CGALmesh: a Generic Framework for Delaunay Mesh Generation

Clément Jamin, Pierre Alliez, Mariette Yvinec, Jean-Daniel Boissonnat

CGALmesh: a Generic Framework for Delaunay Mesh Generation

Clément Jamin, Pierre Alliez, Mariette Yvinec, Jean-Daniel Boissonnat

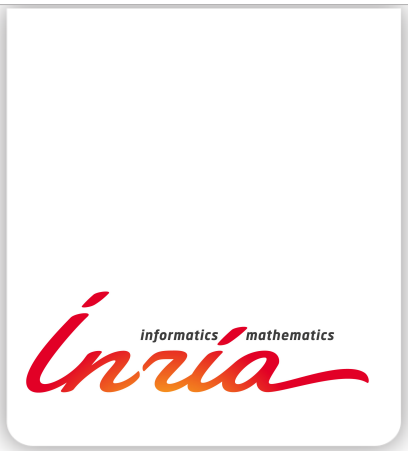
RESEARCH REPORT

N° 8256

January 2014

Project-Teams Geometrica

ISSN 0249-6399 ISRN INRIA/RR--8256--FR+ENG



CGALmesh: a Generic Framework for Delaunay Mesh Generation

Clément Jamin*†, Pierre Alliez*, Mariette Yvinec*, Jean-Daniel Boissonnat*

Project-Teams Geometrica

Research Report n° 8256 — January 2014 — 31 pages

Abstract: CGALmesh is the mesh generation software package of the Computational Geometry Algorithm Library (CGAL). It generates isotropic simplicial meshes – surface triangular meshes or volume tetrahedral meshes – from input surfaces, 3D domains as well as 3D multi-domains, with or without sharp features. The underlying meshing algorithm relies on restricted Delaunay triangulations to approximate domains and surfaces, and on Delaunay refinement to ensure both approximation accuracy and mesh quality. CGALmesh provides guarantees on approximation quality as well as on the size and shape of the mesh elements. It provides four optional mesh optimization algorithms to further improve the mesh quality. A distinctive property of CGALmesh is its high flexibility with respect to the input domain representation. Such a flexibility is achieved through a careful software design, gathering into a single abstract concept, denoted by the oracle, all required interface features between the meshing engine and the input domain. We already provide oracles for domains defined by polyhedral and implicit surfaces.

Key-words: Mesh generation, Delaunay refinement, mesh optimization, isosurface extraction

Author's addresses: Inria Sophia Antipolis - Méditerranée 2004 route des Lucioles BP69 06902 Sophia Antipolis, France; e-mail: firstname.lastname@inria.fr

* Inria
† Université Lyon 1, LIRIS, UMR5205

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

CGALmesh : un cadre générique pour la génération de maillages de Delaunay

Résumé : CGALmesh est le composant logiciel de génération de maillages de la bibliothèque CGAL (Computational Geometry Algorithm Library). Ce composant génère des maillages simpliciaux isotropes – maillages de surface triangulaires ou maillages volumiques tétraédriques – à partir de domaines à mailler fournis en entrée. Ces domaines peuvent être des surfaces ou des domaines 3D, avec ou sans arêtes vives. L'algorithme s'appuie sur la notion de triangulation de Delaunay restreinte pour approcher les domaines et les surfaces, et sur le raffinement de Delaunay pour fournir une approximation précise et s'assurer de la qualité des maillages produits. CGALmesh offre des garanties sur la qualité d'approximation, ainsi que sur la taille et la forme des éléments du maillage. Quatre algorithmes d'optimisation de maillage sont proposés pour améliorer la qualité du maillage. Une propriété distinctive de CGALmesh est sa grande flexibilité quant aux domaines fournis en entrée. Cette flexibilité est obtenue grâce à une conception logicielle générique qui rassemble en un seul concept abstrait, appelé oracle, toute l'interface nécessaire entre le mailleur et le domaine fourni en entrée. Des oracles pour les domaines définis par des polyèdres et par des surfaces implicites sont fournis avec la bibliothèque.

Mots-clés : Génération de maillages, raffinement de Delaunay, optimisation de maillages, extraction d'isosurfaces

1 Introduction

Surface triangular meshes and 3D tetrahedral meshes play a central role in computational engineering for simulating and visualizing physical phenomena. They are also commonplace for modeling and animating complex scenes in special effects or multimedia applications. For rendering, surface meshes must provide a good approximation of object boundaries. For numerical simulation, 3D volume meshes must fulfill additional constraints over the shape, orientation and size of the elements.

Meshes are used not only for rendering and simulation, but also as means to solve other problems. Example of such problems include image segmentation for medical applications [46] and spatio-temporal coherent segmentation of time-varying data sets such as beating hearts. As for images, meshes are now commonly used as central data structures for a variety of algorithms. This poses new challenges in terms of complexity and genericity of the mesh generation algorithms.

The description of a domain can take many forms, ranging from implicit surfaces to triangular meshes through 3D images and point sets. This variety of inputs is a notoriously enduring challenge for mesh generation algorithms. Most techniques are designed to handle a single type of domain description and are not suited to other descriptions. Choosing the proper algorithm often reduces to selecting the one which can deal with the input description. Such choice may be too limited, especially when restricting to open-source software solutions. For this reason, each type of input corresponds to a specific methodology: meshing from implicit surfaces is referred to as *isosurface extraction*, meshing from point sets as *reconstruction*, meshing from triangular meshes as *remeshing*, to name a few. These methodologies correspond to different research areas, each with its own challenges.

1.1 Related Work

Mesh generation is an active research field. The mesh generation algorithms divide into the following meshing strategies: advancing front [52, 39], lattice-based methods [38, 36, 19], and Delaunay refinement. The latter can be further classified into two classes of algorithms. The first class of Delaunay refinement algorithms requires to mesh or remesh the input domain boundary (often a trial and error process) before meshing the input domain while preserving the boundary mesh [27, 28, 26, 56]. The second class of Delaunay refinement algorithms, our choice for CGALmesh, meshes the input surface and volume altogether in a unified manner, with the ability to mesh multi-domains and curved surfaces while approximating them [43, 48, 45]. For a detailed review of mesh generation methods we refer the reader to comprehensive books and surveys [32, 55, 17, 25].

There is a plethora of commercial solutions for the automatic generation of 3D tetrahedral meshes. Some of them are used in CAD or simulation software packages for computational fluid dynamics or mechanical analysis [33]. The open source offer is more limited, with, e.g., NETGEN from the University of Linz (Advancing Front) [53], GRUMMP from the University of British Columbia (boundary-preserving, Delaunay-based) [42], QMG from the Cornell University [58], and TETGEN from the Weierstrass Institute of Berlin [56]. These solutions all share the common requirement that the domain boundary must be provided in a specific form, *e.g.* as a defect-free polyhedral surface mesh or parametric surface, and hence are not so generic. A more flexible design appears, e.g., in the form of a common geometry model (CGM) [34]. CGM offers a wide range of functions, but is restricted to the class of models that are the output of solid modelers. Furthermore, most software solutions require a quality surface mesh as input (with well-shaped triangles) as they preserve the input surface mesh exactly, instead of approximating

it. The preparation of quality input surface meshes from defect-laden inputs requires applying a sequence of algorithms, which is notoriously labor-intensive and prone to robustness issues.

The main paradigm behind our algorithms is the refinement of a Delaunay triangulation restricted to the input domain and surfaces (in the sequel the input surfaces refer to both subdividing and boundary surfaces). The restricted Delaunay triangulation approach, detailed in Section 2.1, consists in extracting from the 3D Delaunay triangulation a subset of its simplices: vertices, facets approximating the input surfaces, edges approximating the sharp creases (if any), and tetrahedra within the input domain. The refinement process is devised to sample the sharp features, surfaces as well as 3D volumes so that the restricted Delaunay triangulation provides a faithful approximation of surfaces and sharp features, together with a subdivision of the input 3D volume with well-shaped tetrahedra. The paradigm of *restricted Delaunay refinement* has received a special attention due to its versatility and theoretical foundations for mesh generation and shape approximation. It is amenable to algorithms that are proven to terminate and that provide control on the size and quality of the mesh elements. Delaunay refinement has been first used in 2D [21], then in 3D for polyhedral domains [41], for smooth surfaces [22], for 3D domains bounded by smooth surfaces [43, 9] as well as for 3D domains bounded by piecewise smooth surfaces [48, 18, 16, 10].

The restricted Delaunay refinement approach is more than just a methodological feature of our approach. It is amenable to a novel functionality of the mesh generation software: in case of a multi-component domain, the restricted Delaunay refinement provides in a single refinement process a mesh approximating all components of a subdivided domain and of its subdividing and bounding surfaces. This approach departs from the common pipeline where a surface mesh is first generated for each input surface, then 3D meshes respecting these surface meshes are generated separately for each domain component before merging them to form the final mesh (Figure 1).

Despite its versatility, Delaunay refinement alone is facing two issues. The first issue occurs in the presence of sharp features (creases, corners, darts, cusps, tips) on the input domain boundary. Sharp features subtending either small or large angles may jeopardize the termination of the refinement process. In CGALmesh [2], sharp features are handled through so-called protecting balls placed before refinement [16] so as to ensure both the termination of the refinement process and a faithful approximation of the sharp features in the final mesh. The second issue is related to the shape quality of the mesh elements. More specifically, the Delaunay refinement algorithm is shown to terminate when the shape quality governing the refinement is an upper bound on the radius-edge ratio of the mesh elements (surface triangles and volume tetrahedra). The radius-edge ratio is the ratio between the circumradius of the mesh element and the length of its shortest edge. Unfortunately, a bound on radius-edge ratios does not provide a bound on the dihedral angle of tetrahedra. Applied to a 3D domain, the Delaunay refinement process yields a final mesh that may include quasi-degenerate tetrahedra of a special kind, referred to as slivers. Slivers are ill-shaped tetrahedra formed by four vertices close to a circle and roughly equally spaced on this circle; slivers have small radius-edge ratio. To overcome this issue CGALmesh offers optimization processes [1, 30, 15, 57] to reduce the number of slivers.

1.2 Contributions

Our main contribution is a generic software design for mesh refinement and optimization. CGALmesh is able to handle a wide range of domains (surfaces, multi-domain volumes, sharp features), and has no equivalent in terms of flexibility. Our means to achieve such flexibility is the notion of so-called "oracle": all required interactions between the algorithms and the input domain are

embedded in a single oracle. Combined with a general restricted Delaunay refinement process, this oracle enables generating in a single refinement process the mesh of a multi-component domain, departing from the common approach which requires generating a surface mesh for each boundary/subdividing surface before generating a 3D mesh preserving those boundary meshes for each volume component.

Our notion of domain oracle departs in the sense that it is completely independent from the input representation: its design results from a careful analysis of the atomic geometric queries performed by the meshing algorithm. Our goal was to embed into the C++ concept of the oracle the minimal number of geometric queries in order to ease future implementations of this concept for a wide range of inputs. Some users have already implemented oracles for application domains ranging from geological modeling to simulation of fluid dynamics through medical imaging. We provide example implementations for common inputs such as polyhedral domains, implicit domains or domains described by grey level or segmented 3D images.

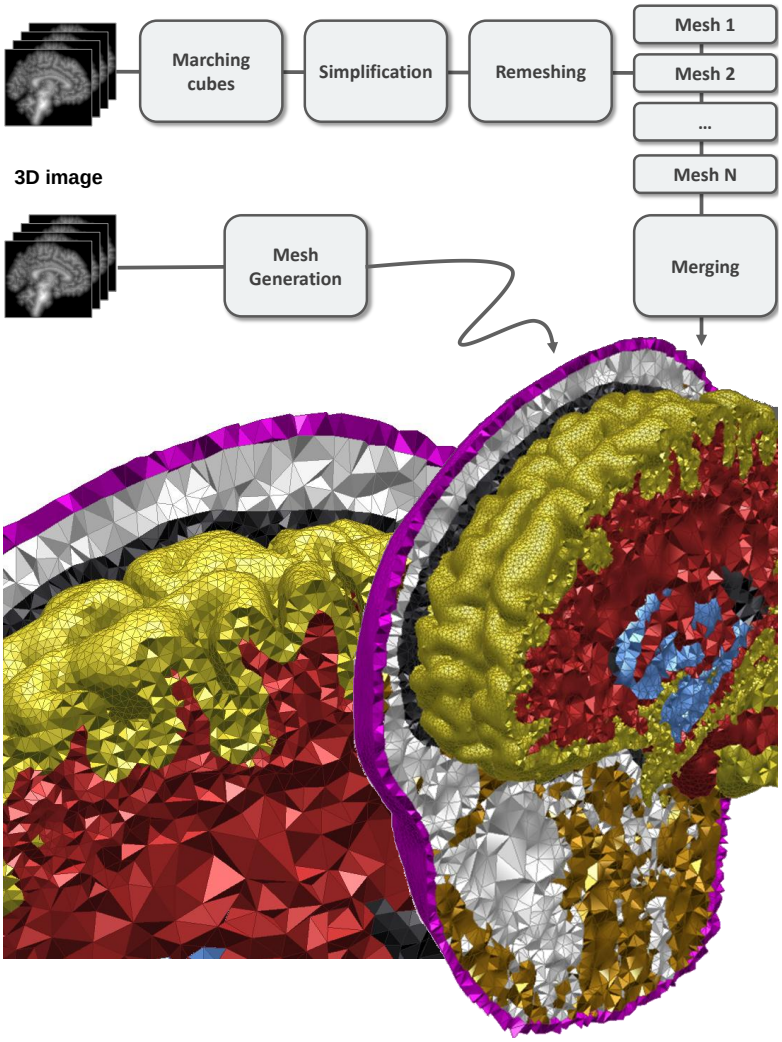


Figure 1: Mesh generation from 3D images. The usual pipeline applies marching cubes to extract isosurfaces, simplifies, remeshes, generates 3D meshes separately for each domain and finally merges the resulting 3D sub-meshes. Our mesh generation technique generates a multi-domain 3D mesh at once from 3D images. The output mesh is by construction conforming at all domain interfaces, and the whole process does not require trial-and-error processes.

2 Algorithms

The meshing algorithm includes three main phases: initialization, Delaunay refinement and optimization. The initialization phase provides an initial set of mesh vertices and deals with sharp 0 and 1-dimensional input features, if any. The Delaunay refinement phase gradually inserts new vertices into the triangulation, guided by a set of *criteria*. Finally, the optimization phase aims at improving the distribution of tetrahedron dihedral angles.

The *input* to the meshing algorithm is a domain description provided as a model of the *oracle* concept, and a set of parameters tuning to the user's needs the Delaunay refinement *criteria* and the optimization phase. The input domain Ω may be subdivided into several subdomains, each denoted by Ω_i.

The *output* of the meshing algorithm is an isotropic 3D simplicial mesh $\mathcal{M}$ formed by tetrahedral cells (volume mesh) in which triangle facets that form an approximation of the domain boundary surfaces (surface mesh) are marked. CGALmesh can thus be used to generate either 3D meshes or surface meshes.
Before stepping into the details, we describe below the notion of *restricted Delaunay triangulation*, central in CGALmesh algorithms.

2.1 Restricted Delaunay Triangulations

The meshing algorithm maintains the current set P of vertices, its 3D Delaunay triangulation and the restrictions of this triangulation to the surface and volume of the input domain. Figures 2, 3 and 4 depict the 2D counterparts of the notion of Delaunay triangulations and restricted Delaunay triangulations. More specifically, we denote by:

Ω	The input 3D domain (Figure 3a and 4a);
Ω_i	The i^{th} subdomain (Figure 4a);
$S = \bigcup_i bd(\Omega_i)$	The surface, defined as the union of all subdomain boundaries (Figure 3a and 4a);
P	The current set of vertices (Figure 3a and 4a);
$Del(P)$	The 3D Delaunay triangulation of P, *i.e.*, the triangulation T such that no point in P lies inside the circumsphere of any tetrahedra of T (Figure 2c);
$Vor(P)$	The Voronoi diagram of P, dual of $Del(P)$ (Figure 2b);
$Del_{\|\Omega}(P)$	The restriction of the Delaunay triangulation to the domain, *i.e.*, the subcomplex of $Del(P)$ formed by all tetrahedra whose circumsphere centers (circumcenters) lie inside Ω (Figure 3c and 4c);
$Del_{\|S}(P)$	The restriction of the Delaunay triangulation to the surface, *i.e.*, the subcomplex of $Del(P)$ formed by triangles whose dual Voronoi edges intersect S (Figure 3b and 4b). Those triangles are referred to as *surface facets*.

By definition, each surface facet f has at least one circumscribed ball empty of points of P and whose center lies on the surface. Such a ball, referred to as a *surface Delaunay ball* and denoted by $SDB(f)$, is centered at a point where the dual Voronoi edge intersects the surface (Figure 5a).

At any step of the algorithm, $Del_{|S}(P)$ is the current approximation of the surface S and $Del_{|\Omega}(P)$ is the current approximation of the domain Ω. In the case of a mono-domain, at the end of the Delaunay refinement, $Del_{|S}(P)$ is the boundary of $Del_{|\Omega}(P)$. In the case of a multi-domain the triangles whose two adjacent tetrahedra are from different subdomains belong to $Del_{|S}(P)$ (Figure 4). Examples of such multi-domains are depicted by Figure 1 and 10 (liver

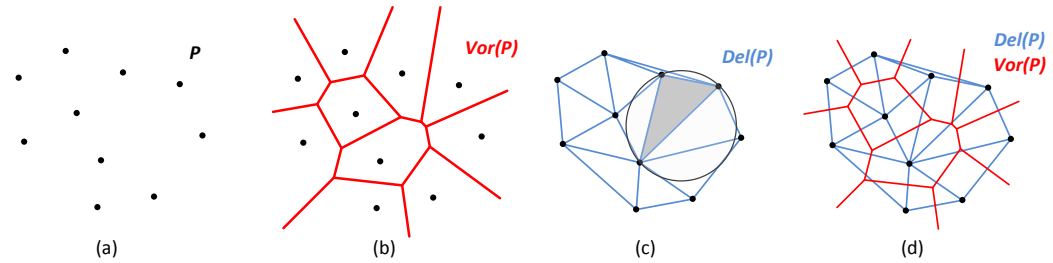


Figure 2: (a) Points P sampling the surface. (b) Voronoi diagram of P. (c) Delaunay diagram of P. We depict in black the circumcircle of the gray triangle, highlighting the empty sphere property. (d) $Vor(P)$ is the dual diagram of $Del(P)$.

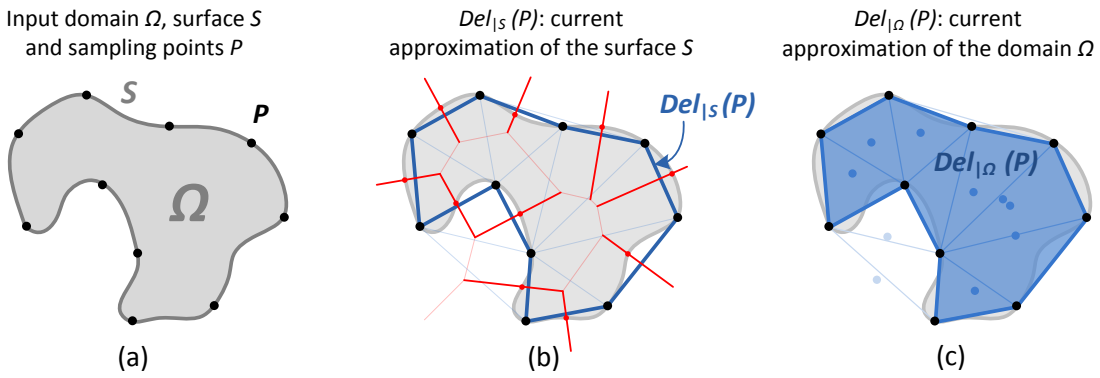


Figure 3: (a) Input domain Ω (gray area), boundary S (dark gray curve), and points P (black dots) sampling the surface. (b) $Del_{|S}(P)$, the Delaunay triangulation restricted to surface S, is formed by edges (thick blue segments) whose dual Voronoi edges (thick red segments) intersect S. (c) $Del_{|\Omega}(P)$, the Delaunay triangulation restricted to the domain Ω, is formed by triangles (in blue) whose circumcenters (blue dots) lie inside Ω.

kidney gallblader).

2.2 The Oracle

A model of the oracle concept provides a means for the meshing algorithm to probe the input domain Ω and surface S.

First, the oracle is required to construct a set of initial points on the surface S. When S contains sharp features, (creases and corners), the oracle is further required to enumerate corners and to sample creases.

The second requirement on an oracle is to provide the predicates required to extract from a Delaunay triangulation its restrictions to the domain Ω and the surface S, as well as constructors to further sample the surface when requested during refinement:

- Predicates to know whether a query point is inside Ω, and in the positive, in which subdomain Ω_i;

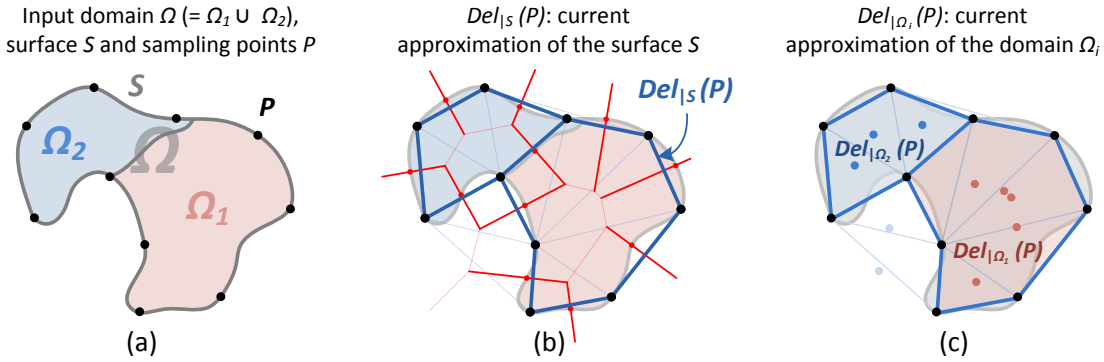


Figure 4: (a) Input domain $\Omega = \Omega_1 \bigcup \Omega_2$ (pink and light blue areas), boundary S (dark gray curve), and points P (black dots) sampling the surface. (b) $Del_{|S}(P)$, the Delaunay triangulation restricted to surface S, is formed by edges (thick blue segments) whose dual Voronoi edges (thick red segments) intersect S. (c) $Del_{|\Omega}(P)$, the Delaunay triangulation restricted to the domain Ω, is formed by triangles (in pink for Ω_1, in blue for Ω_2) whose circumcenters (blue dots) lie inside Ω.

- Predicates to know if a given line, ray or line segment query intersects S;
- Constructors to compute the intersection points between S and a line, ray or line segment query.

2.3 Criteria

The Delaunay refinement phase is governed by a set of user-tuned criteria to be met by the mesh elements (domain tetrahedron cells and surface triangle facets). The elements in $Del_{|\Omega}(P)$ and $Del_{|S}(P)$ are said to be *bad* when they do not meet some of the criteria. The refinement process computes and inserts new vertices – referred to as Steiner points – to remove these bad elements. Facet and cell criteria are designed to control the quality of the output mesh, albeit only to a certain extent in order to ensure the termination of the Delaunay refinement process. As there is additional room for tuning however, the user can provide a set of parameters to tune these criteria to the targeted application.

Facet criteria. Three criteria are used to control the properties of the surface facets:

1. **Size**: an upper bound for the radii of surface Delaunay balls (R depicted by Figure 5b), specified either as a uniform or as a non-uniform scalar sizing field (*i.e.* a function from $\mathbb{R}^3$ to $\mathbb{R}$).

2. **Shape**: an upper bound on the radius-edge ratio of surface facets equivalent to a lower bound on the angles of surface facets, (θ depicted by Figure 5b).

3. **Approximation**: an upper bound for the approximation error of surface facets (ε depicted by Figure 5b). The local approximation error for each surface facet f is estimated as the distance between the circumcenter of f and the center of its surface Delaunay ball. The bound can be specified either as a uniform or as a non-uniform scalar field.

Cell criteria. For tetrahedral volume mesh generation, two additional criteria are used to control the properties of the mesh tetrahedron cells:

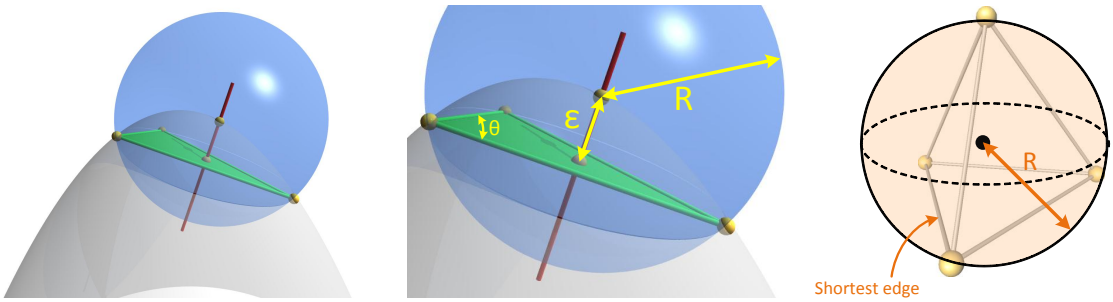


Figure 5: (a) Surface Delaunay ball. (b) Surface Delaunay ball radius (R), approximation error of the surface (ε) and triangle angles (*e.g.* θ) are used by the facet criteria. (c) Circumsphere radius (R) and shortest tetrahedron edge are used by the cell criteria.

1. **Size**: an upper bound for the circumradii of the cells in $Del_{|\Omega}(P)$ (R depicted by Figure 5c), specified either as a uniform or as a non-uniform scalar sizing field.
2. **Shape**: an upper bound for the radius-edge ratio of the cells in $Del_{|\Omega}(P)$. The radius-edge ratio relates the circumradius of a cell to the length of its shortest edge (Figure 5c). Enforcing such an upper bound on the radius-edge ratio leads to a vast majority of well-shaped (isotropic) cells in the final mesh, and possibly a few slivers.

2.4 Delaunay Refinement

Starting from a set of points computed in the initialization phase (Section 2.5), the Delaunay refinement process gradually inserts new Steiner points into P until the mesh elements (in $Del_{|S}(P)$ and $Del_{|\Omega}(P)$) meet the criteria. The location of each inserted Steiner point is computed so as to suppress from $Del_{|S}(P)$ and $Del_{|\Omega}(P)$ the surface facets or domain cells that do not meet the criteria.

Definition 1. *The Steiner point computed to suppress a bad surface facet f is the center of its surface Delaunay ball $SDB(f)$ (Section 2.1), or the center of one of those balls if there are several.*

Definition 2. *The Steiner point computed to suppress a bad cell c is the circumcenter of c.*

The Delaunay refinement phase involves two steps: the *surface mesh step* and the *volume mesh step*. The surface mesh step handles bad surface facets, adding only Steiner points onto the surface, and running until all surface facets meet the criteria. The surface mesh step first performs a *facet scan* process to initialize the restricted Delaunay triangulation $Del_{|S}(P)$ and a priority queue Q_F including the bad surface facets of $Del_{|S}(P)$.

Facet scan All triangle facets of $Del(P)$ are scanned as follows: for each triangle t whose dual edge intersects the input surface:

1. Add surface facet t to $Del_{|S}(P)$;
2. Test t against facet criteria;
3. If t does not meet all the facet criteria, add t to queue Q_F.

Queue Q_F now contains all bad facets of $Del_{|S}(P)$. The *surface mesh refinement* step is then started and runs until Q_F is empty. Note that Q_F is sorted such that facets with large surface Delaunay balls are popped and refined first.

Surface mesh refinement For each bad facet f of Q_F the following operations are performed:

1. Compute the Steiner point p of f (Definition 1);
2. Insert p into $Del(P)$;
3. Update $Del_{|S}(P)$;
4. Update Q_F by removing destroyed facets and adding newly created bad surface facets if any.

The surface mesh step terminates when Q_F is empty for the first time. The surface mesh $Del_{|S}(P)$ now meets all facet criteria. If only a surface mesh is sought after by the user, then $Del_{|S}(P)$ is the final interpolating output mesh. If a 3D mesh is sought after, the algorithm proceeds to the *volume mesh step*. The latter consists of a *cell scan* process to initialize a queue Q_C of bad cells, followed by the *volume mesh refinement* that will handle bad cells *and* bad facets through inserting Steiner points either inside Ω or on S.

Cell scan P now contains the initial points as well as all Steiner points inserted during surface mesh refinement. All bad cells are now collected as follows. For each cell c of $Del(P)$ whose dual circumcenter is in Ω:

1. Add c to $Del_{|\Omega}(P)$;
2. Test c against the cell criteria;
3. If c does not meet all the cell criteria, add it to the cell refinement queue Q_C.

Q_C now contains all bad cells of $Del_{|\Omega}(P)$. Q_C is a priority queue sorted by decreasing size of circumradius so that cells with large circumsphere radii are popped and refined first,.

Volume mesh refinement The volume mesh refinement requires maintaining the two priority queues Q_C and Q_F for bad cells and facets. The two following rules are applied as long as one of Q_C and Q_F is not empty. Rule 1 has priority and is repeatedly applied as long as Q_F is not empty.

Rule 1 If Q_F is not empty, pop a facet f then:

1. Compute Steiner point p of f (Definition 1);
2. Insert p into $Del(P)$;
3. Update $Del_{|S}(P)$ and $Del_{|\Omega}(P)$;
4. Update Q_F by removing destroyed facets and adding all newly created surface facets which do not meet all the criteria.
5. Update Q_C by removing destroyed cells, and adding newly created bad cells.

Rule 2 If Q_C is not empty, pop a cell c and perform the following operations:

1. Compute the refinement point p of c (Definition 2);

2. If p is inside the surface Delaunay ball of a surface facet f (p is said to *encroach* f), add f to Q_F and leave c in Q_C.
3. Otherwise:
   (a) Insert p into $Del(P)$;
   (b) Update $Del_{|\Omega}(P)$;
   (c) Update Q_C by removing destroyed cells, and adding newly created bad cells.

The algorithm terminates when both Rule 1 and Rule 2 do not apply anymore, *i.e.*, when both Q_F and Q_C are empty. The final 3D mesh $Del_{|\Omega}(P)$ now meets both facet and cell criteria.

2.5 Initialization and Sharp Features

The initialization phase generates through the oracle a small set of points lying on the surface, which are inserted into P. Depending on the input, these initial points may be obtained by probing the surface through, e.g., random ray shooting, by using a user-provided point set or by any other sampling method best suited to the targeted application.

During refinement Steiner points are inserted either inside Ω or on S. These Steiner points are either generated by probing Ω with Voronoi vertices (cell circumcenters) or by probing S with Voronoi edges (dual of Delaunay facets). Sharp features (creases and corners) are not probed during refinement. While probing sharp creases with Voronoi faces (dual of Delaunay edges) is conceptually possible, it is known to jeopardize the termination of the refinement when sharp features subtend small or large angles. Sharp features are thus not faithfully approximated by the refined mesh unless they are dealt with during the initialization phase (Figure 6a). To ensure an accurate representation of sharp features in the final mesh, CGALmesh relies on the so-called protecting-balls approach introduced by Cheng et al. [16]. The main idea behind the protecting-balls is to discretize sharp features during initialization phase and ensure – through the protecting balls – that this discretization appears unchanged in the final mesh. This is achieved as follows. During initialization phase we add one vertex per corners, sample the creases and compute all protecting balls – a set of balls centered either at corners or at crease sample points, and such that:

- The union of the balls entirely covers the sharp features;
- Three balls do not mutually intersect;
- Each ball does not include any other ball center;
- Two balls centered on different creases do not intersect.

The refinement process is then modified to use a weighted Delaunay triangulation. It is started with an initial set of weighted points including weighted points corresponding to the protecting balls: those weighted points are located at the ball centers and their weights are set to the squared radii of the balls. Any other Steiner point later added by the refinement process is given a zero weight. Such weights ensure that each segment joining the centers of two consecutive protecting balls along a sharp crease exists as an edge in the final mesh. Protecting balls guarantee the termination of the Delaunay refinement process whatever may be the angles formed by input surface patches or input creases. Furthermore, the input surface S is never probed by the refinement process inside the union of protecting balls, which is an advantage in cases where input data are noisy around sharp creases.

Figures 6 and 11 depict examples of CAD models meshed without and with protecting the sharp features. Note that in case of domains with sharp features, the domain oracle must be

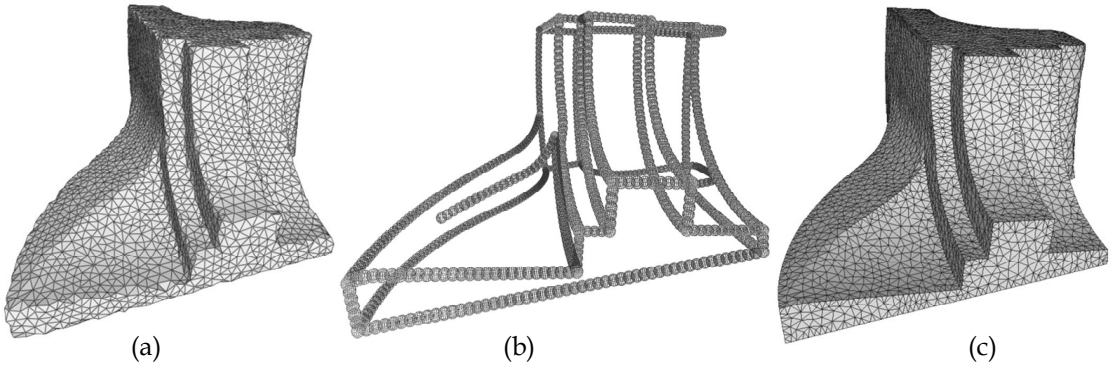


Figure 6: Sharp features on a CAD model: (a) Meshing without features. (b) Protecting balls. (c) Meshing with features.

enriched with the description of these features. More specifically, the oracle must provide the list of sharp corners and capabilities to sample sharp creases by computing a sample point on a sharp crease at a given distance from another sample point. The oracle must also provide topological information relating to the incidence graph of corners, creases and surface patches. Depending on the oracle implementation, these features may be provided by the user, or automatically detected – *e.g.* based on an angle criterion.

2.6 Termination Proof and Guarantees

Termination proof If the minimum facet angle criterion is set to be $\leq 30^\circ$ and the maximum cell radius-edge-ratio criterion is set to be ≥ 2, then the above algorithm is shown to terminate after inserting a finite number of Steiner vertices [22, 49, 54].

Guarantees The resulting mesh includes surface facets and cells meeting all specified criteria about the element size, element shape and boundary approximation error.

Note however that the approximation criterion met by all surface facets does not imply the accuracy of the approximation of S by the surface mesh. Indeed, together with the sizing field, this criterion provides a bound on the one-sided Hausdorff distance from the surface mesh to the input surface but not the other way round.

Further guarantees about the accuracy of the surface approximation may be obtained, relying upon the notion of *medial axis* and *local feature size*. Given an input surface S, the medial axis of S is defined as the locus of points that have more than one closest point on S. The local feature size $lfs(x, S)$ is a sizing field equating for each point x its distance to the medial axis of S. The local feature size is thus related to both the surface curvature – a local property – and the thickness or separation of the domain – a global property.

It has been shown [5, 23] (see also [24]) that, if a point set P is a sampling of the surface S, locally dense enough with respect to the feature size of S,

- $Del_{|S}(P)$ is homeomorphic to S and $Del_{|\Omega}(P)$ is homeomorphic to Ω;
- Hausdorff-distance$(Del_{|S}(P), S) \approx 0$;
- $Del_{|S}(P)$ provides a good estimation of normals, area as well as curvature of S.

Further results [6] show that the Delaunay refinement process described above provides a sampling of the surface that is dense enough with respect to the local feature size, provided that the sizing field input to the refinement process is itself small enough with respect to this local feature size. Unfortunately, in general the local feature size of the surface is unknown. The approximation criterion is a way to adapt locally the sizing of the mesh to the surface curvature but this criterion is insensitive to thickness and separation. From the user point of view a conservative solution is to get an estimation for the minimum thickness/separation of the domain and to specify a uniform sizing function that equates to this estimation.

In addition, the initial point set P must be dense enough to avoid that the mesh refinement process misses completely some small connected components of the surface. The proposed oracles use random probing and/or sampling of sharp features for this purpose, which yields very good results in most cases. Some specific initialization is however required if one needs absolute guarantees that no surface components are missed. One way to provide such a guarantee is the so-called *persistent facet* approach [6]. The latter consists in adding to the initial set of vertices, three points per surface connected component, sufficiently close to construct a surface facet that persists throughout the refinement process. Note however that such a solution may yield small overly refined areas in the final mesh.

2.7 Mesh Optimization

As discussed above, the Delaunay refinement process provides control upon the topological and geometric accuracy of the domain and surfaces approximation, on the size of mesh elements, and on the shape of triangles in the surface approximation. It also provides control upon the shape of mesh tetrahedra through the radius-edge criterion, albeit this is not enough to provide a non-trivial bound on the dihedral angles of tetrahedra. The radius-edge criterion ensures that the output mesh contains no ill-shaped tetrahedra except for one class of them denoted by slivers. Slivers are formed by four vertices close to a circle and roughly equally space along this circle. Slivers are the only class of quasi degenerate tetrahedra that do not have a large radius-edge ratio, which explains why Delaunay refinement is blind to slivers. Figure 7 illustrates the slivers remaining after meshing a sphere. Removing slivers is the main purpose of the mesh optimization phase that we describe next.

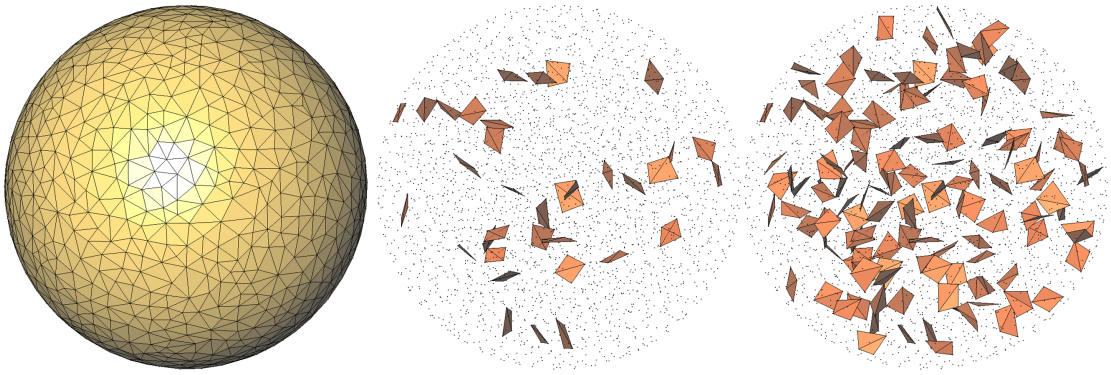


Figure 7: Slivers after Delaunay refinement. The final tetrahedral mesh contains about 38K tetrahedra, some of them being slivers, which are shown for a dihedral angle bound of respectively 5 degrees (middle – 35 slivers) and 10 degrees (right – 136 slivers).

Many mesh quality improvements techniques have been proposed in the literature [31, 35, 44, 51]. CGALmesh offers four different mesh optimization methods which can be combined sequentially. These methods fall into two main categories: global optimization and local optimization. The first category includes Lloyd relaxation [29, 30] and Optimal Delaunay Triangulation (ODT for short) relaxation [14, 1]. The second category includes vertex perturbation [37, 57] and sliver exudation [15].

While global optimizers substantially reduce the total number of slivers, local optimizers focus on the worst tetrahedra. Our experiments show that a Lloyd global optimization followed by vertex perturbation often provides us with the best results when seeking for the largest minimum dihedral angle for the tetrahedra.

Global optimization Global optimizers consider an energy that is a function of the mesh vertices and connectivity, and aim at minimizing this energy by relocating all mesh vertices at once, before updating the mesh connectivity.

Both Lloyd and ODT smoothers use an energy defined as the L_1-norm of the approximation error between the isotropic paraboloid function $f(x) = x^2$ (whose graph is in $\mathbb{R}^4$) and a piecewise linear interpolation of this function on the domain.

In the Lloyd smoother, the linear interpolation f_{PWL}^{dual} is defined on each Voronoi cell as the linear expansion of $f(x)$ from its value at the Voronoi site. Consider the unit isotropic paraboloid $y = x^2$ of $\mathbb{R}^4$, the interpolating points (x_i, x_i^2) obtained by lifting the mesh vertices on the paraboloid, and the polytope P_{Lloyd} defined as the intersection of the half-spaces above the hyperplans tangent to the paraboloid at the lifted mesh vertices. The L_1-norm of the approximation error, $\|f - f_{PWL}^{dual}\|_{L_1}$, measures the volume enclosed between the paraboloid and the boundary of the polytope P_{Lloyd} (Figure 8).

In the ODT smoother, the linear interpolation f_{PWL}^{primal} is defined as the linear interpolation of $f(x)$ on each mesh tetrahedron. The L_1-norm of the approximation error, $\|f_{PWL}^{primal} - f\|_{L_1}$, measures the volume enclosed between the unit paraboloid of $\mathbb{R}^4$ and the lower boundary of the convex hull of the lifted vertices (Figure 8).

Note that after Delaunay refinement the final mesh may have a non-uniform density of vertices, reflecting a non-uniform sizing field that is the pointwise minimum between a (possibly non-uniform) user-defined sizing field and the local feature size of the meshed domain. To preserve this non-uniform density throughout the optimization process, the Lloyd and ODT energy integrals are computed using a weighted version of the error, where the weights are locally estimated from the average length of edges incident to each vertex of the mesh after refinement. For both optimizers, at each optimization step, closed form formulas provide the new location of the mesh vertices as a function of the current mesh vertices and connectivity [30, 1]. Each optimization step computes the new position of all mesh vertices, relocates them and updates the Delaunay triangulation as well as both restricted Delaunay triangulations.

The user is provided control upon the termination of the optimization process through the following parameters:

- A time limit;
- A maximum number of optimization steps;
- A "convergence" ratio r. The optimization process stops when, during the last iteration, the displacement of any vertex v is lower than a given fraction r of the length of the shortest edge incident to v.

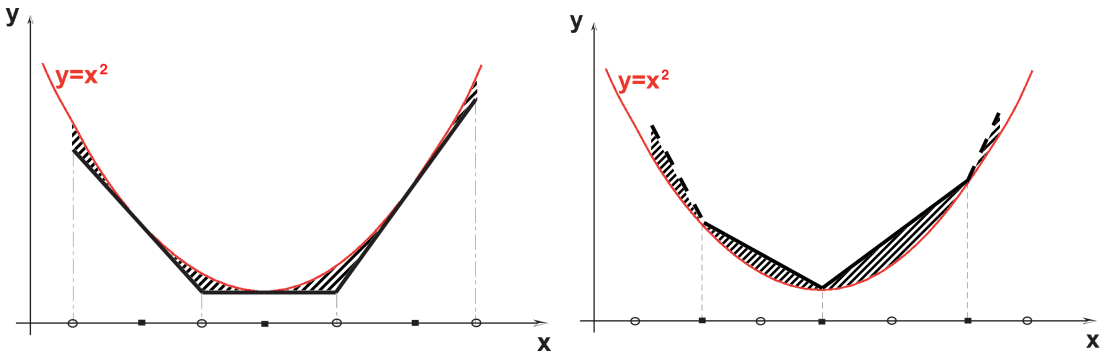


Figure 8: 1D depiction of the mesh energy minimized by Lloyd (left) and ODT (right) smoothers. Black dots on the x axis represent the mesh vertices. White circles represent Voronoi vertices. The energy to be minimized is the volume measure of the shaded region.

Local optimization The vertex perturber and the sliver exuder act locally in the sense that they first parse the mesh in order to find the worst elements and process them individually. The focus is on improving the worst elements first.

The vertex perturber [37, 57] improves the mesh by relocating the vertices in turn, starting with vertices incident to the worst sliver tetrahedra. Each vertex relocation is followed by updating the mesh connectivity to restore the Delaunay and restricted Delaunay properties of the mesh. For each vertex, the algorithm searches for a new location that yields an improvement of the vertex star. More specifically, the new vertex location is not computed so as to improve the shape of the incident sliver, but instead so as to make this sliver disappear when the Delaunay mesh connectivity is restored. The new vertex location is thus first searched so as to increase the incident sliver circumradius. If this fails to yield a change in the mesh connectivity which improves the star, another vertex location is searched so as to invert the worst incident tetrahedron. If this again fails, random perturbations of the vertex location are attempted.

The sliver exuder [15] turns the Delaunay triangulation into a weighted Delaunay triangulation. It aims at removing slivers through re-weighting the mesh vertices with optimal weights that trigger combinatorial changes inducing sliver exudation. As for the global optimizers, the user is provided control upon the termination of local optimization processes, through the following parameters:

- A time limit;
- A lower bound on dihedral angles of the mesh tetrahedra.

3 Implementation Details and Software Design

CGALmesh [2] is the 3D Mesh Generation package of the CGAL library [13], which is available on the CGAL website [12]. For Linux systems, CGAL is part of the package repositories of Arch Linux, Debian-based distributions (Ubuntu, Mint, KNOPPIX, etc.) and distributions using the RPM package manager.

CGALmesh has been carefully designed and implemented in order to achieve both reliability and flexibility. We now go through the main design choices made during the implementation process.

3.1 Generic Programming

CGAL is a C++ library, designed to be highly generic through the use of templates. Each CGAL package provides the users with the ability to use their own data types for elementary geometric objects as long as they provide the required attributes types and functions. Being part of CGAL, CGALmesh follows such generic design and provides an additional level of genericness through the use of an abstract oracle to interface with the input domain representation. This oracle encapsulates the required domain knowledge. Supporting a novel type of domain representation boils down to implementing a new oracle. The current software provides oracles for polyhedral domains, implicit surfaces and 3D labeled images.

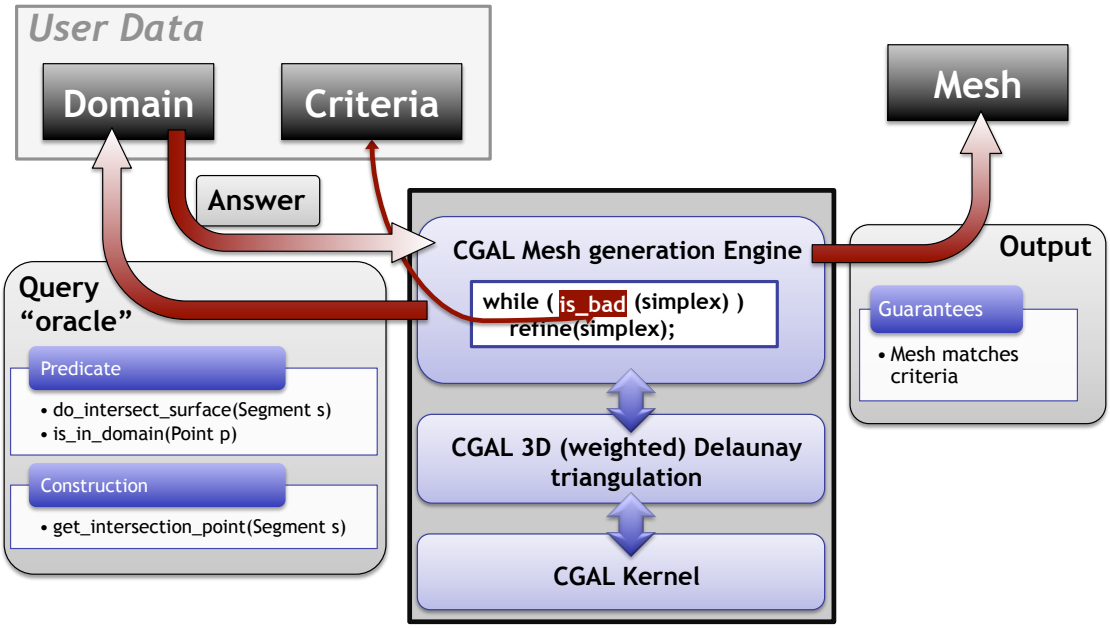


Figure 9: Overall design of the mesh generation process.

Figure 9 depicts the overall modular design of the mesh generation algorithm and the interaction between the different modular components. The core mesh generation engine is described in a dedicated publication [47].

3.2 Oracles

CGALmesh already provides oracles for domains described through polyhedral meshes, implicit functions and 3D images.

The polyhedral mesh oracle takes as input a triangle surface mesh. To provide efficient surface intersections during refinement a hierarchical data structure of axis-aligned bounding boxes (AABB tree) is first created [4]. The oracle for implicit functions uses the concept of bipolar Voronoi edges to detect intersections. A Voronoi edge is said to be bipolar if the implicit function values have opposite signs at its endpoints [6]. Only bipolar Voronoi edges are considered to intersect the surface and the intersection points are obtained through dichotomy, *i.e.*, by recursively subdividing the edge into two equal edges, keeping the one whose endpoints have opposite signs, until reaching a specified precision. The 3D image oracle computes implicit functions from input 3D images, and relies on the so-constructed implicit function oracle for predicates and constructions [45].

The complexity of the predicates and constructor required in the oracle greatly varies among the different models for the oracle. For the case of implicit function and 3D images, testing for intersection boils down to evaluating the function at two locations, whereas the polyhedral oracle requires a compute-intensive tree search. Note however that computing the intersection by dichotomy is more complex than computing the intersection point between a Voronoi edge and a triangle in the leaf of the AABB tree.

The flexibility offered by the oracle design makes CGALmesh particularly well suited to applications where the input domain is either known through measurements [50, 40, 3] or described with complex application-specific data structures such as composed implicit functions in geophysics. As the input domain is queried only at meshing time in a coarse-to-fine manner the CGALmesh algorithms are also favored in constrained-memory applications, where complex data processing such as robust denoising can be delegated to the oracle [11]. We also performed prototype experiments with oracles designed to mesh smooth subdivision surfaces and smooth parametric NURBS surfaces, where the oracles embed complex numerical procedures to compute intersection points.

3.3 Data Structures

The CGALmesh package primarily relies on the CGAL 3D Delaunay triangulation package. The latter provides fast and reliable algorithms to create, modify and access the Delaunay triangulation of a set of 3D points. The meshing algorithm maintains the restricted Delaunay triangulations to surfaces and volumes which are respectively a 2D and a 3D complex embedded in the 3D Delaunay triangulation of the vertices. Two additional data structures are thus used to represent 2D and 3D complexes embedded in a 3D triangulation.

4 Meshing Experiments

We now provide a series of experimental results on a variety of input domains (Figure 10). Timings are measured on a PC running Windows 7 with an Intel Xeon 2.4 GHz CPU and 128GB RAM, using CGAL 4.4, 64-bit version. Note that CGAL – which includes CGALmesh – is also available for Linux, MacOS X and Solaris; it can be downloaded on the CGAL website [12].

4.1 Mesh Refinement

Table 4.1 and Figure 12 focus on the refinement step of the mesh generation process, using the three provided oracles: polyhedral surface meshes, implicit functions and 3D images. Timings for the facet scan process, negligible, are not shown. The first set of input domains contains polyhedral surfaces that are piecewise linear approximations of smooth surfaces. The second set contains CAD polyhedral surfaces with sharp features. The third set contains domains defined by implicit functions. The fourth set contains 3D images. Figure 12 shows that most domains exhibit nearly similar refinement speed, except for the *cheese* and the *ecrou-larousse* models which are slower due to a large number of sharp creases. Note that the *cheese* model, containing many sharp creases incident to small angles, is correctly meshed, albeit slowly, by the meshing process (Figure 11). These results detail the volume and surface meshing performances.

Table 4.1 compares CGALmesh with LocPSC [28], a Delaunay-based surface mesh refinement algorithm based on CGAL triangulations. By not specifying any tetrahedron-related criteria, we use CGALmesh to produce a triangular mesh of similar size and quality than the ones produced by LocPSC. Those results show that the main strength of CGAL is its processing times, while LocPSC consumes less memory through a localized Delaunay refinement process. We observe that CGALmesh is slightly more parsimonious.

Regarding tetrahedral mesh generation, most available tools, such as TETGEN [56], require a triangle surface mesh as input, and the quality of the generated tetrahedral mesh directly relates to the quality of the input. Although comparing two different kinds of methods is difficult, we provide through table 4.1 a comparison between the volume refinement step of CGALmesh to the mesh generation process of TETGEN. CGALmesh is not as fast as TETGEN, but yields honorable speed when considering the fact that it maintains a Delaunay triangulation of the surface while refining the volume.

4.2 Mesh Optimization

Figures 13, 14 and 15 illustrate how combining global and local optimization helps improving the distribution of tetrahedron dihedral angles. The global optimization step is effective at concentrating the distribution, but leaves a few extreme dihedral angles. The local optimization process removes most extreme angles while leaving other angles unchanged. On the smooth Bimba model the Lloyd optimization step improves the whole distribution of angles. On the CAD models the Lloyd optimization step improves mostly near curved areas, as the elements on flat areas are already well shaped straight after refinement. On CAD models the protecting balls prevent the vertices on the sharp creases to be relocated during optimization. The local optimization is thus not as effective as for smooth models, and a few slivers remain.

Table 1: Timings for mesh refinement – domains are depicted by Figure 10

Domain (#tri)	**#V**[1]	**#F**[1]	**#T**[1]	**S.R.**[2] **(s)**	**C.S.**[3] **(s)**	**V.R.**[4] **(s)**	**Total (s)**	**#V/s**[5]	**Mem(MB)**
Elephant (5k)	536k	126k	3,218k	8.8	0.9	36.4	46.1	11,622	1,187
	4,152k	507k	25,709k	35.5	3.5	352.0	391.1	10,615	8,638
	9,764k	903k	60,927k	64.0	6.9	782.4	853.3	11,443	19,997
Bimba (384k)	965k	139k	5,941k	7.8	1.3	53.5	62.6	15,426	2,235
	4,400k	387k	27,486k	22.1	3.7	277.3	303.0	14,520	9,505
	14,736k	874k	92,734k	49.9	8.6	1067.3	1125.8	13,089	30,410
lucy (100k)	566k	179k	3,327k	14.0	1.9	31.6	47.5	11,923	1,275
	4,343k	723k	26,585k	56.2	8.5	289.1	353.7	12,277	9,159
	28,042k	2,550k	175,063k	196.2	30.7	2192.9	2419.8	11,588	57,625
david (200k)	731k	198k	4,350k	12.7	1.9	41.3	55.9	13,063	1,668
	3,288k	554k	20,119k	35.6	5.6	211.1	252.3	13,031	6,916
	10,936k	1,250k	67,866k	80.3	13.0	799.1	892.4	12,255	22,701
ecrou-larousse (27k)	836k	346k	4,760k	59.9	8.1	45.5	113.5	7,363	1,531
	3,581k	799k	21,584k	128.3	16.5	226.5	371.3	9,645	6,418
	11,784k	1,703k	72,553k	271.3	33.4	912.6	1217.3	9,680	20,659
pump_carter (30k)	600k	197k	3,519k	12.5	1.3	30.2	44.0	13,645	1,132
	4,578k	780k	28,002k	48.0	5.5	278.4	331.9	13,792	8,093
turbine (18k)	449k	201k	2,544k	19.3	2.0	24.1	45.4	9,885	884
	3,367k	788k	20,243k	71.6	8.0	202.8	282.5	11,919	6,001
cheese (18k)	1,025k	463k	5,797k	92.8	9.7	65.8	168.3	6,092	1,952
	5,179k	1,396k	30,836k	268.6	28.5	363.7	660.7	7,838	9,097
	39,851k	5,505k	245,751k	1053.6	117.2	3657.9	4828.7	8,253	69,635
Thin_cylinder_fct	316k	406k	1,149k	16.1	0.7	3.4	20.3	15,569	655
	1,585k	1,665k	6,607k	68.4	3.3	29.2	100.8	15,722	3,187
	6,115k	6,607k	24,888k	343.5	13.7	143.8	500.9	12,207	12,267
Klein_function	754k	639k	3,734k	24.5	1.4	33.9	59.8	12,608	1,554
	3,062k	2,014k	16,226k	81.1	4.4	147.5	233.0	13,142	6,320
	21,721k	9,817k	122,315k	394.8	26.6	1871.6	2292.9	9,473	44,164
Pancake_function	485k	881k	1,645k	34.1	1.9	6.6	42.7	11,371	1,002
	4,402k	6,676k	15,896k	290.4	14.6	76.9	382.0	11,523	8,781
Tanglecube_function	389k	482k	1,664k	18.4	1.0	14.8	34.2	11,379	818
	1,478k	1,522k	6,905k	60.1	3.2	59.0	122.3	12,085	3,091
liver_kidney_gallbladder	1,289k	215k	7,915k	10.6	0.5	65.0	76.2	16,934	2,722
	4,630k	525k	28,781k	26.3	1.5	262.4	290.1	15,957	9,623
	19,666k	1,459k	123,437k	74.0	4.7	1319.0	1397.7	14,071	40,459
VisibleHuman 1mm	345k	715k	2,055k	25.3	1.3	17.6	44.2	7,811	1,116
	3,200k	4,328k	19,795k	155.5	10.0	204.9	370.4	8,640	7,036
	10,448k	10,812k	65,426k	424.6	31.1	725.9	1181.6	8,842	21,859

[1] #V = Number of vertices / #F = Number of triangles / #T = Number of tetrahedra
[2] S.R. = Surface mesh refinement
[3] C.S. = Cell scan
[4] V.R. = Volume mesh refinement
[5] #V/s = Number of vertices created per second

Table 2: Refinement results for surface mesh generation, compared to the LocPSC software.

Model	**Algorithm**	**Criteria**	**#V**[1]	**#F**[1]	**Time (s)**	**Mem (MB)**
Fandisk	LocPSC	$\lambda = 0.0035$	468,652	937,300	309	129
	CGAL 4.4	FS[2] = 0.001788	455,649	911,262	63	893
Rocker	LocPSC	$\lambda = 0.0045$	468,543	937,086	757	137
	CGAL 4.4	FS[2] = 0.001365	459,674	919,318	79	1,000
3 Holes	LocPSC	$\lambda = 0.0011$ & AT[3] = 0.01	8,165,426	16,330,860	9,470	2,138
	CGAL 4.4	FS[2] = 0.03764	8,042,469	16,084,914	1,268	15,976
Lucy	LocPSC	$\lambda = 0.004$ & AT[3] = 0.01	371,656	743,308	309	143
	CGAL 4.4	FS[2] = 0.001341	365,579	731,110	87	864
Fertility	LocPSC	$\lambda = 0.0035$ & AT[3] = 0.01	611,537	1,223,086	606	338
	CGAL 4.4	FS[2] = 0.2568	603,847	1,207,658	111	1,553

[1] #V = Number of vertices / #F = Number of triangles
[2] FS = Facet size
[3] AT = Angle threshold

Table 3: Refinement results for volume mesh generation, compared to the Tetgen software.

Model	**Algorithm**	**#V**[1]	**#F**[1]	**#T**[1]	**Time (s)**
Fandisk	Tetgen	42,654	16,194	247,376	1.79
	CGAL 4.4	43,171	19,178	244,542	C.S.[2] = 0.02 & V.R.[3] = 3.4
Lucy	Tetgen	72,157	144,310	248,081	1.88
	CGAL 4.4	78,415	145,022	245,536	C.S.[2] = 2.5 & V.R.[3] = 1.5
David	Tetgen	100,979	201,974	335,144	2.71
	CGAL 4.4	111,103	203,730	335,834	C.S.[2] = 3.1 & V.R.[3] = 4.9

[1] #V = Number of vertices / #F = Number of triangles / #T = Number of tetrahedra
[2] C.S. = Cell scan
[3] V.R. = Volume mesh refinement

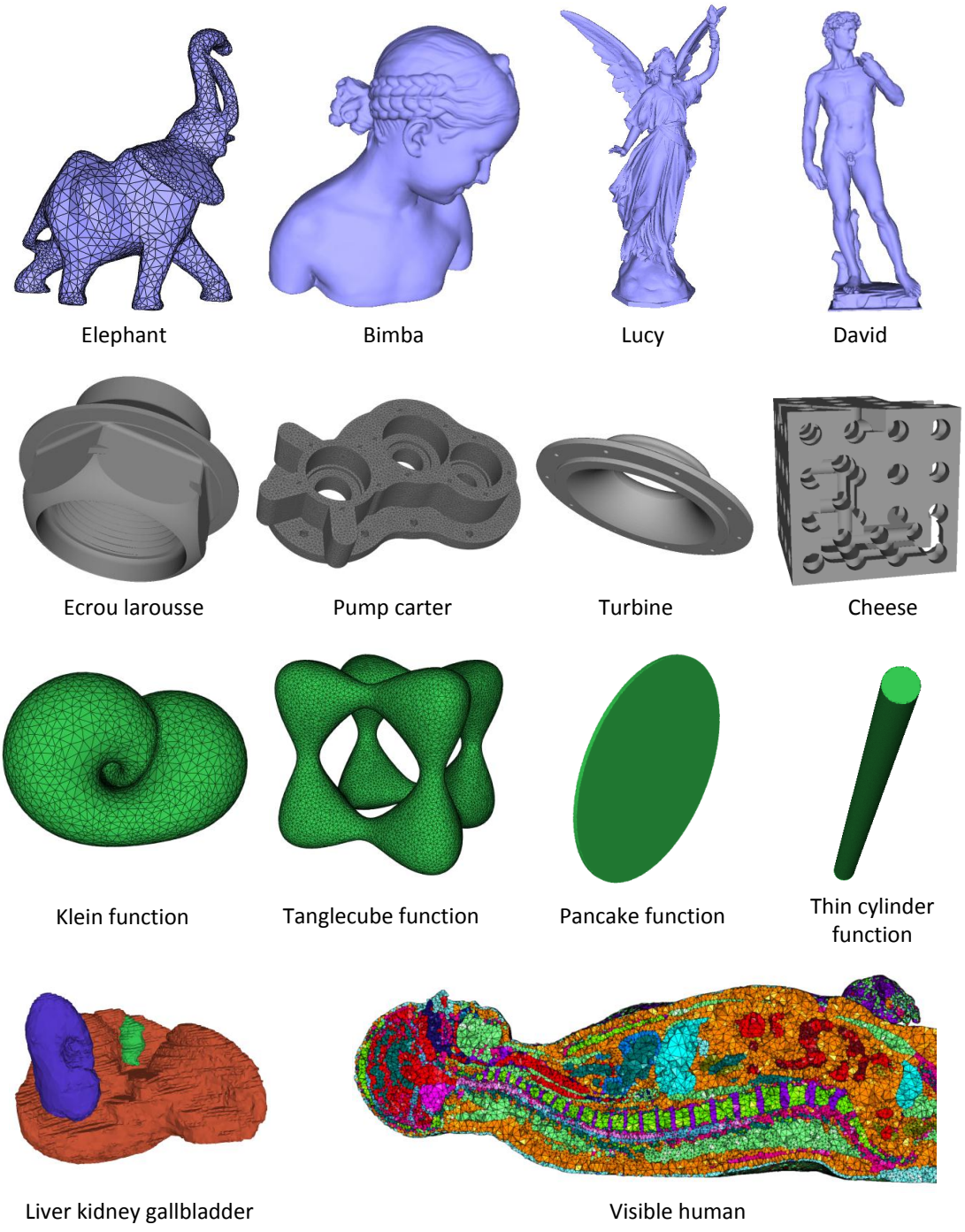


Figure 10: Input domains: domains bounded by smooth surfaces (top row, in blue), CAD models with sharp features (second row, in grey), implicit functions (third row, in green), 3D images (bottom row, multicolored). All non implicit models are available from the VISIONAIR shape repository, except for the Visible human model which is available from the Visible Human Project (http://www.nlm.nih.gov/research/visible/getting_data.html).

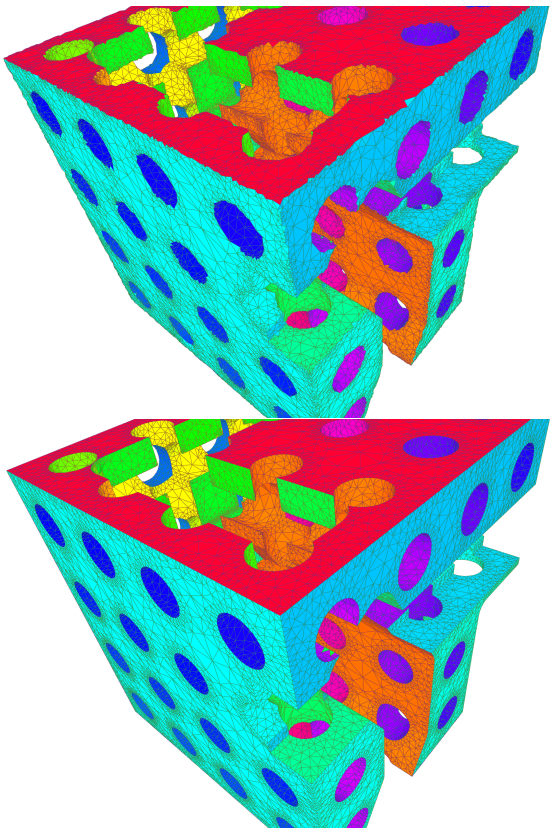


Figure 11: Cheese model. (left) Meshing without sharp features. (right) Meshing with sharp features. Input domain courtesy Distene.

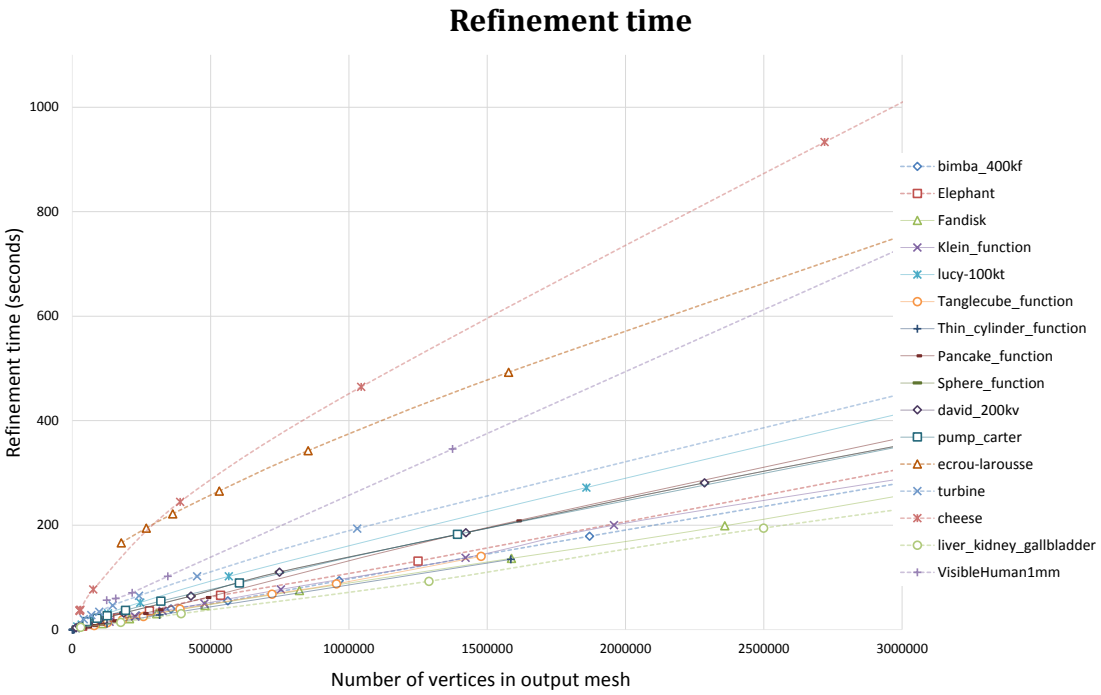


Figure 12: Refinement times with respect to the complexity of the output mesh.

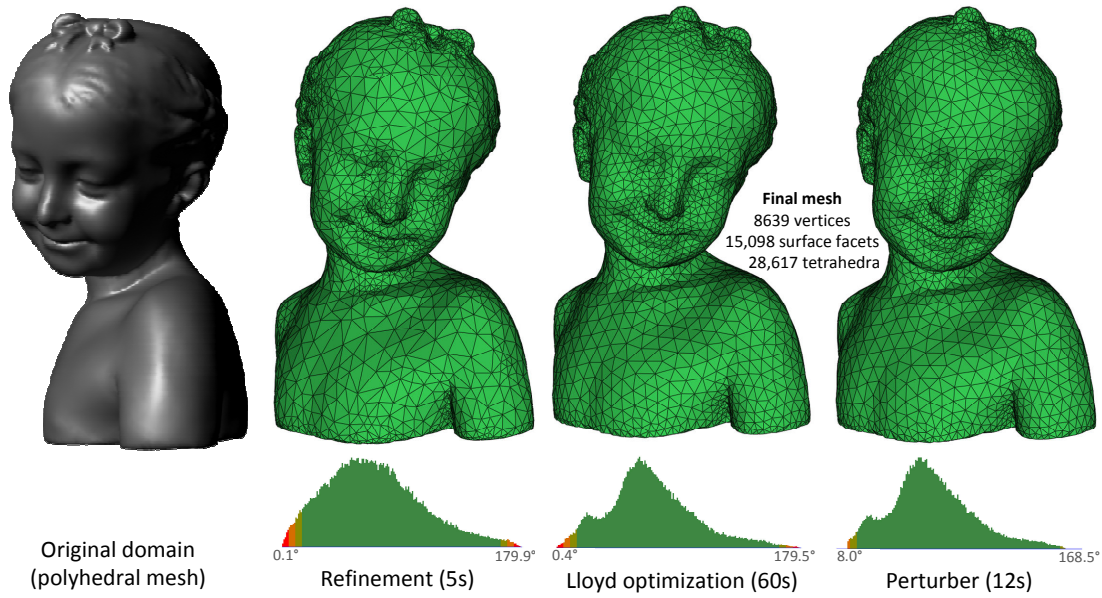


Figure 13: Mesh refinement and optimization on the Bimba model. The distribution of dihedral angles, with both min and max values, is depicted under each mesh.

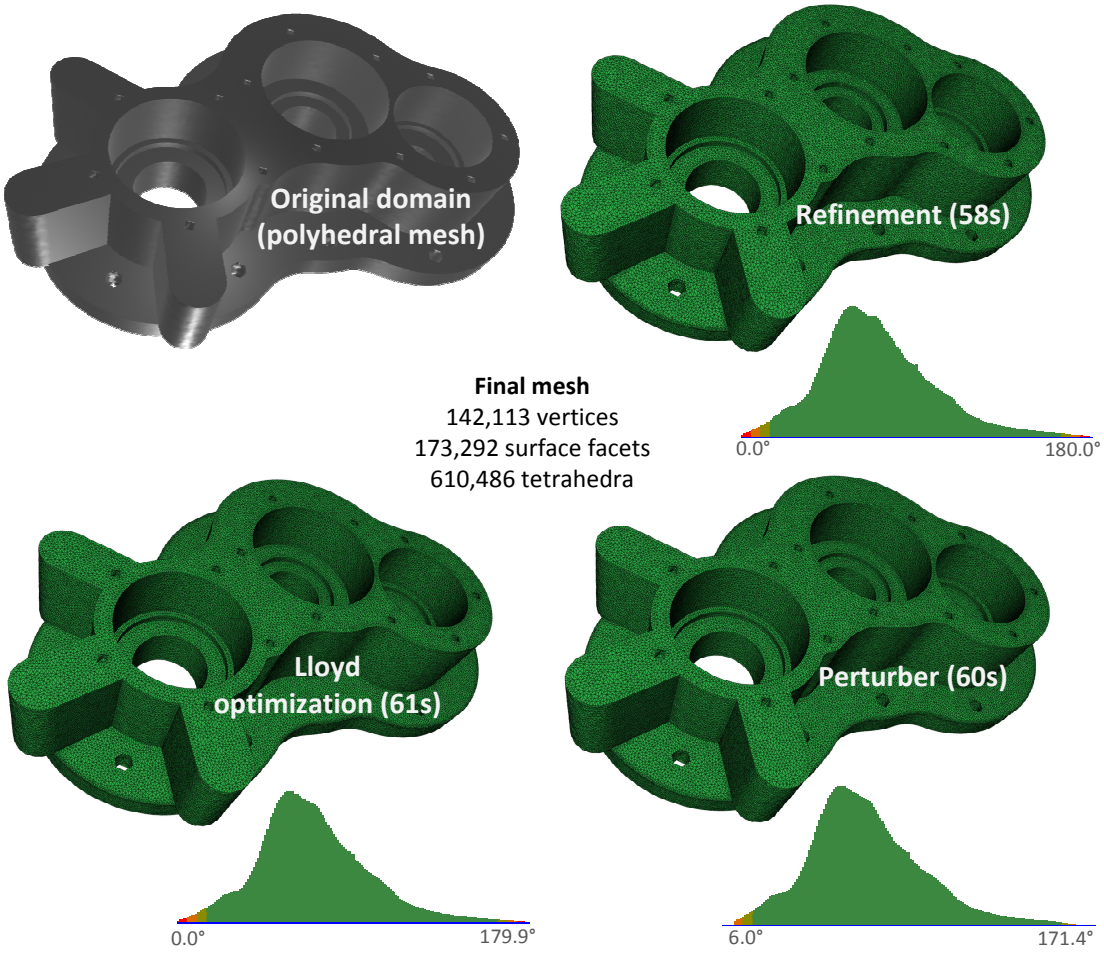


Figure 14: Mesh refinement and optimization of the Pump_carter model. The distribution of dihedral angles, with both min and max values, is depicted under each mesh.

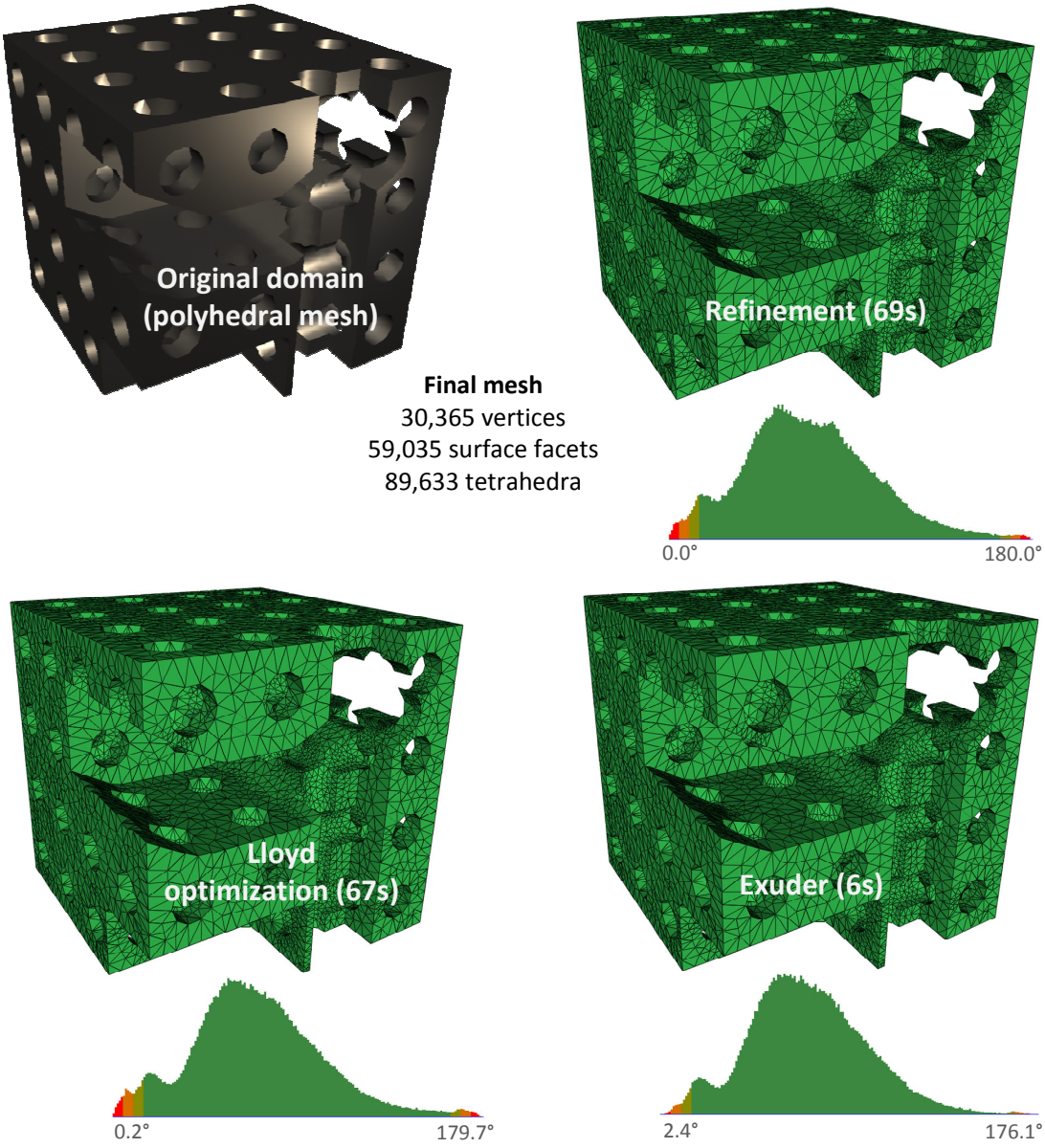


Figure 15: Mesh refinement and optimization of the Cheese model. The distribution of dihedral angles – not taking into account dihedral angles at input creases –, with both min and max values, is depicted under each mesh.

5 Conclusion

CGALmesh is the 3D mesh generation package provided by the CGAL library. It offers algorithms to generate isotropic surface triangular meshes and 3D tetrahedral meshes. The user is provided with control over the output meshes through several criteria, and four optional optimization algorithms are used to trade mesh quality for time. The Delaunay refinement process embedded in the main mesh generation algorithm comes with several theoretical and practical guarantees which translate from the end user point of view into reliability and conservative matching of the user-specified criteria.

The input domain is queried in an abstract and generic manner through *oracles*. This results in a versatile computational software tool which can be used, among other applications, for meshing CAD models – preserving sharp edges and creases –, extracting isosurfaces, remeshing polyhedral surfaces or meshing from 3D segmented images[1].

In the near future we plan to release parallel versions of the proposed algorithms so as to take advantage of modern multi-core processor architectures. As already shown [20] mesh generation is far from being an embarrassingly parallel problem.

Another stimulating work for CGALmesh is the generation of anisotropic simplicial surface and volume meshes [8, 7], which will be integrated into CGAL in the near future.

6 Acknowledgments

This work was funded by Inria Technology Transfer and Innovation, and by the European Research Council (ERC Starting Grant "Robust Geometry Processing", Grant agreement 257474).

References

[1] P. Alliez, D. Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. Variational Tetrahedral Meshing. *ACM Transactions on Graphics (SIGGRAPH)*, 24:617–625, 2005.

[2] Pierre Alliez, Laurent Rineau, Stéphane Tayeb, Jane Tournois, and Mariette Yvinec. 3D mesh generation. In CGAL user and reference manual. CGAL Editorial Board, 4.2 edition, 2013. http://www.cgal.org/Manual/latest/doc_html/cgal_manual/packages.html#Pkg:Mesh_3.

[3] Pierre Alliez, Laurent Saboret, and Gael Guennebaud. Surface reconstruction from point sets. In CGAL user and reference manual. CGAL Editorial Board, 4.2 edition.

[4] Pierre Alliez, Stéphane Tayeb, and Camille Wormser. 3D fast intersection and distance computation (AABB tree). In CGAL user and reference manual. CGAL Editorial Board, 4.2 edition, 2013. http://www.cgal.org/Manual/latest/doc_html/cgal_manual/packages.html#Pkg:AABB_tree.

[5] Nina Amenta and Marshall Bern. Surface reconstruction by Voronoi filtering. In *Proc. 14th Annu. Sympos. Comput. Geom.*, pages 39–48, 1998.

[6] J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67:405–451, 2005.

[1] Projects using CGAL: http://www.cgal.org/projects.html

[7] J.-D. Boissonnat, Kan-Le Shi, Jane Tournois, and Mariette Yvinec. Anisotropic Delaunay meshes of surfaces. *ACM Transactions on Graphics*, 2012.

[8] J.-D. Boissonnat, C. Wormser, and M. Yvinec. Locally uniform anisotropic meshing. In *Proc. 24th Annual Symp. on Computational Geometry*, pages 270–277, 2008.

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

[10] Dobrina Boltcheva, Mariette Yvinec, and Jean-Daniel Boissonnat. Mesh generation from 3D multi-material images. In *Medical Image Computing and Computer-Assisted Intervention*, volume 5762 of *Lecture Notes in Computer Science*, pages 283–290, 2009.

[11] Ricard Campos, Rafael Garcia, Pierre Alliez, and Mariette Yvinec. Splats-based surface reconstruction from defect-laden point sets. *Graphical Models*, 2013. to appear.

[12] CGAL Editorial Board. Downloading cgal. http://www.cgal.org/download.html.

[13] CGAL Editorial Board. Cgal, Computational Geometry Algorithms Library. http://www.cgal.org.

[14] Long Chen and Jinchao Xu. Optimal Delaunay triangulations. *J. Comput. Math.*, 22(2):299–308, 2004.

[15] Siu-Wing Cheng, Tamal K. Dey, Herbert Edelsbrunner, Michael A. Facello, and Shang-Hua Teng. Sliver exudation. *J. ACM*, 47:883–904, 2000.

[16] Siu-Wing Cheng, Tamal K. Dey, and Edgar A. Ramos. Delaunay refinement for piecewise smooth complexes. In *Proc. of the 18th annual ACM-SIAM Symp. on Discrete Algorithms*, pages 1096–1105, 2007.

[17] Siu-Wing Cheng, Tamal K. Dey, and Jonathan R. Shewchuk. *Delaunay Mesh Generation.* CRC Press, 2012.

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

[19] Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. O'Brien, and Jonathan R. Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation 2007*, pages 219–228, 2007.

[20] Andrey N. Chernikov and Nikos P. Chrisochoides. A template for developing next generation parallel Delaunay refinement methods. *Finite Elements in Analysis and Design*, 46(1-2):96 – 113, 2010. Mesh Generation - Applications and Adaptation.

[21] L. P. Chew. Guaranteed-quality triangular meshes. Technical Report TR-89-983, Dept. Comput. Sci., Cornell Univ., Ithaca, NY, April 1989.

[22] L.P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *SCG '93: Proc. of the 9th Annual Symp. on Computational Geometry*, pages 274–280. ACM New York, NY, USA, 1993.

[23] David Cohen-Steiner and Jean-Marie Morvan. Restricted Delaunay triangulations and normal cycle. In *Proc. 19th Annual Symposium on Computational Geometry*, pages 237–246, 2003.

[24] Tamal K Dey. *Curve and surface reconstruction: algorithms with mathematical analysis*, volume 23. Cambridge University Press, 2007.

[25] Tamal K Dey. Delaunay mesh generation of three dimensional domains. *Tessellations in the sciences: Virtues, Techniques and Applications of Geometric Tilings*, 2007.

[26] Tamal K. Dey, Firdaus Janoos, and Joshua A. Levine. Meshing interfaces of multi-label data with Delaunay refinement. *Engineering with Computers*, 28(1):71–82, January 2012.

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

[28] Tamal K. Dey, Joshua A. Levine, and A. Slatton. Localized Delaunay refinement for sampling and meshing. *Computer Graphics Forum (Special Issue of Eurographics SGP)*, 29(5):1723–1732, July 2010.

[29] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi tesselations : Applications and algorithms. *SIAM Review*, 41:637–676, 1999.

[30] Qiang Du and Desheng Wang. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *International Journal for Numerical Methods in Engineering*, 56(9):1355–1373, 2003.

[31] Lori A. Freitag and Paul Plassmann. Local optimization-based simplicial mesh untangling and improvement. *International Journal for Numerical Methods in Engineering*, 49(1-2):109–125, 2000.

[32] Pascal Jean Frey and Paul-Louis George. *Mesh Generation: Application to Finite Elements*. ISTE, 2007.

[33] P.-L. George. *Tetmesh-GHS3D, Tetrahedral Mesh Generator*. INRIA - Simulog-Technologies, 2005.

[34] Serge Gosselin and Carl Ollivier-Gooch. Constructing constrained Delaunay tetrahedralizations of volumes bounded by piecewise smooth surfaces. *International Journal of Computational Geometry & Applications*, 21(05):571–594, 2011.

[35] Bryan Matthew Klingner and Jonathan Richard Shewchuk. Aggressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, pages 3–23, 2008.

[36] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph.*, 26(3):57.1–57.10, July 2007.

[37] X. Y. Li. Spacing control and sliver-free Delaunay mesh. In *Proc. 9th Int. Meshing Round Table*, 2000.

[38] William E. Lorensen and Harvey E. Cline. Seminal graphics. chapter Marching cubes: a high resolution 3D surface construction algorithm, pages 347–353. ACM, New York, NY, USA, 1998.

[39] David L. Marcum. Advancing-front/local-reconnection (aflr) unstructured grid generation. *Computational Fluid Dynamics Review*, 1:140–157, 1998.

[40] Patrick Mullen, Fernando de Goes, Mathieu Desbrun, David Cohen-Steiner, and Pierre Alliez. Signing the unsigned: Robust surface reconstruction from raw pointsets. *Computer Graphics Forum*, 29:1733–1741, 2010. Special issue 7th Annu. Sympos. Geometry Processing.

[41] Démian Nave, Nikos Chrisochoides, and L. Paul Chew. Guaranteed-quality parallel Delaunay refinement for restricted polyhedral domains. *Comput. Geom. Theory Appl.*, 28(2-3):191–215, June 2004.

[42] C. Ollivier-Gooch. *GRUMMP User Guide*. Department of Mechanical Engineering, University of British Columbia, 2010.

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

[44] Jeonghyung Park and Suzanne M. Shontz. Two derivative-free optimization algorithms for mesh quality improvement. *Procedia Computer Science*, 1(1):387 – 396, 2010. ICCS 2010.

[45] J.-P. Pons, F. Ségonne, J.-D. Boissonnat, L. Rineau, M. Yvinec, and R. Keriven. High-quality consistent meshing of multi-label datasets. In *Information Processing in Medical Imaging*, pages 198–210, 2007.

[46] Jean-Philippe Pons and Jean-Daniel Boissonnat. Delaunay deformable models: Topology-adaptive meshes based on the restricted Delaunay triangulation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.

[47] L. Rineau and M. Yvinec. A generic software design for Delaunay refinement meshing. *Comput. Geom. Theory Appl.*, 38:100–110, 2007.

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

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

[50] Nader Salman, Mariette Yvinec, and Quentin Mérigot. Feature preserving mesh generation from 3D point clouds. *Computer Graphics Forum*, 29:1623–1632, 2010. Special issue for EUROGRAPHICS Symp. on Geometry Processing.

[51] ShankarP. Sastry, SuzanneM. Shontz, and StephenA. Vavasis. A log-barrier method for mesh quality improvement and untangling. *Engineering with Computers*, pages 1–15, 2012.

[52] Joachim Schöberl. NETGEN an advancing front 2D/3D-mesh generator based on abstract rules. *Computing and Visualization in Science*, 1:41–52, 1997.

[53] Joachim Schöberl. *NETGEN - 4.3*. Johannes Kepler University Linz, 2003.

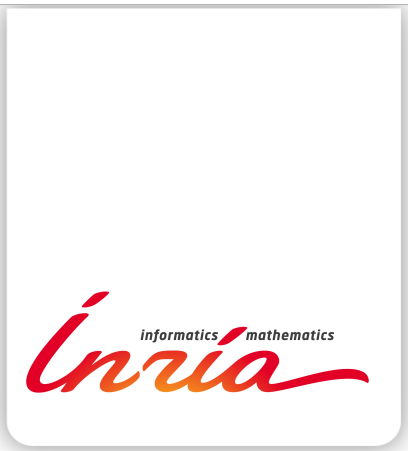
[54] J.R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Proc. of the Fourteenth Annual Symp. on Comp. Geometry*, pages 86–95, 1998.

[55] J.R. Shewchuk. *Combinatorial Scientific Computing*, chapter 10: Unstructured Mesh Generation, pages 257–297. CRC Press, 2012.

[56] H. Si. *TetGen A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator*. Weierstrass Institute for Applied Analysis and Stochastics (WIAS), Mohrenstr. 39, 10117 Berlin, Germany, 2006.

[57] Jane Tournois, Camille Wormser, Pierre Alliez, and Mathieu Desbrun. Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Transactions on Graphics*, 28(3):75:1–75:9, 2009. SIGGRAPH '2009 Conference Proc.

[58] Stephen A. Vavasis. QMG 2.0 reference manual, 1999.



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399