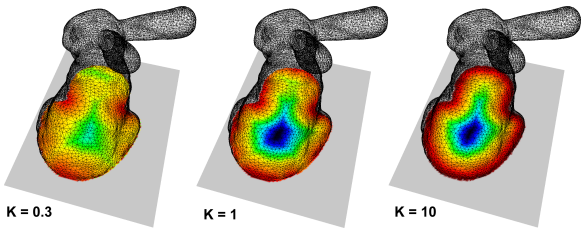


Figure 9: *Sizing fields computed for three increasing values for K. The smaller K, the smoother the grading. For large values, the ideal edge size is rapidly increasing as one moves away from the boundary.*

3.4 Initial Point Sampling

Given the potential complexity of the input boundary and the optimized 3D Delaunay triangulation in terms of geometry and topology (with possibly multiple connected components and holes), a good initialization of the tet mesh vertices is desirable. We "spread" the requested number of vertices throughout the domain while roughly matching the desired local density through error diffusion over the sizing field. This initial sampling proceeds in two passes. In order to calibrate the sampling so as to fit the vertex budget specified by the user, the first pass sums up the values $dv/\mu(\mathbf{x})^3$ for each interior node of the sizing field grid, where dv denotes the volume of the node and $\mathbf{x}$ its position. The second pass iterates over the same grid nodes in serpentine order, computing for each node its corresponding (floating point) number of initial vertices to lay down locally, quantizes this number to the nearest integer and diffuses the corresponding residual to its neighbors: this process is a straightforward extension of [Ostromoukhov 2001] to volumetric images. Although these placements do not guarantee any quality on the resulting Delaunay mesh, we achieve a local density of vertices consistent with the sizing field for a very low computational effort.

3.5 Energy Minimization

The energy minimization phase, alternating connectivity and geometry optimization, is the core of our algorithm. From the current vertex positions, the energy is minimized by computing the 3D Delaunay triangulating of these sites. For a given connectivity, the energy is further minimized by moving each *interior* vertex $\mathbf{x}_i$ to its optimal placement within its 1-ring (Eq. 8).

Boundary Vertices We must treat the current boundary vertices differently to provide adequate boundary conditions to our minimization, as well as a good isotropically-sampled approximation of the domain boundary. A simple and practical solution is to use a variant of the constrained centroidal Voronoi tessellation approach (CCVT [Du et al. 2003]). First, in order to identify the current boundary vertices, we examine each boundary quadrature sample $\mathbf{s}_i$, and locate its nearest vertex in the current mesh (through a fast kD tree query), and accumulate at that vertex the quadrature value at $\mathbf{s}_i$ times the coordinates of $\mathbf{s}_i$. Subsequently, we focus on the vertices with a non-zero quadrature sum, since those are the boundary vertices that require a specific treatment. We move these boundary vertices to the average value they each have accumulated during the pass over all quadrature samples. This position provides a good, low-cost approximation of the *centroid* of the intersection between the 3D Voronoi cell of the boundary vertex and the input boundary $\partial\Omega$. We proceed similarly for the feature quadrature samples involved in the piecewise smooth case, where we must fit sharp creases too. As demonstrated in our results (see Fig. 10 for an illustration of several steps of optimization on a simple boundary), this simple procedure results in well-shaped triangles that fit the domain boundary, and whose size is in agreement with the sizing field.

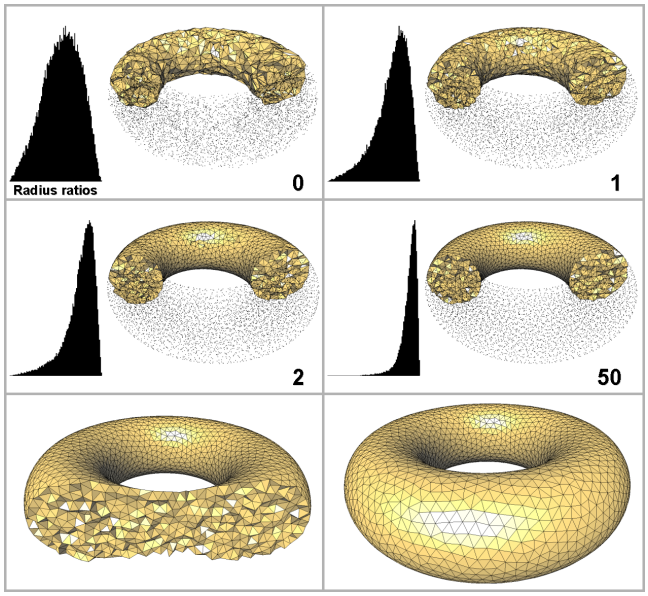


Figure 10: *Optimization steps for 1000 vertices in a torus. $100K$ quadrature samples were spread on the boundary, and the sizing field is constant to get a uniform mesh. Observe how the radius ratio distribution shifts towards 1.*

3.6 Accelerating Convergence

Although our quadratic energy minimization provides a powerful tool to design high-quality meshes, the convergence rate can be slow for large number of vertices. We have successfully experimented with the following practical shortcuts to get faster results.

Delaunay Refinement Direct minimization of more than $100,000$ vertices can be computationally expensive. Instead, we prefer starting the energy minimization with a much smaller number of vertices. Before it even reaches a minimum, we increase the number of vertices by adding a specified fraction (typically, 50%) of the vertices at a subset of the Voronoi centers of the current mesh. To select the latter subset, we sort all tets by decreasing size of circumradius divided by the local desired edge length (to know where refinement is most needed). After another round of minimization steps, we repeat this procedure, until the requested number of vertices is reached. The speed-up is considerable, while achieving the same final quality.

Selective Optimizations A straightforward improvement of vertex position update optimizes *only* the vertices adjacent to bad quality tets (say, with a radius ratio less than 0.3) and their immediate neighbors. Although no theoretical guarantees back up this

trick, it works remarkably well in practice. We recommend switching to such a selective optimization once the full-blown optimization steps are relatively small in amplitude.

Boundary Vertex Jittering As expected from our energy, the inside tets are well shaped after minimization. However, because of the boundary constraints that we must satisfy, a few slivers can remain *adjacent* to boundary vertices. We have implemented a fast "jittering" of these points; in order to snap a sliver, we slightly move one of its adjacent boundary vertices in the local tangent plane. Similar in spirit to the more general procedure of sliver exudation [Cheng et al. 1999], but for the easier case of tets on boundaries, this jittering suffices to remove the remaining slivers.

Vertex Teleportation or Insertion We also recommend sporadically removing the vertex with the smallest Voronoi cell w.r.t. the desired edge length (*i.e.*, in the densest region), and inserting it at the centroid of the interior tet with the worst radius ratio. Such tunnelling of vertices is particularly useful when tight control over the worst element is required. If the vertex budget does not have to be maintained, one can directly add vertices inside the worst tets.

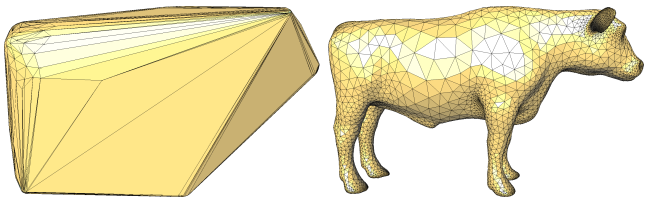


Figure 11: *Final Extraction: As the Delaunay triangulation covers the convex hull, the last stage of our algorithm must extract the inside tets.*

3.7 Final Mesh extraction

To produce the final mesh, we need to peel off the Delaunay tets of our mesh that are outside the domain (remember that a Delaunay mesh triangulates the convex hull of the vertices—see Fig. 11). A first approach is to consider the Delaunay triangulation restricted to the input domain, by tagging a tet as *outside* when its circumsphere center is located outside Ω. Similar in spirit to the Cocone algorithm [Amenta et al. 200], we consider instead the Delaunay triangulation restricted to a slightly *thicker* version of the input domain Ω. For each tet initially tagged outside, we compute the ratio between the distance d from the circumsphere center to the boundary $\partial\Omega$ and the circumradius; if this ratio is smaller than a predefined threshold (0.4 in all our experiments) we tag the tet inside.

4 Results and Discussions

The figures in this paper illustrate the robustness and versatility of our technique: our implementation can handle large and/or complex domains of arbitrary topology in a matter of minutes. Although a visual inspection cannot provide a thorough assessment of our results, all the cutaway views as well as the radius ratio distributions that we obtained exhibit high quality tet shapes throughout the domain. In contrast to many other methods we *do not* a priori assign vertices to be either on the boundary or in the interior. It is the minimization procedure that will make them stick to boundary or not, driven by the sizing field and number of vertices required. This feature partially explains the quality of the results, since the mesh is not constrained to a given budget of boundary vertices. Also, our experience shows that global optimization of the connectivity through a Delaunay triangulation renders the results significantly better: this handling of the connectivity is possibly the sharpest departure from common approaches that perform local updates only.

Results can be obtained in a matter of seconds or minutes. For instance, Fig. 10 was obtained in 16 seconds (for the 50 iterations, which include a Delaunay triangulation and the vertex position optimizations at *each* iteration) on a Pentium IV 3GHz. A more complex model, such as the hand in Fig. 12 requires on average 2.1

Figure 12: *Scanned hand: on this highly detailed mesh ($36K$ vertices, $174K$ tets, with color-coded* lfs*), mesh gradation is a must: a uniform mesh fine enough to capture the surface details would have millions of tets; Instead, our algorithm can reproduce all the fine surface features while using large tets inside the domain. Sizing field parameter: $K = 1$. As the radius ratio distribution shows, there are no degenerate tets. Worst radius ratio $= 0.29$, average radius ratio $= 0.86$, average dihedral angle $= 70^o$. Mean symmetric distance from input boundary: 0.024% of the bounding box.*

seconds per iteration, including Delaunay triangulation, boundary quadrature, and vertex updates for the $36K$ vertices. For good meshes, 10 to 20 iterations are sufficient, but we often increase this number to 50, and use the speed-ups described in Section 3.6 for a final high quality result. Although very few timings are available for previous optimization methods, we consider the time involved in our technique for mesh design practical. Notice that we can also deal with sharp features, as Figure 16 demonstrates—a full treatment treatment of such mechanical would however require a good Constrained Delaunay mesher.

Figure 13 demonstrates the quality of approximation of the boundary for the Bunny model, for increasing vertex budgets ranging from 1K to 10K vertices. We plot the mean symmetric ($\mathcal{L}^2$) distance in percentage of the bounding box against the number of sites, as well as a color-coded illustration of the approximation error.

Judging the quality of the results is, however, a difficult task. First, many (often contradictory) quality measures have been proposed over the years. Second, when averages of radius ratios are given, they do not tell the whole story: many slivers can be present even

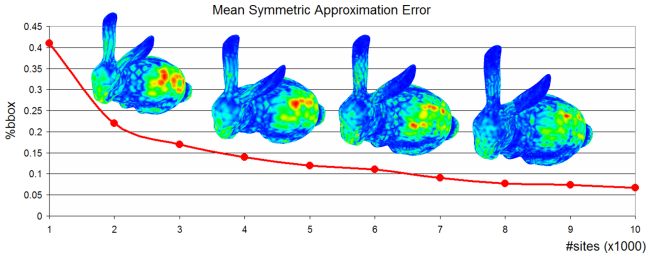


Figure 13: *Mean symmetric approximation error against the number of sites measured with Metro. The approximation error is expressed in percentage of the bounding box. The four bunnies shown correspond respectively to 2, 4, 6 and 8K sites, with their approximation error in false colors.*

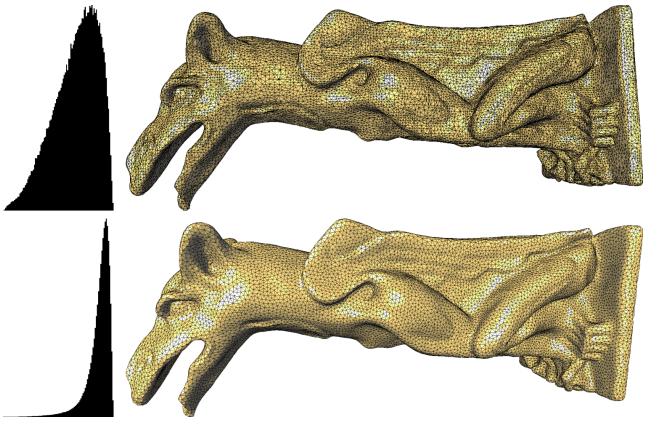


Figure 14: *Gargoyle: a comparison with [Cutler et al. 2004] gives further evidence of the quality of our results. Our distribution in terms of radius ratios (one of the fairest measures [Shewchuk 2002a]) is far superior to a standard optimization technique (mesh courtesy B. Cutler).*

when the average is high. Finally, we could not find or get tet meshes of usual CG models (such as the bunny) or of canonical shapes, aside from the results presented in [Cutler et al. 2004] (see Fig. 14 for comparison). nevertheless, we provide typical numbers of our results in Table 1 for comparative purposes. The distribution curves given in most of our figures also indicate how well shaped most of the tets are. We also indicate relevant numbers in each caption for completeness.

Finally, in Fig. 15 we compare our optimization technique with the unit-mesh approach [Frey and George 2000] used in commercial meshers. This technique has been applied on a high-quality uniform input boundary mesh of the bunny generated using the technique presented in [Surazhsky et al. 2003], resulting in 275K tets and 49K vertices in 12.5s (the number of vertices cannot be specified and results from the conforming of the boundary). As this unit-mesh approach splits long edges into smaller equal-length edges during the meshing process, the final mesh exhibits directional aliasing in the form of lines of tets, potentially detrimental to the simulation of isotropic phenomena. To provide a fair comparison, we used our technique with a uniform sizing field and the same number of final vertices. Our mesh was obtained in 4 min. Both distributions of radius ratios show no slivers; however, our approach yields better shaped tets overall, albeit at the price of a higher cost.

Limitations Our design choice to approximate the input boundary instead of conforming to it can also be seen as a limitation for certain applications. Additionally, we do not have theoretical bounds on the quality measures of the resulting tets. However, our results indicate that in practice, our minimization procedure generates well-shaped tets inside the domain, with better radius ratio distribution curves than any of the tet meshes we came across. Note that this high quality, most desirable for simulation purposes, naturally comes with higher computational cost than typical greedy (*e.g.* Delaunay refinement) methods.

Model	#v	#tets	min/average radius ratio	$\mathcal{L}^2$ error (%bb)
Torus	1K	4K	0.42 / 0.88	0.17
Bunny	49K	275K	0.37 / 0.89	0.04
Hand	36K	174K	0.29 / 0.86	0.024
Gargoyle	50K	260K	0.23 / 0.88	0.053
Fandisk	3K	14K	0.29 / 0.87	0.021

Table 1: *Min/average radius ratios and mean approximation errors of the input boundary obtained on a series of models. "Bunny" refers to Fig. 15*

5 Conclusions

We have introduced a novel approach to the construction of high-quality, isotropic tetrahedral meshes. Based on a sound variational principle, our technique provides a robust mesh design tool that can accommodate requirements on the final number of vertices and on the mesh gradation, for arbitrary domain complexity and topology. We demonstrated the scalability of our approach by meshing large, complex domains, even with sharp features. In future work, we wish to explore how to extend our approach to anisotropic meshing using not just a mass density, but a tensor field. Other boundary conditions for our optimization could also be studied.

Acknowledgments The authors wish to thank Peter Schröder as one of the instigators of this project. Many thanks to Alexandre Olivier-Mangon and George Drettakis for providing us with the torso model. Our gratitude also goes to Joe Warren, Sean Mauch, Peter Krysl, Fehmi Cirak and Tamer, Barbara Cutler, Steve Oudot, Sylvain Pion, and Andreas Fabri for precious help along the way. Sponsors include NSF (CARGO DMS-0221669 and DMS-0221666, CAREER CCR-0133983, and ITR DMS-0453145), DOE (DE-FG02-04ER25657), the EU Network of Excellence AIM@SHAPE (IST NoE No 506766), and Pixar.

References

AMENTA, N., AND BERN, M. 1998. Surface Reconstruction by Voronoi Filtering. In *Proc. of* 14^{th} *Symp. on Computational Geometry (SCG'98)*, 39–48.

AMENTA, N., CHOI, S., DEY, T., AND LEEKHAU, N. 200. A Simple Algorithm for Homeomorphic Surface Reconstruction. In *Proceedings of the Symposium on Computational geometry*, 213–222.

BOISSONNAT, J.-D., AND OUDOT, S. 2005. Provably Good Sampling and Meshing of Surfaces. *Graphical Models (special issue on Solid Modeling)*. To appear.

BOROUCHAKI, H., GEORGE, P., HECHT, F., LAUG, P., AND SALTEL, E. 1997. Delaunay mesh generation governed by metric specifications. Part 1 : Algorithms. *Finite Elements in Analysis and Design 25*, 61–83.

BOROUCHAKI, H., GEORGE, P., AND MOHAMMADI, B. 1997. Delaunay mesh generation governed by metric specifications. Part 2 : Application examples. *Finite Elements in Analysis and Design 25*, 85–109.

CAREY, G. F. 1997. *Computational Grids: Generation, Adaptation, and Solution Strategies*. Taylor & Francis eds.

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

CHEN, L. 2004. Mesh smoothing schemes based on optimal Delaunay triangulations. In *Proceedings of 13th International Meshing Roundtable*, 109–120.

CHENG, S. W., AND DEY, T. K. 2002. Quality meshing with weighted Delaunay refinement. In *Proc.* 13^{th} *ACM-SIAM Sympos. Discrete Algorithms*, 137–146.

CHENG, S.-W., AND POON, S.-H. 2003. Graded conforming Delaunay tetrahedralization with bounded radius-edge ratio. In *Proc, of the* 14^{th} *ACM-SIAM Symposium on Discrete algorithms (SODA)*, 295–304.

CHENG, S.-W., DEY, T. K., EDELSBRUNNER, H., FACELLO, M. A., AND TENG, S.-H. 1999. Sliver Exudation. In *Proc. 15th ACM Symp. Comput. Geom.*, 1–13.

CHENG, S.-W., DEY, T. K., RAMOS, E., AND RAY, T. 2004. Quality Meshing for Polyhedra with Small Angles. In *Proc. of ACM Symp. on Comp. Geom.*, 290–299.

COHEN-STEINER, D., DE VERDIERE, E. C., AND YVINEC, M. 2002. Conforming Delaunay triangulations in 3D. In *Proc. of Symp. on Comp. Geom.*, 237–246.

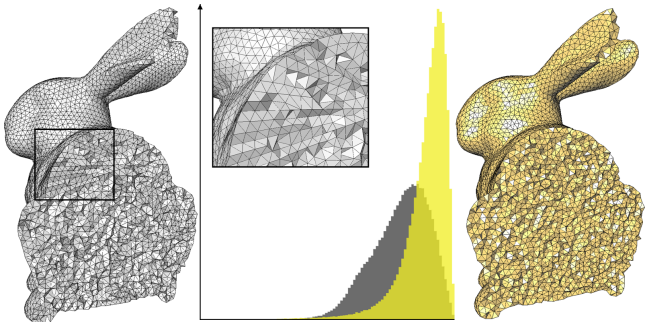


Figure 15: *Comparison of our method (right, obtained in 4 minutes) with the unit mesh approach [Frey and George 2000] (left, obtained in 12 seconds). To make a fair comparison we provide as input a uniform mesh and specify a uniform density for both methods. The distribution of radius ratios (middle) demonstrates the quality of our resulting tetrahedra.*

COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational Shape Approximation. *ACM Trans. on Graphics (SIGGRAPH)*, 905–914.

CUTLER, B., DORSEY, J., AND MCMILLAN, L. 2004. Simplification and Improvement of Tetrahedral Models for Simulation. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 95–104.

DU, Q., AND WANG, D. 2003. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *International Journal on Numerical Methods in Engineering 56(9)*, 1355–1373.

DU, Q., GUNZBURGER, M., AND JU, L. 2003. Constrained Centroidal Voronoi Tessellations for Surfaces. *SIAM J. Sci. Comput. 24*, 5, 1488–1506.

EPPSTEIN, D., 2001. Global optimization of mesh quality. Tutorial at the 10^{th} Int. Meshing Roundtable, Newport Beach.

FABRI, A., GIEZEMAN, G.-J., KETTNER, L., SCHIRRA, S., AND SCHÖNHERR, S. 2000. On the Design of CGAL, a Computational Geometry Algorithms Library. *Softw. – Pract. Exp. 30*, 11, 1167–1202. www.cgal.org.

FREITAG, L., AND OLLIVIER-GOOCH, C. 1996. A comparison of Tetrahedral Mesh Improvement Techniques. In *Proc. of 6th Int. Meshing Roundtable*, 87—1000.

FREY, J. L., AND GEORGE, P. L. 2000. *Mesh Generation: Applications to Finite Elements*. Hermès, Paris.

HARDIN, D. P., AND SAFF, E. B. 2004. Discretizing Manifolds via Minimum Energy Points. *Notices of the AMS 51(10)*, 1186–1194.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. "Mesh Optimization". *ACM Trans. on Graphics (SIGGRAPH)*, 19–26.

KRYSL, P., AND ORTIZ, M. 2001. Variational Delaunay Approach to the Generation of Finite Element Meshes. *Int. J. for Num. Meth. in Eng. 50(7)*, 1681–1700.

LI, X.-Y., AND TENG, S.-H. 2001. Generate Sliver Free Three Dimensional Mesh. In *Proc.* 12^{th} *ACM-SIAM Sympos. Discrete Algorithms*.

LI, X.-Y., TENG, S.-H., AND UNGOR, A. 2000. "Biting: Advancing Front Meets Sphere Packing". *Int. J. on Num. Methods in Eng. 49*, 1, 61–81.

LLOYD, S. P. 1957. Least Squares Quantization in PCM's. Tech. rep., Bell Telephone Laboratories, Murray Hill, NJ.

MITCHELL, S., AND VAVASIS, S. 2000. Quality Mesh Generation in Higher Dimensions. *SIAM J. Sci. Comput. 29*, 1334–1370.

MOLINO, N., BRIDSON, R., TERAN, J., AND FEDKIW, R. 2003. A Crystalline, Red Green Strategy for Meshing Highly Deformable Objects with Tetrahedra. In *Proceedings of the 12th International Meshing Roundtable*, 103–114.

OSTROMOUKHOV, V. 2001. A simple and efficient error-diffusion algorithm. In *Proceedings of ACM SIGGRAPH*, 567–572.

OWEN, S. J. 1998. A Survey of Unstructured Mesh Generation Technology. In *Proceedings of the 7th International Meshing Roundtable*, 239–267.

QUADROS, W. R., SHIMADA, K., AND OWEN, S. J., 2004. 3D Discrete Skeleton Generation by Wave Propagation on PR-Octree for Finite Element Mesh Sizing. *Poster*, Solid Modeling Conference.

RUPPERT, J. 1993. A New and Simple Algorithm for Quality 2-Dimensional Mesh Generation. In *Proc. of the* 4^{th} *ACM/SIAM Symp. on Disc. Algo. (SODA)*, 83–92.

SHEWCHUK, J. R. 1998. A Condition Guaranteeing the Existence of Higher-Dimensional Constrained Delaunay Triangulations. In *Proc.* 14^{th} *Annu. ACM Sympos. Comput. Geom.*, 76–85.

SHEWCHUK, J. R. 1998. Tetrahedral mesh generation by Delaunay refinement. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, 86–95.

SHEWCHUK, J. 2002. What Is a Good Linear Element? Interpolation, Conditioning, and Quality Measure. In *Proc. of* 11^{th} *Int. Meshing Roundtable*, 115–126.

SHEWCHUK, J. R. 2002. Delaunay Refinement Algorithms for Triangular Mesh Generation. *Computational Geometry: Theory and Applications 22*, 21–74.

SURAZHSKY, V., ALLIEZ, P., AND GOTSMAN, C. 2003. Isotropic Remeshing of Surfaces: a Local Parameterization Approach. In *Proc. of* 12^{th} *Int. Meshing Roundtable*.

TENG, S.-H., WONG, C. W., AND LEE, D. T. 2000. Unstructured Mesh Generation: Theory, Practice, and Perspectives. *International Journal Computational Geometry and Applications 10*, 3 (June), 227–266.

WARREN, J., SCHAEFER, S., HIRANI, A., AND DESBRUN, M., 2004. Barycentric Coordinates for Convex Sets. Preprint.

A Lipschitz Sizing Field

We wish to prove that the sizing field defined in Eq. (9) is both K-Lipschtiz and maximal. Because we want the function to be K-Lipschitz and agree with *lfs* on the boundary, one can easily show the following property:

$$\mu(x) \leq \inf_{s \in \partial\Omega} \left[K\ d(x,s) + \mathit{lfs}(s) \right].$$

We now need to show that the rhs is K-lipschitz and coincides with the *lfs* on the boundary: if so, the rhs will be the maximal sizing field we seek.

For $\mathbf{x} \in \Omega$, let

$$y(\mathbf{x}) = \mathrm{argmin}_{\mathbf{s} \in \partial\Omega} \left[K\ d(\mathbf{x},\mathbf{s}) + \mathit{lfs}(\mathbf{s}) \right].$$

If $\mathbf{x}'$ is in Ω, we have by definition:

$$\begin{aligned} \mu(\mathbf{x}') &\leq K\ d(\mathbf{x}', y(\mathbf{x})) + \mathit{lfs}(y(\mathbf{x})) \\ &\leq K\ d(\mathbf{x}', \mathbf{x}) + K\ d(\mathbf{x}, y(\mathbf{x})) + \mathit{lfs}(y(\mathbf{x})) \\ &\leq K\ d(\mathbf{x}', \mathbf{x}) + \mu(\mathbf{x}) \end{aligned}$$

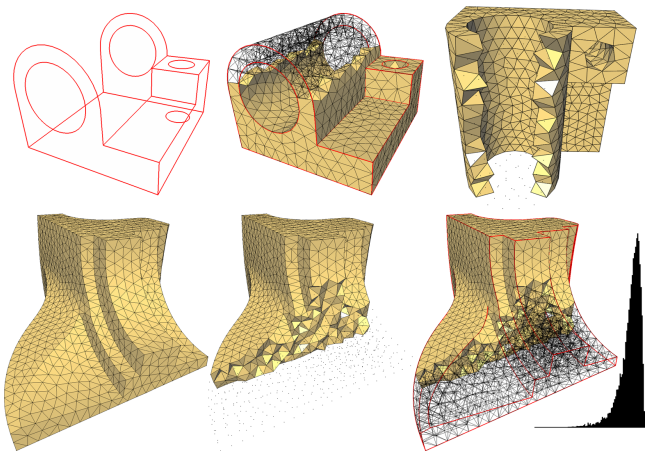
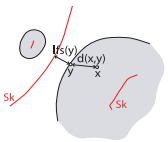


Figure 16: *Mechanical parts: Even in the presence of sharp features, our tet meshing algorithm exhibits excellent behavior for low or high vertex count while capturing features and corners remarkably well. We tagged edges as sharp features if their dihedral angles are more than* 20^o*—more sophisticated segmentation techniques could of course be used. (Top, middle) Joint model (*$1.2K$ *vertices); (Bottom) Fan disk model (*$3K$ *vertices, mean/max symmetric distance from input boundary: 0.021%/0.5% of the bounding box), along with its radius ratio distribution. Notice the good aspect ratio of both tets (see cutaway views) and surface triangles. A constant sizing field has been used for both models to obtain uniform tet meshes.*

which shows that μ is K-Lipschitz. Since *lfs* is 1-Lipschitz, we cannot hope that μ coincides with *lfs* on $\partial\Omega$ unless K is at least 1. If so, then for $\mathbf{x} \in \partial\Omega$ we have

$$K\ d(\mathbf{x},\mathbf{y}) + \mathit{lfs}(\mathbf{y}) \geq \mathit{lfs}(\mathbf{x})$$



for all $\mathbf{y} \in \partial\Omega$, with equality when $\mathbf{y} = \mathbf{x}$. Thus, μ does coincide with *lfs* on $\partial\Omega$. Note that when $K = 1$, $\mu(\mathbf{x})$ boils down to the length of the shortest path from x to the medial axis of $\partial\Omega$ while passing by a point on $\partial\Omega$. When K is less than 1, we get that $\mu(x)$ can be less than $\mathit{lfs}(x)$ on the boundary due to the Lipschitz constraint; however, the gradation is respected and the boundary sampling will be better than what is necessary: it is therefore still a good choice of sizing field.

B Transforming the Energy E_{ODT}

Let us start with the definition:

$$E_{\text{ODT}} = ||f - f^{\text{primal}}_{\text{PWL}}||_{\mathcal{L}^1} = \sum_j \int_{T_j} |f - f^{\text{primal}}_{\text{PWL}}|. \qquad (10)$$

In the tet T_j with vertices $\mathbf{x}_i$ $i = 1 \ldots 4$, the error function can be expressed as a function of the barycentric coordinates $\lambda_i(\mathbf{x})$:

$$|f(\mathbf{x}) - f^{\text{primal}}_{\text{PWL}}(\mathbf{x})| = \sum_i \lambda_i(\mathbf{x})\mathbf{x}_i^2 - \mathbf{x}^2 = \sum_i \lambda_i(\mathbf{x})\,(\mathbf{x}_i - \mathbf{x})^2\,. \qquad (11)$$

Notice that Eq. (3) is easily derived from this last expression by plugging it into Eq. (10). Rewriting $\mathbf{x}_i - \mathbf{x}$ as $((\mathbf{x}_i - \mathbf{c}_j) + (\mathbf{c}_j - \mathbf{x}))$, where $\mathbf{c}_j$ is the circumcenter of T_j, and plugging it into Eq. (10), we get the following confirmation of Eq. (7):

$$E_{\text{ODT}} = \sum_j \int_{T_j} \left(R_j^2 - ||\mathbf{x} - \mathbf{c}_j||^2 \right) d\mathbf{x} = \sum_j \left(|T_j| R_j^2 - \int_{T_j} ||\mathbf{x} - \mathbf{c}_j||^2 d\mathbf{x} \right).$$

C Updates as Weighted Circumcenters

Notice that the energy E_{ODT} inside a tet T is always extremal at the circumcenter $\mathbf{c}_T$. As a consequence, the optimal position of a vertex that has only four neighbors is exactly *at* $\mathbf{c}_T$. Using Eq. (5) 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 on $\mathbf{x}_p$, we get:

$$\begin{aligned} \mathbf{c}_T = \mathbf{x}_p - \frac{1}{2\,|\Omega_i|} (\quad & \nabla_{\mathbf{x}_p} |T|) \left[\textstyle\sum_{\mathbf{x}_k \in T} ||\mathbf{x}_p - \mathbf{x}_k||^2 \right] + \mathrm{F}(\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_r) \\ & + \mathrm{F}(\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_s) + \mathrm{F}(\mathbf{x}_p, \mathbf{x}_r, \mathbf{x}_s) \ ) \end{aligned}$$

where the extra terms on the rhs only depend on each face of the tet. Applying this formula to an arbitrary 1-ring centered on $\mathbf{x}_p$, the face terms cancel each other if we sum the contributions from all the tets, simplifying the expression drastically, and resulting in Eq. (6).