Triangulations in CGAL - To non-Euclidean spaces and beyond!
Monique Teillaud

To cite this version:
Monique Teillaud. Triangulations in CGAL - To non-Euclidean spaces and beyond!. EuroCG 2020 - 36th European Workshop on Computational Geometry, Mar 2020, Würzburg, Germany. hal-02510046

HAL Id: hal-02510046
https://hal.inria.fr/hal-02510046
Submitted on 17 Mar 2020

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.
Triangulations in

CGAL

EuroCG 2020

Monique Teillaud
Triangulations in

CGAL

To non-Euclidean spaces and beyond!

EuroCG
2020

To infinity and beyond!

By Edo-biscuit - Own work, CC BY-SA 4.0

Monique Teillaud
History of the **CGAL** project

**1995 - Start**

Academic project
History of the CGAL project

1995 - Start
Academic project

January 2003 - Creation of Geometry Factory
as an INRIA startup, by Andreas Fabri
sells commercial licenses, support, customized developments

November 2003 - Open Source project
new contributors

November 2019 - CGAL 5.0
C++14
Goals

Gather efforts

PlaGeo, SpaGeo (Utrecht)

XYZ GeoBench (ETH Zürich)

LEDA (MPII Saarbrücken)

C++GAL
  (INRIA Sophia Antipolis)

...
Goals

Gather efforts

Promote research in CG

“make the large body of geometric algorithms developed in the field of computational geometry available for industrial applications”

⇒ high quality review process, . . .

⇒ robust code
Goals

Gather efforts

Promote research in CG

Reward structure for implementations in academia

⇒ high quality review process, . . .

⇒ robust code
Technical choices

C++

Genericity/flexibility through templates

Exact Geometric Computation [Yap]

Exact predicates $\leadsto$ exact decisions
> 80 chapters in the manual

- Bounding Volumes
- Polyhedral Surface
- Boolean Operations
- Triangulations
- Voronoi Diagrams
- Mesh Generation
- Subdivision
- Simplification
- Parameterization
- Streamlines
- Ridge Detection
- Neighbour Search
- Kinetic Data structures
- Lower Envelope
- Arrangement
- Intersection Detection
- Minkowski Sum
- PCA
- Polytope distance
- QP Solver

CGAL
Monique Teillaud
<table>
<thead>
<tr>
<th>Triangulations in CGAL</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Triangulations (1997)</td>
<td></td>
</tr>
<tr>
<td>3D Triangulations (2000) and meshes (2009)</td>
<td></td>
</tr>
<tr>
<td>dD Triangulations (2015)</td>
<td></td>
</tr>
<tr>
<td>2D Periodic Triangulations (2013)</td>
<td></td>
</tr>
<tr>
<td>3D Periodic Triangulations (2009) and meshes (2018)</td>
<td></td>
</tr>
<tr>
<td>2D Hyperbolic Triangulations (2019)</td>
<td></td>
</tr>
<tr>
<td>2D Periodic Hyperbolic Triangulations (2019)</td>
<td></td>
</tr>
</tbody>
</table>
General design

Triangulation < Geom_traits, TDS >
General design

Triangulation \( \langle \text{Geom_traits}, \text{TDS} \rangle \)

geometry

objects (points, etc)
predicates (e.g., in_circle)
constructions (e.g., circumcenter)

data structure

Cell = \( d \)-simplex
\( \rightarrow d \) vertices
\( \rightarrow d \) adjacent cells

Vertex
point
\( \rightarrow \) one incident cell
Euclidean Delaunay Triangulations
Euclidean Delaunay Triangulations

Bowyer’s incremental algorithm
Euclidean Delaunay Triangulations

Bowyer’s incremental algorithm

find simplices in conflict

Triangulation using Geom_traits
Euclidean Delaunay Triangulations

Bowyer’s incremental algorithm

find simplices in conflict
Triangulation
using Geom_traits

the conflict region forms a topological ball
Euclidean Delaunay Triangulations

Bowyer’s incremental algorithm

find simplices in conflict
Triangulation using Geom_traits

create new simplices
TDS
Euclidean Delaunay Triangulations

compactification of $\mathbb{R}^d$

$TDS = \text{triangulation of } S^d$

no boundary

each cell = $d$-simplex
Euclidean Delaunay Triangulations

Robustness

Exact Geometric Computation paradigm [Yap]

exact predicates \neq exact arithmetics

Filtering

easy cases are more frequent
\implies cost \simeq cost of approximate (double) computation
Euclidean Delaunay Triangulations

Approximate evaluation $P^a(x)$
+ Error $\epsilon$

$P^a(x) > \epsilon$

Filtering

Yes

$\text{Sign}(P(x)) = \text{Sign}(P^a(x))$

No

Exact computation

Robustness

Arithmetic issues

Yes

No

Arithmetic issues
symbolic perturbation

only perturbs the \texttt{in\_sphere} predicate

[Devillers, T. SODA’03, CGTA’11]
Euclidean Delaunay Triangulations

Tricks to improve efficiency

EuroCG’12 invited talk by Olivier Devillers
https://hal.inria.fr/hal-00850561
Euclidean Delaunay Triangulations

```cpp
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Delaunay_triangulation_3>

typedef CGAL::Exact_predicates_inexact_constructions_kernel K;
typedef CGAL::Delaunay_triangulation_3<K> Delaunay;
typedef Delaunay::Point Point;

int main()
{
    Delaunay T;
    T.insert(Point(0,0,0));
    T.insert(Point(1,0,0));
    T.insert(Point(0,1,0));
    T.insert(Point(0,0,1));
    T.insert(Point(2,2,2));
    T.insert(Point(-1,0,1));

    return 0;
}
```
Euclidean Delaunay Triangulations

```cpp
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Projection_traits_xy_3.h>
#include <CGAL/Delaunay_triangulation_2.h>
#include <fstream>

typedef CGAL::Exact_predicates_inexact_constructions_kernel K;
typedef CGAL::Projection_traits_xy_3<K> Gt;
typedef CGAL::Delaunay_triangulation_2<Gt> Terrain;
typedef K::Point_3 Point;

int main()
{
    Terrain T;
    T.insert(Point(0,0,0));
    T.insert(Point(1,0,0));
    T.insert(Point(0,1,0));
    // etc

    return 0;
}
```

Points : 3D
Predicates : on their 2D projections

Flexibility
Euclidean Delaunay Triangulations

```cpp
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Delaunay_triangulation_3.h>
#include <CGAL/Delaunay_triangulation_cell_base_3.h>
#include <CGAL/Triangulation_vertex_base_with_info_3.h>
#include <CGAL/IO/Color.h>

typedef CGAL::Exact_predicates_inexact_constructions_kernel K;
typedef CGAL::Triangulation_vertex_base_with_info_3<CGAL::Color, K> Vb;
typedef CGAL::Delaunay_triangulation_cell_base_3<K> Cb;
typedef CGAL::Triangulation_data_structure_3<Vb, Cb> Tds;
typedef CGAL::Delaunay_triangulation_3<K, Tds> Delaunay;
typedef Delaunay::Point Point;

int main()
{
    Delaunay T;
    T.insert(Point(0,0,0));
    // etc
    Delaunay::Finite_vertices_iterator vit;
    for (Delaunay::Vertex_handle v : T.finite_vertex_handles())
        if (T.degree(v) == 6)
            v->info() = CGAL::red();
    return 0;
}
```

Monique Teillaud
Euclidean Delaunay Triangulations

- fully dynamic
- fully robust
- also weighted

2D
- (insertion with flips?)
- $\approx 10$ M points / second
  - [Yvinec CGAL'97]
  - also constrained

3D
- $\approx 1$ M points / second
  - [T. CGAL'00] [Pion, T. CGAL'01*]
  - multicore version
  - [Jamin CGAL'14]
Hyperbolic triangulations
Hyperbolic triangulations

Once upon a time...

Compute the 3D Delaunay triangulation

of a set of points

lying in two parallel planes
Hyperbolic triangulations

[Boissonnat 1988]

$P_1$

$P_2$
Hyperbolic triangulations

[Boissonnat 1988]

$P_1$

$P_2$
Hyperbolic triangulations

[Boissonnat 1988]

Monique Teillaud
Hyperbolic triangulations

[Boissonnat 1988]

$P_1$

$P_2$
Hyperbolic triangulations

[Boissonnat 1988]

Monique Teillaud
Hyperbolic triangulations

[Boissonnat 1988]
Hyperbolic triangulations

[Boissonnat 1988]
Hyperbolic triangulations

[Boissonnat 1988]

Monique Teillaud
Hyperbolic triangulations

[Boissonnat 1988]

Monique Teillaud
Hyperbolic triangulations

[Boissonnat 1988]
Hyperbolic triangulations

[Boissonnat 1988]
Hyperbolic triangulations

What if the planes are not parallel?

[Boissonnat, Cérézo, Devillers, T. IJCGA’96]
Hyperbolic triangulations

What if the planes are not parallel?

[Boissonnat, Cérézo, Devillers, T. IJCGA’96]
Hyperbolic triangulations

What if the planes are not parallel?

pencil with limit points

[Boissonnat, Cérézo, Devillers, T. IJCGA’96]
Hyperbolic triangulations

What if the planes are \textbf{not} parallel?

pencil with limit points

[Boissonnat, Céręzo, Devillers, T. IJCGA’96]
Hyperbolic triangulations

What if the planes are not parallel?

pencil with limit points

hyperbolic line in the Poincaré half-plane

[Boissonnat, Cérézo, Devillers, T. IJCGA’96]
Hyperbolic triangulations

What if the planes are not parallel?

pencil with limit points

concentric pencil in the Poincaré half-plane

[Boissonnat, Cérézo, Devillers, T. IJCGA’96]
Hyperbolic triangulations

What if the planes are not parallel?

same algorithm

hyperbolic Voronoi diagram

Poincaré half-plane model

[Boissonnat, Cérézo, Devillers, T. IJCGA’96]
Hyperbolic triangulations

- hyperbolic circles are Euclidean circles
- same combinatorics as Euclidean Voronoi diagram

Poincaré half-plane model

hyperbolic Voronoi diagram
Hyperbolic triangulations

Poincaré half-plane model

Hyperbolic circles are Euclidean circles
\[ \mapsto \] same combinatorics as Euclidean Voronoi diagram

except that some Euclidean circles are not hyperbolic circles

[Boğdanov, Devillers, T. JoCG’14]

hyperbolic Voronoi diagram

Monique Teillaud
Hyperbolic triangulations

Hyperbolic_Delaunay_triangulation_2

inherits from

Delaunay_triangulation_2

[Bogdanov, Devillers, T. JoCG’14]

[Bogdanov, Iordanov, T. CGAL’19]
#include <CGAL/Hyperbolic_Delaunay_triangulation_2.h>
#include <CGAL/Hyperbolic_Delaunay_triangulation_traits_2.h>
#include <vector>

typedef CGAL::Hyperbolic_Delaunay_triangulation_traits_2<> Gt;
typedef Gt::Point_2 Point_2;
typedef CGAL::Hyperbolic_Delaunay_triangulation_2<Gt> Dt;

int main(int argc, char** argv)
{
    std::vector<Point_2> pts; \ \ pts filled in some way

    Dt dt_during; \ \ // hyperbolic filtering at each step:
    std::vector<Point_2>::iterator ip;
    for(ip = pts.begin(); ip != pts.end(); ++ip) dt_during.insert(*ip);

    Dt dt_end; \ \ // hyperbolic filtering only once at the end:
    dt_end.insert(pts.begin(), pts.end());

    return 0;
}
Periodic Delaunay triangulations
Periodic Delaunay triangulations

motivation: cosmic web
[v.d. Weijgaert, Groningen]

material for bone scaffold
[Moesen\textsuperscript{+}, Leuven]

granular material
[Ludig\textsuperscript{+}, Twente]
Periodic Delaunay triangulations
Periodic Delaunay triangulations
Periodic Delaunay triangulations

[Caroli, T. SoCG’08 video]
Periodic Delaunay triangulations

flat torus

\[ \mathbb{T}^2 \sim \mathbb{R}^2 / G \]

\[ G = \langle t_x, t_y \rangle \]

universal covering space
Periodic Delaunay triangulations

flat torus

\[ T^3 \sim \mathbb{R}^3 / G \]

\[ G = \langle t_x, t_y, t_z \rangle \]
Periodic Delaunay triangulations

Bowyer’s incremental algorithm

the conflict region forms a topological ball
Periodic Delaunay triangulations

Bowyer’s incremental algorithm

the conflict region does not form a topological ball
Periodic Delaunay triangulations

add a few dummy points (e.g., 36 in 3D)
and remove them asap

or

compute in a covering space (e.g., 27-sheeted in 3D)
and switch back to 1 sheet asap

so that the triangulation is always a simplicial complex

$\Phi(\text{largest empty disk}) < \text{systole}/2$

[Caroli, T. ESA’09]
Periodic Delaunay triangulations

Periodic Delaunay triangulation< TDS Geom_traits >

combinatorial triangulation of $S^3$ reused for $T^3$

Cell = $d$-simplex
$\rightarrow d$ vertices
$\rightarrow d$ adjacent cells
$d$ translations

Vertex
canonical point
$\rightarrow$ one incident cell
Periodic Delaunay triangulations

Periodic Delaunay triangulations

```cpp
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Periodic_3_Delaunay_triangulation_traits_3.h>
#include <CGAL/Periodic_3_Delaunay_triangulation_3.h>
#include <vector>

typedef CGAL::Exact_predicates_inexact_constructions_kernel K;
typedef CGAL::Periodic_3_Delaunay_triangulation_traits_3<K> Gt;
typedef CGAL::Periodic_3_Delaunay_triangulation_3<Gt> P3DT3;
typedef P3DT3::Point Point;
typedef P3DT3::Iso_cuboid Iso_cuboid;

int main(int, char**)
{
    Iso_cuboid domain(-1,-1,-1,2,2,2); // the fundamental domain

    std::vector<Point> pts;
    // pts is filled in some way...

    P3DT3 T(pts.begin(), pts.end(), domain);

    return 0;
}
```
Periodic Delaunay triangulations

Users

- Astrophysics
- Particle physics
- Nanostructures
- Granular materials

[v.d. Weijgaert$^+$]
[SoCG’12 video]
[Sousbie$^+$]
[Melnikov$^+$]
Periodic Delaunay triangulations

Users are always asking for more...

non-cubic case

dD closed flat manifolds [Caroli, T. SoCG’11 DCG’16]

uses covering spaces

in practice: dummy points     In progress

\( \Phi(\text{largest empty disk}) < \text{systole}/2 \)
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface
Delaunay triangulation of the Bolza surface

Motivation: e.g., in mathematical physics

- Chaotic motion
  - Balazs, Voros '86

- Glass-forming liquid
  - Sausset, Tarjus, Viot '08

- Visual perception of textures
  - Chossat, Faye, Faugeras '11
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

surface of genus 2

Monique Teillaud
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

surface of genus 2

Monique Teillaud
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

the regular octagon tiles the hyperbolic plane = covering space

Monique Teillaud
Delaunay triangulation of the Bolza surface

\[ M_2 \sim \mathbb{H}^2 / G \quad G = \langle a, b, c, d \mid \text{relation} \rangle \]

a, b, c, d hyperbolic translations

do not commute
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

points on the surface
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

translated images
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

translated images
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

translated images
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

Delaunay triangulation of the infinite point set
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

Delaunay triangulation of the surface

(14 "dummy" points)

$\Phi(\text{largest empty disk}) < \text{systole}/2$

[Bogdanov, T., Vegter SoCG’16]
[Iordanov, T. SoCG’17]
[Iordanov, T. CGAL’19]
The **CGAL** project
Becoming a user

http://www.cgal.org

Many platforms

Linux, MacOS, Windows

g++, VC++, clang, . . .
Becoming a **CGAL** user

http://www.cgal.org

Many platforms

- Linux, MacOS, Windows
- g++, VC++, clang, . . .

Download from github or use

- Linux distribution (Debian, . . .)
- Mac distribution (macports, brew, . . .)

Euclidean 2D and 3D triangulations

- in Matlab
- in Python through CGAL bindings
Becoming a CGAL user

http://www.cgal.org

Follow “Getting started”

easier than ever!

- “Since CGAL version 5.0, CGAL is header-only by default, which means that there is no need to build CGAL before it can be used. ”

- Simple viewer
Becoming a \texttt{CGAL} user

http://www.cgal.org

Follow “Getting started”

Read the \textit{User manual} of your favorite package

Read, compile and run \texttt{examples}
Becoming a **CGAL** user

http://www.cgal.org

Follow “Getting started”

Read the **User manual** of your favorite package

Read, compile and run **examples**

**Write your own code!**

looking at the **Reference manual**

Obey license GPLv3+ (or buy a commercial license)
Becoming a CGAL user

http://www.cgal.org

Molecular Modeling
Particle Physics, Fluid Dynamics, Microstructures
Medical Modeling and Biophysics
Geographic Information Systems
Games
Motion Planning
Sensor Networks
Architecture, Buildings Modeling, Urban Modeling
Astronomy
2D and 3D Modelers
Mesh Generation and Surface Reconstruction
Geometry Processing
Computer Vision, Image Processing, Photogrammetry
Computational Topology and Shape Matching
Computational Geometry and Geometric Computing
Becoming a **CGAL** user

some commercial users (2012)
Becoming a CGAL user

Ipelets!
Ipelets!

Generators

e.g. 200 points in a disk

$\mathbb{R}^2$
Ipelets!

Generators
e.g. 200 points in a disk

\[ \mathbb{R}^2 \]

“Diagrams”
\[ \rightarrow \] Voronoi

Monique Teillaud
Becoming a CGAL user

Ipelets!

Generators
e.g. 200 points in a disk \( \mathbb{R}^2 \)

“Triangulations” \( \rightarrow \) Delaunay
Becoming a CGAL developer

Any idea for a new package?

Contact the editorial board!

even before starting to code

http://www.cgal.org

Design and coding style

Infrastructure

Review and integration process
Becoming a **CGAL** developer

Contributors are clearly acknowledged

- Appear as Authors in manual chapters

**Visibility**

- Listed in the “People” web page

---

**CGAL 5.0.1 - 3D Periodic Triangulations**

**User Manual**

**Authors**

Manuel Caroli, Aymeric Pellé, Mael Rouxel-Labbé and Monique Teillaud
Becoming a **CGAL** developer

Contributors are clearly acknowledged

Visibility

appreciated by

companies (Google, Intel, Apple, . . . ) 😊 😞

research institutes (INRIA, ETHZ, . . . ?)
Becoming a CGAL developer

Contributors are clearly acknowledged

[The institution of the] authors keep the copyright

( must agree to distribute the code in CGAL under GPLv3+ )
Becoming a developer

Key for success = Diversity of its members

gather many skills:

- maths
- algorithms
- C++
- development tools
- ...

Join the crowd!
Becoming a CGAL developer

Monique Teillaud
Thanks to

past, present, future

developers and

users of CGAL

you all!

Otfried for Ipe
Triply periodic minimal surfaces

(a) Gyroid

(b) Schwartz P

Diamond

Gyroid

(c) Diamond

(d) Gyroid