

CGAL - The Computational Geometry Algorithms Library

Andreas Fabri, Monique Teillaud

▶ **To cite this version:**

Andreas Fabri, Monique Teillaud. CGAL - The Computational Geometry Algorithms Library. 10e colloque national en calcul des structures, May 2011, Giens, France. pp.6. hal-00592685

HAL Id: hal-00592685
https://hal.science/hal-00592685

Submitted on 3 May 2011

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.

CSMA 2011
10e Colloque National en Calcul des Structures
9-13 Mai 2011, Presqu'île de Giens (Var)

CGAL - The Computational Geometry Algorithms Library

A. Fabri[1], M. Teillaud[2]

[1] GEOMETRYFACTORY, *France, Andreas.Fabri@geometryfactory.com*
[2] *INRIA Sophia Antipolis - Méditerranée, France, Monique.Teillaud@inria.fr*

Résumé — We present fundamental geometric data structures and algorithms offered by CGAL, the Computational Geometry Algorithms Library. As geometry is ubiquitous this library is used by application developers in a variety fileds such as medical imaging, VLSI, CAD/CAM, geophysics, computer graphics, GIS... In the presention we will give a broad overview, and go in more detail on surface and volume mesh generation, as well as on geometry for periodic domains.
Mots clés — geometric data structures and algorithms, open source, C++, generic programming, exact geometric computing

1 Introduction

CGAL, the Computational Geometry Algorithms Library, is a C++ library of geometric algorithms and data structures which is developed by the CGAL Open Source project [1]. This project started in 1996 as a joint effort of several research groups working in computational geometry, and it has been partially funded by the European and national science foundations. CGAL consists of 600K lines of code, organized in 80 packages, documented in a 3500 pages user and reference manual, with 1000 subscribers to the user forum, and about 100 commercial users, about 20 active developers, and two types of licenses : open source and commercial. The project is steered by an Editorial Board, which is also responsible for reviewing submissions of new packages.

2 Algorithms

As CGAL is not an end user application, but a collection of geometric software components, the demos given during the presentation serve as illustrations of the algorithms and they are not specific for a particular application domain. The goal of the demos is to present an overview of the functionality, to show that CGAL is robust and fast, and can deal with complex data sets.

2D Triangulations and Mesh Generation : CGAL offers triangulations of point sets [4] and it can deal with constraints which allow to delimit domains. The mesh generation algorithms add Steiner points on constraints or inside domains such that triangles fulfill user defined quality criteria as minimal angles, or triangle size. The triangle size can be constant, or it can be governed by a sizing field, for example local feature size.

3D Triangulations, Surface and Volume Mesh Generation : The generated meshes are three-dimensional isotropic simplicial meshes [5, 16]. The discretized region may be a monodomain, i.e. a single connected component, or a multidomain subdivided into different subdomains. The domain is input as an oracle able to answer queries, of a few different types, on the domain. In the current version, the mesh generator is able to discretize domains described through implicit functions, 3D images or polyhedral boundaries. The output is a 3D mesh of the domain volume and conformal surface meshes for all the boundary and subdividing surfaces.

The algorithm can also respect one-dimensional features formed by intersections of surfaces, or sharp features of objects.

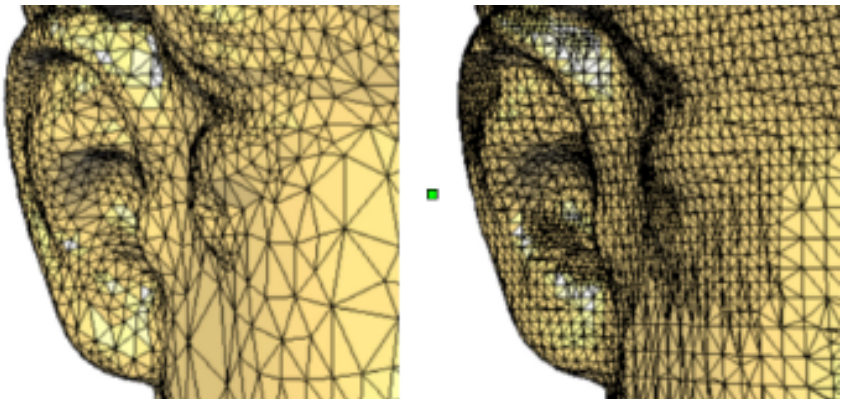


FIGURE 1 – The Delaunay surface mesh has nicely formed triangles and less triangles where the surface is almost flat, compared to the mesh generated with marching cubes.

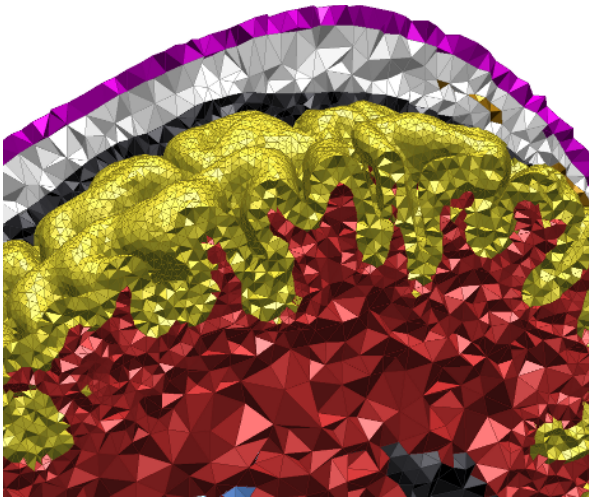


FIGURE 2 – Volume mesh for 3D segmented voxel data.

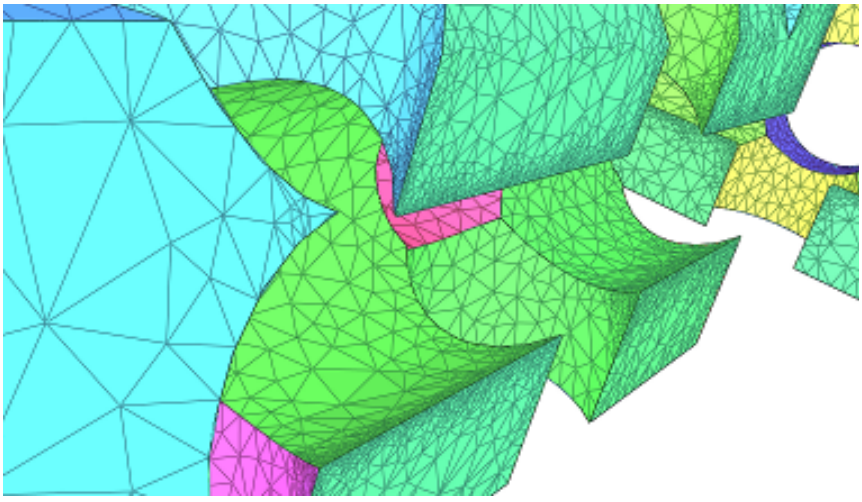


FIGURE 3 – Volume mesh and remeshed surface mesh for a mechanical part.

The package further provides mesh optimization algorithms to improve the aspect ratio of tetrahedra [18, 9].

Periodic Triangulations : This package allows to build and handle triangulations of infinite periodic point sets in 3D (see Figure 4), which can also be seen as triangulations of the three-dimensional flat torus [7, 8]. Triangulations are built incrementally and can be modified by insertion or removal of vertices. The

package provides Delaunay triangulations and offers nearest neighbor queries and primitives to build the dual Voronoi diagrams.

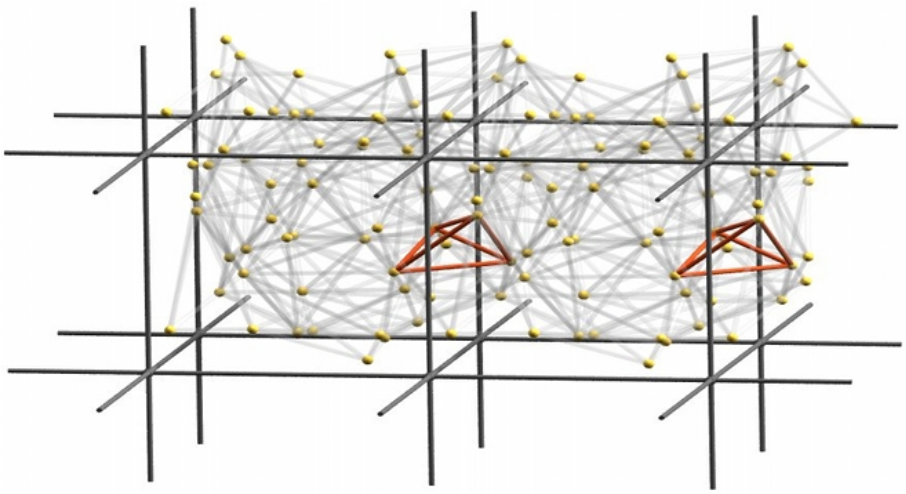


FIGURE 4 – 3D periodic Delaunay triangulation

Some work is in progress regarding periodic surface and volume mesh generation (Figure 5) [6].

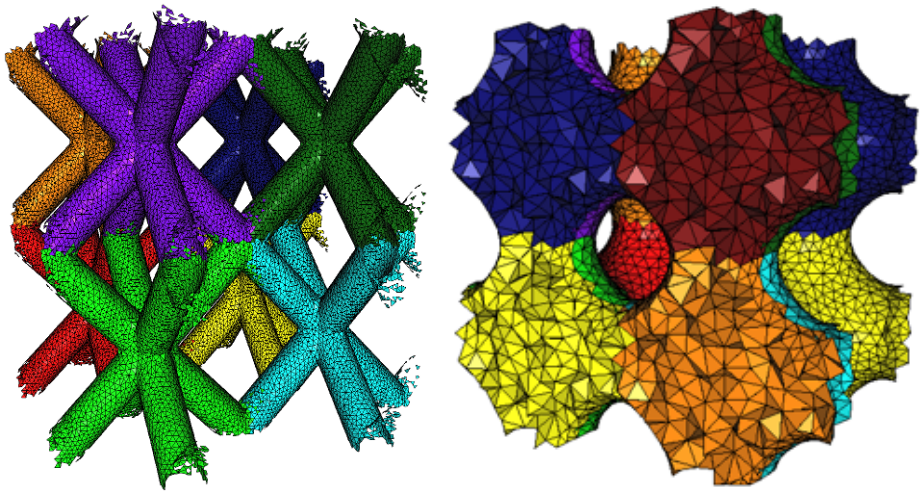


FIGURE 5 – Periodic surface (left) and volume (right) meshes. *(left) Data courtesy of Maarten Moesen, Department of Metallurgy and Materials Engineering (MTM), K.U. Leuven, Belgium.*

Surface Reconstruction : This package allows to take a point cloud coming for example from a laser range scanner. Given a set of 3D points with oriented normals sampled on the boundary of a 3D solid, the Poisson surface reconstruction method [13] solves for an approximate indicator function of the inferred solid, whose gradient best matches the input normals. The output scalar function, represented in a refined Delaunay triangulation, is then iso-contoured using the Surface mesh generator (see Figure 6).

CGAL offers tools to clean the input which is often noisy, has outliers, and must often be locally simplified, and tools to estimate normals.

Polyhedral Surfaces : CGAL offers algorithms that operate on polyhedral surfaces, such as mesh simplification [15], estimation of local differential properties, principal component analysis, as well as various surface subdivision schemes. The polyhedral surface are represented by a halfedge data structure. As they are limited to manifold surfaces, efforts are under way to implement combinatorial maps [10].

Boolean Operations : The 2D Boolean operations package is based on overlays of arrangements of curves [19]. Polygon edges can be line segments, as well as circular arcs or Bezier curves. The former

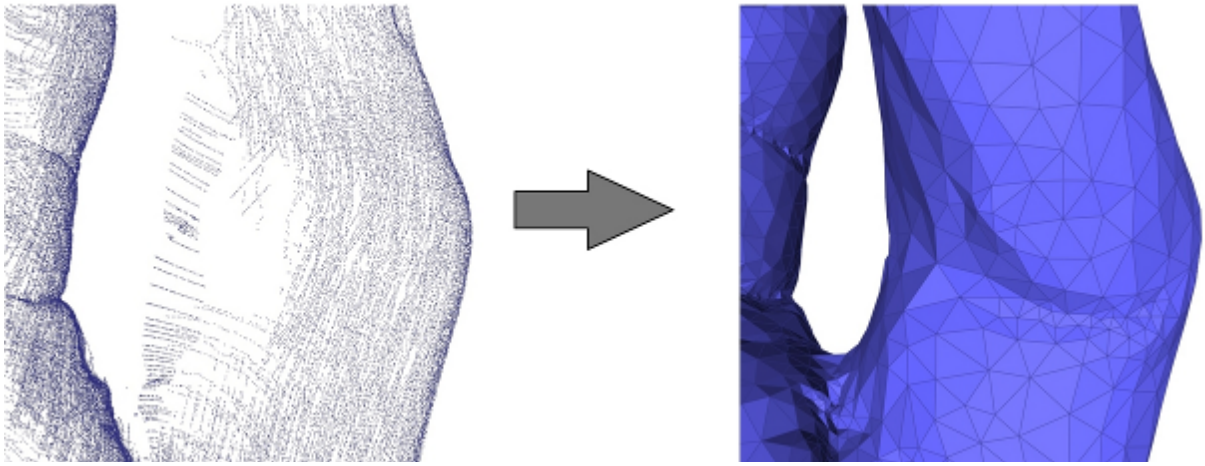


FIGURE 6 – The Poisson surface recinstruction algorithm copes well with undersampled areas.

appear often in 2D CAD data, the latter in descriptions of fonts.

The 3D Boolean operations package is based on the theory of the Nef polyhedra, which can represent the result of any finite sequence of Boolean operations applied on half spaces [3]. This allows open faces, antennas, isolated points, or volumes sharing only an edge.

Polygon Offsets : CGAL offers two variants for computing inner or outer offsets of a polygon. The first one is based on Minkowski sums which results in round caps at corners, and the second one is based on the straight skeleton which is a polygon with only straight line segments (see Figure 7). Some users simplify polygons by computing the inner offset of the outer offset of the input polygon.

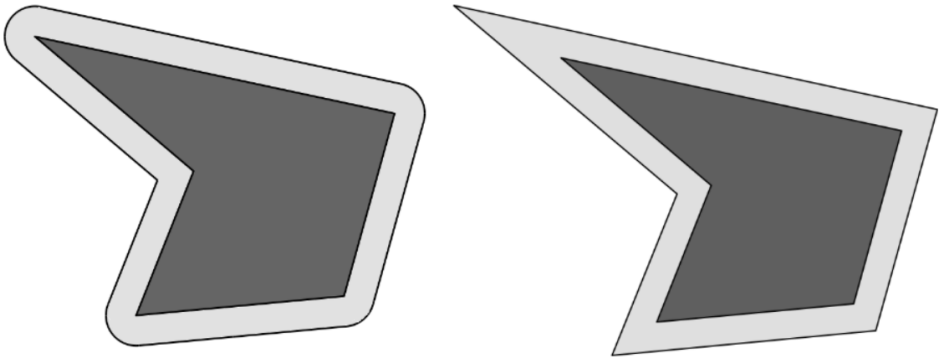


FIGURE 7 – Offsets, based on : Minkowski sum (left) and straight skeleton (right)

Neighbor Search : CGAL offers data structures for computing k-nearest/furthest neighbors for points in arbitrary dimensions. There are variants for approximative queries, and for various metrics. These data structures are based on the kd-tree.

CGAL further offers data structures to perform efficient intersection and distance queries against sets of finite 3D geometric objects. The set of geometric objects stored in the data structure can be queried for intersection detection, intersection computation and distance. The intersection queries can be of any type, provided that the corresponding intersection predicates and constructors are implemented in the traits class. The distance queries are limited to point queries. Examples of intersection queries include line objects (rays, lines, segments) against sets of triangles, or plane objects (planes, triangles) against sets of segments. An example of a distance query consists of finding the closest point from a point query to a set of triangles. The underlying data structure is an AABB tree [17].

Bounding Areas : This package offers algorithms to compute the smallest enclosing circle, ellipse, rectangle, quadrilateral, or strip for a given point set. It further offers the convex hull algorithm.

Voronoi Diagrams : For a given set of sites, a Voronoi diagram decomposes space in cells, generally one cell per site. CGAL offers Voronoi diagrams for 2D points, segments, and circles. The Voronoi edges are then arcs of lines, parabolas, and hyperbolas. The latter are dealt with symbolically, that is they are not discretized polylines. The segment Voronoi diagram is often used for computing the medial axis of a polygon.

Polyline Simplification : This package is work in progress that allows to simplify several polylines simultaneously while maintaining the topology. The simplification of a coastline will not produce an island which is no longer surrounded by water, and isolines will not touch each other, if they did not touch before the simplification.

3 Software issues

The design of CGAL [11, 12] follows the generic programming and the exact geometric computing paradigm.

3.1 Generic Programming

Just as a `std::set` of the Standard Template Library, can store objects of an arbitrary type as long as a comparison operator for this type is defined, a CGAL data structure that deals with points can operate on an arbitrary point type as long as several geometric predicates like lexicographical order, and incircle test are defined for this point type. The types and predicates that must be provided are part of the documentation for each CGAL data structure. In the same spirit CGAL provides the models of the Boost Graph Library (BGL) `boost::graph` concept, which allows to run any graph algorithm of the BGL directly on CGAL data structures.

Generic Programming [2], with the notion of concepts defining an API and models implementing it, is key to reuse, and to the ease of integration of CGAL components into existing applications. This is important, as few users start from scratch and build entire applications only on CGAL.

3.2 Exact geometric computing

Geometric algorithms are notorious for their lack of robustness when faced with numerical instability issues. In order to deal with this problem, CGAL follows the Exact Geometric Computing paradigm [14]. This paradigm states that computing the low-level primitives exactly guarantees the correctness of the higher-level algorithms.

These primitives are categorized in two types. On one hand, geometric predicates compute Boolean-like properties, like the orientation of three points in the plane, or the decision whether two segments intersect. On the other hand, geometric constructions build new geometric objects or numerical quantities, like the computation of a distance or the intersection point of two lines.

3.3 Language bindings

CGAL is written in C++, but there is work in progress on offering Java and Python bindings based on Swig. The 2D and 3D triangulations of MATLAB are wrapped version of the CGAL classes. [1]

4 License

CGAL is distributed under an Open Source which makes that the library has a large user community. As the QPL license does not allow closed source development, the library is at the same time available

1. video : `http://www.mathworks.com/products/demos/shipping/matlab/New-MATLAB-Mathematics-Features-in-R2009a.html`

under commercial licenses. Commercial users include companies as Agilent, Total, ESRI, The Mathworks, Schlumberger, Navteq, to name just a few.

5 Acknowledgments

Our thanks to the CGAL developers [2] who wrote the demos and made some of the pictures.

Références

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

[2] M. H. Austern. *Generic Programming and the STL.* Addison Wesley, 1999.

[3] H. Bieri. *Nef polyhedra : A brief introduction*. Computing Suppl., 10 :43-60, 1995.

[4] J-D. Boissonnat, O. Devillers, S. Pion, M. Teillaud, and M. Yvinec. *Triangulations in* CGAL, Computational Geometry : Theory and Applications, 22 :5-19, 2002.

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

[6] M. Caroli, V. Fisikopoulos, and M. Teillaud. *Meshing of triply-periodic surfaces in* CGAL. Poster presentation, Seventh International Conference on Curves and Surfaces, 2010.
ftp://ftp-sop.inria.fr/geometrica/mcaroli/periodic_meshing_poster_small.png

[7] M. Caroli and M. Teillaud. *Computing 3D periodic triangulations*. 17th European Symposium on Algorithms, Lecture Notes in Computer Science, 5757 :37-48, 2009.

[8] M. Caroli and M. Teillaud. *Delaunay triangulations of point sets in closed euclidean d-manifolds*. 26th European Workshop on Computational Geometry, 101-104, 2010.

[9] S-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S-H. Teng. *Sliver exudation*. J. ACM, 47(5) :883-904, 2000.

[10] G. Damiand. *Contributions aux Cartes Combinatoires et Cartes Généralisées : Simplification, Modèles, Invariants Topologiques et Applications*. Habilitation à diriger des recherches, Université Lyon 1, 2010. http://tel.archives-ouvertes.fr/tel-00538456/.

[11] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr. *On the design of* CGAL*, a computational geometry algorithms library*. Softw. Pract. Exp., 30(11) :1167-1202, 2000.

[12] E. Fogel and M. Teillaud. *Generic programming and the* CGAL *library*. Effective Computational Geometry for Curves and Surfaces, J-D. Boissonnat and M. Teillaud, editors, 313-320. Springer-Verlag, Mathematics and Visualization, 2006.

[13] M. Kazhdan, M. Bolitho, and H. Hoppe. *Poisson Surface Reconstruction*. Symp. on Geometry Processing, 61-70, 2005.

[14] C. Li, S. Pion, and C. Yap. *Recent progress in exact geometric computation*. Special issue on the practical development of exact real number computation, Journal of Logic and Algebraic Programming, 64(1) :85-111, 2005.

[15] P. Lindstrom and G. Turk. *Fast and memory efficient polygonal simplification*. IEEE Visualization, 279-286, 1998.

[16] L. Rineau and M. Yvinec. *A generic software design for Delaunay refinement meshing*. Computational Geometry : Theory and Applications, 38 :100-110, 2007.

[17] P. Terdiman. *OPCODE 3D Collision Detection library*, 2005. http://www.codercorner.com/Opcode.htm.

[18] J. Tournois, C. Wormser, P. Alliez, and M. Desbrun. *Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation*. ACM Transactions on Graphics (SIGGRAPH'2009 Proc.), 28(3) :75 :1-75 :9, 2009.

[19] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. *Advanced programming techniques applied to* CGAL*'s arrangement package*. Computational Geometry : Theory and Applications, 38(1-2), 2007.

2. http://www.cgal.org/people.html