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

Monique Teillaud

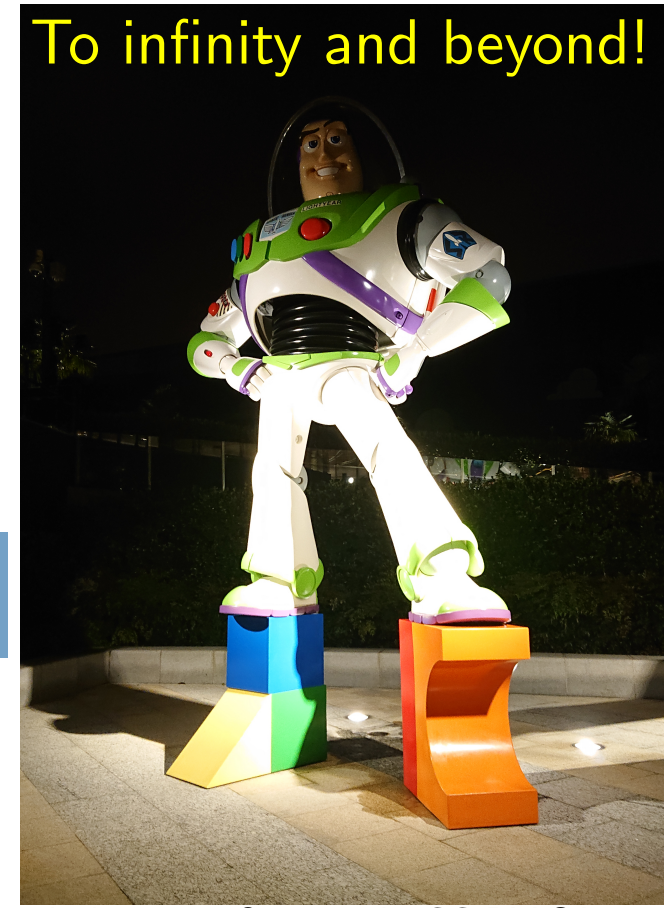
Triangulations in

Monique Teillaud

Triangulations in

CGAL

To non-Euclidean spaces and beyond!



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

EuroCG 2020

Monique Teillaud

Inria cnrs Loria Laboratoire lorrain de recherche en informatique et ses applications UNIVERSITÉ DE LORRAINE

History of the CGAL project

1995 - Start

Academic project

History of the CGAL project

1995 - Start

Academic project

January 2003 - Creation of 

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)

...

Jodelet/Lépinay - CC Attribution-Share Alike 2.0 France

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 "

$\Longrightarrow$ high quality
review process, ...

$\Longrightarrow$ robust code

Goals

Gather efforts

Promote research in CG

Reward structure for implementations in academia

$\Longrightarrow$ high quality
review process, ...

$\Longrightarrow$ robust code

Technical choices

C++

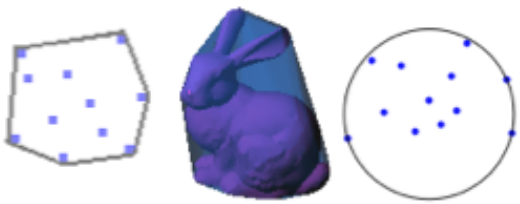
Genericity/flexibility through templates

Exact Geometric Computation [Yap]

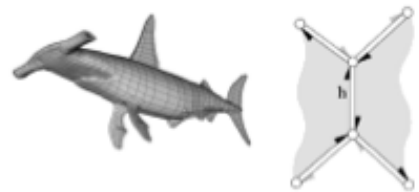
Exact predicates $\rightsquigarrow$ exact decisions

Contents

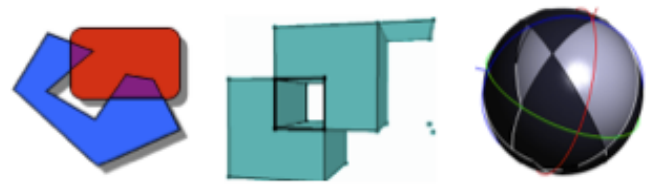
> 80 chapters in the manual



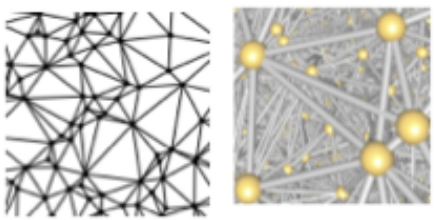
Bounding Volumes



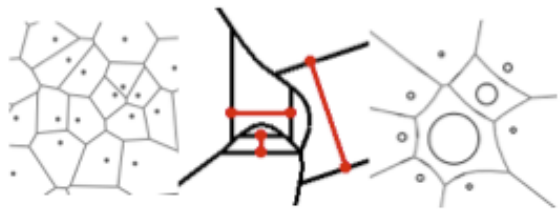
Polyhedral Surface



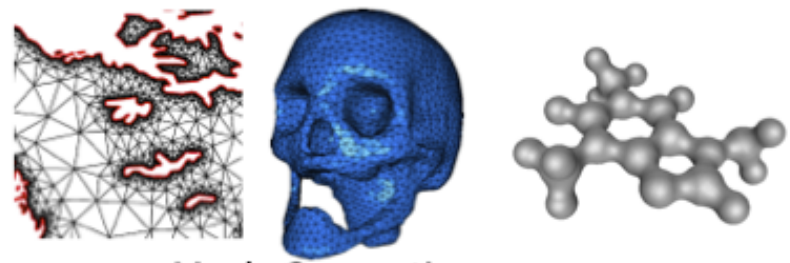
BooleanOperations



Triangulations

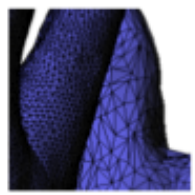


Voronoi Diagrams

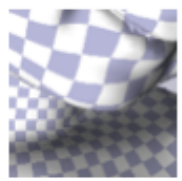


Mesh Generation

Subdivision



Simplification

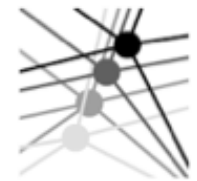


Parameterization

Streamlines

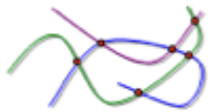
Ridge Detection

Neighbour Search

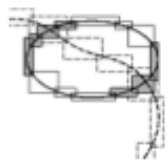


Kinetic Data structures

Lower Envelope

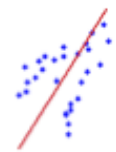


Arrangement



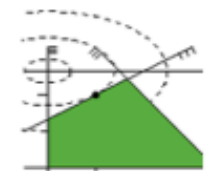
Intersection Detection

Minkowski Sum



PCA

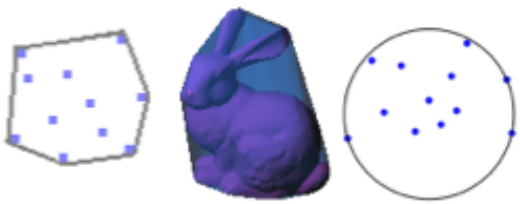
Polytope distance

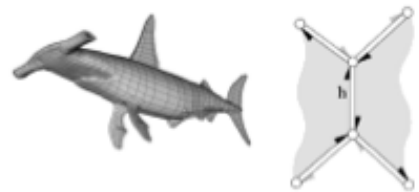


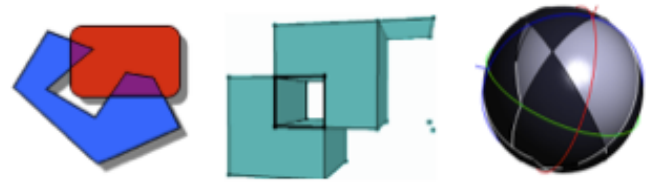
QP Solver

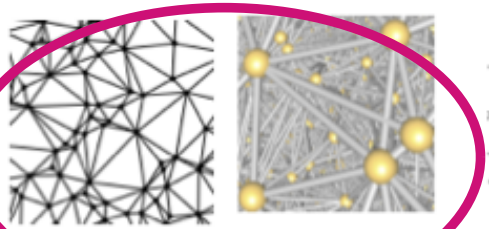
Contents

> 80 chapters in the manual

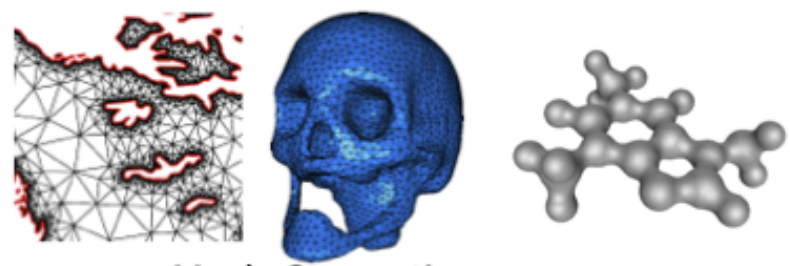

Bounding Volumes


Polyhedral Surface

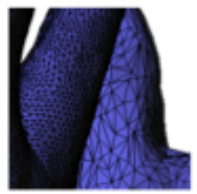

BooleanOperations

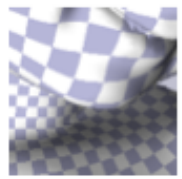

Triangulations


Voronoi Diagrams


Mesh Generation


Subdivision

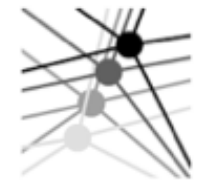

Simplification


Parameterization

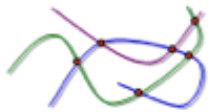

Streamlines

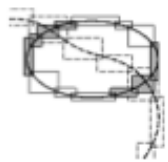

Ridge Detection


Neighbour Search

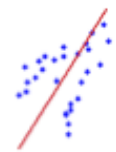

Kinetic Data structures


Lower Envelope

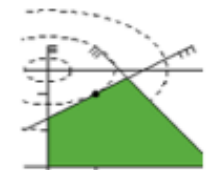

Arrangement


Intersection Detection


Minkowski Sum


PCA


Polytope distance


QP Solver

Triangulations in CGAL

2D Triangulations (1997)

3D Triangulations (2000) and meshes (2009)

dD Triangulations (2015)

2D Periodic Triangulations (2013)

3D Periodic Triangulations (2009) and meshes (2018)

2D Hyperbolic Triangulations (2019)

2D Periodic Hyperbolic Triangulations (2019)

General design

```
Triangulation < Geom_traits, TDS >
```

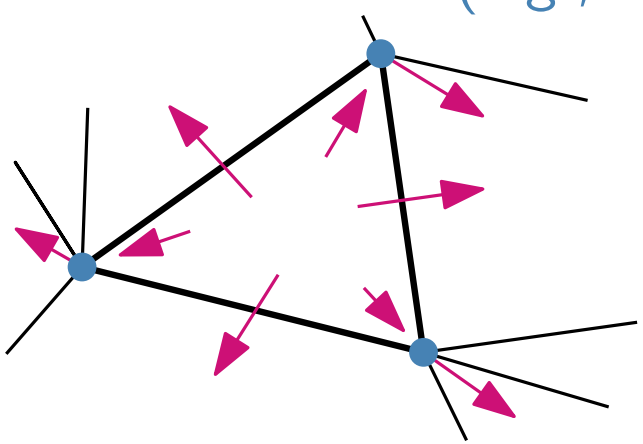
General design

```
Triangulation <Geom_traits, TDS>
```

geometry

data structure

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



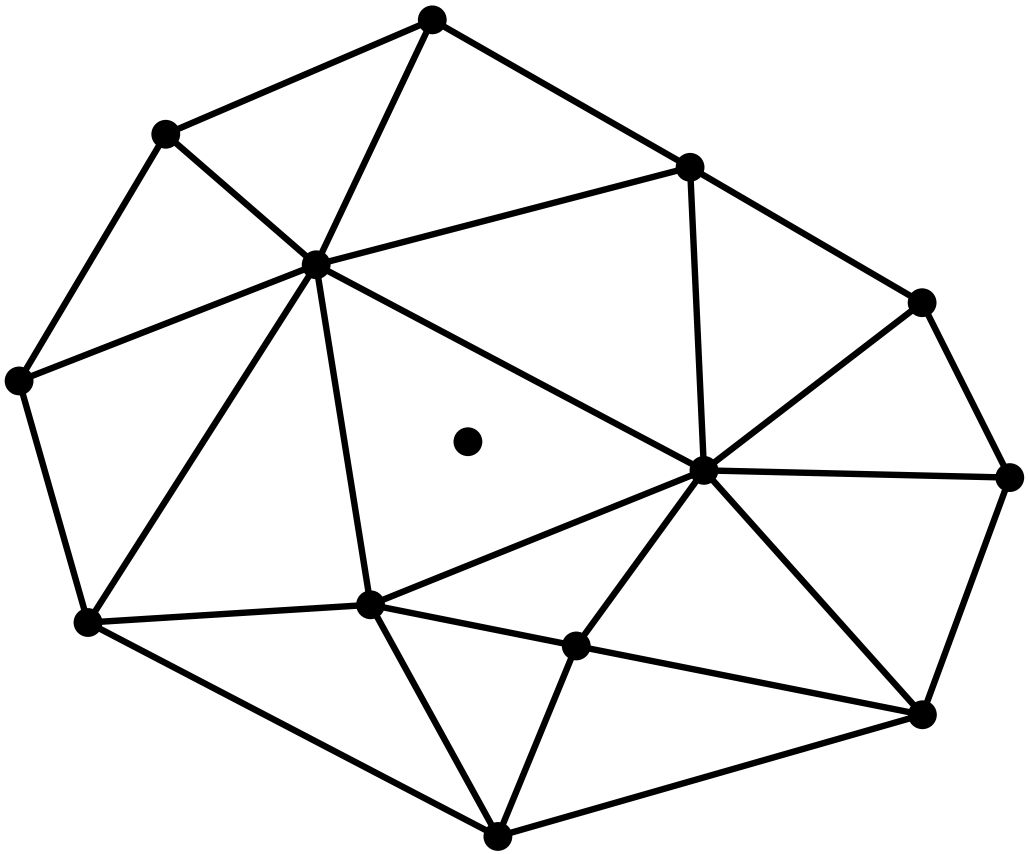
Cell = d-simplex
$\longrightarrow$ d vertices
$\longrightarrow$ d adjacent cells

Vertex
point
$\longrightarrow$ one incident cell

Euclidean Delaunay Triangulations

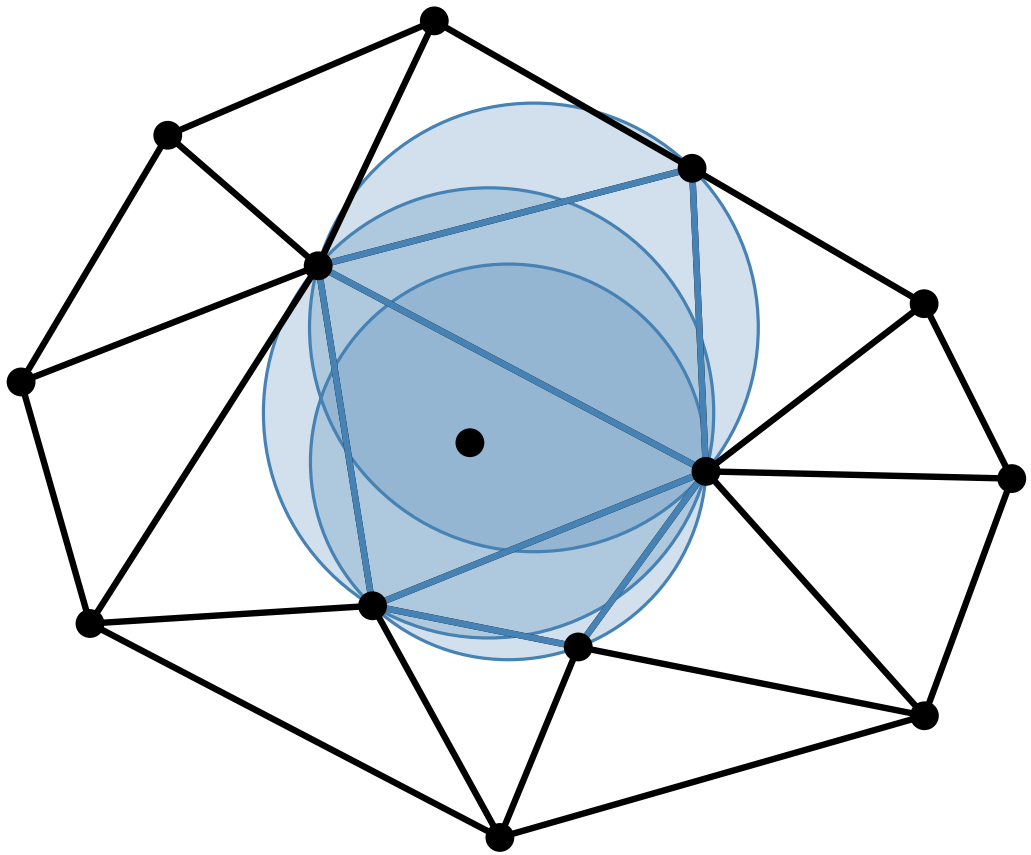
Euclidean Delaunay Triangulations

Bowyer's incremental algorithm



Euclidean Delaunay Triangulations

Bowyer's incremental algorithm



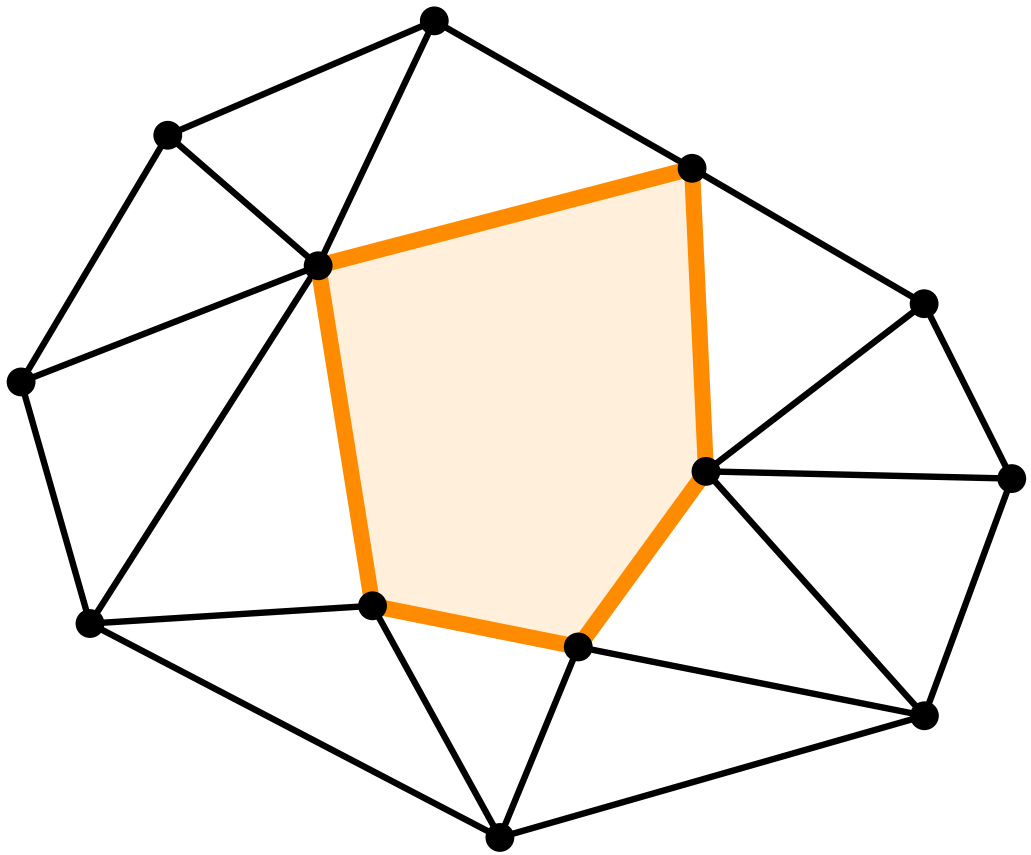
find simplices in conflict

`Triangulation`

using `Geom_traits`

Euclidean Delaunay Triangulations

Bowyer's incremental algorithm

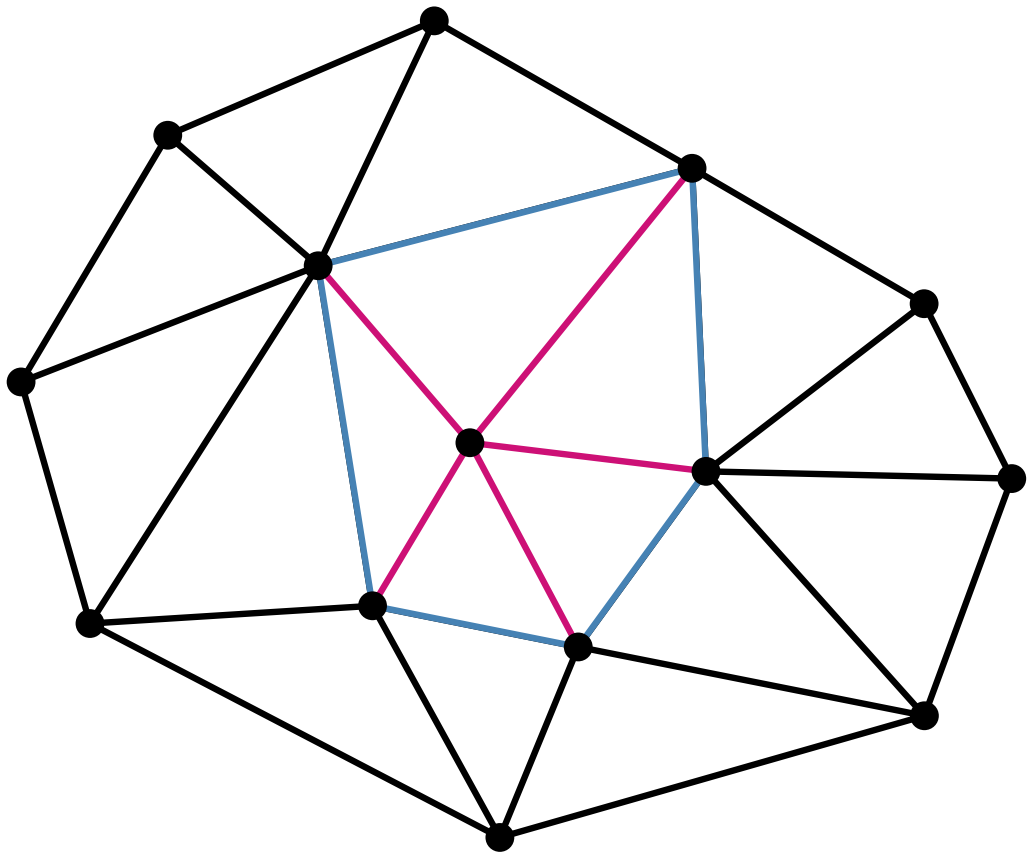


find simplices in conflict
`Triangulation`
using `Geom_traits`

the conflict region forms a topological ball

Euclidean Delaunay Triangulations

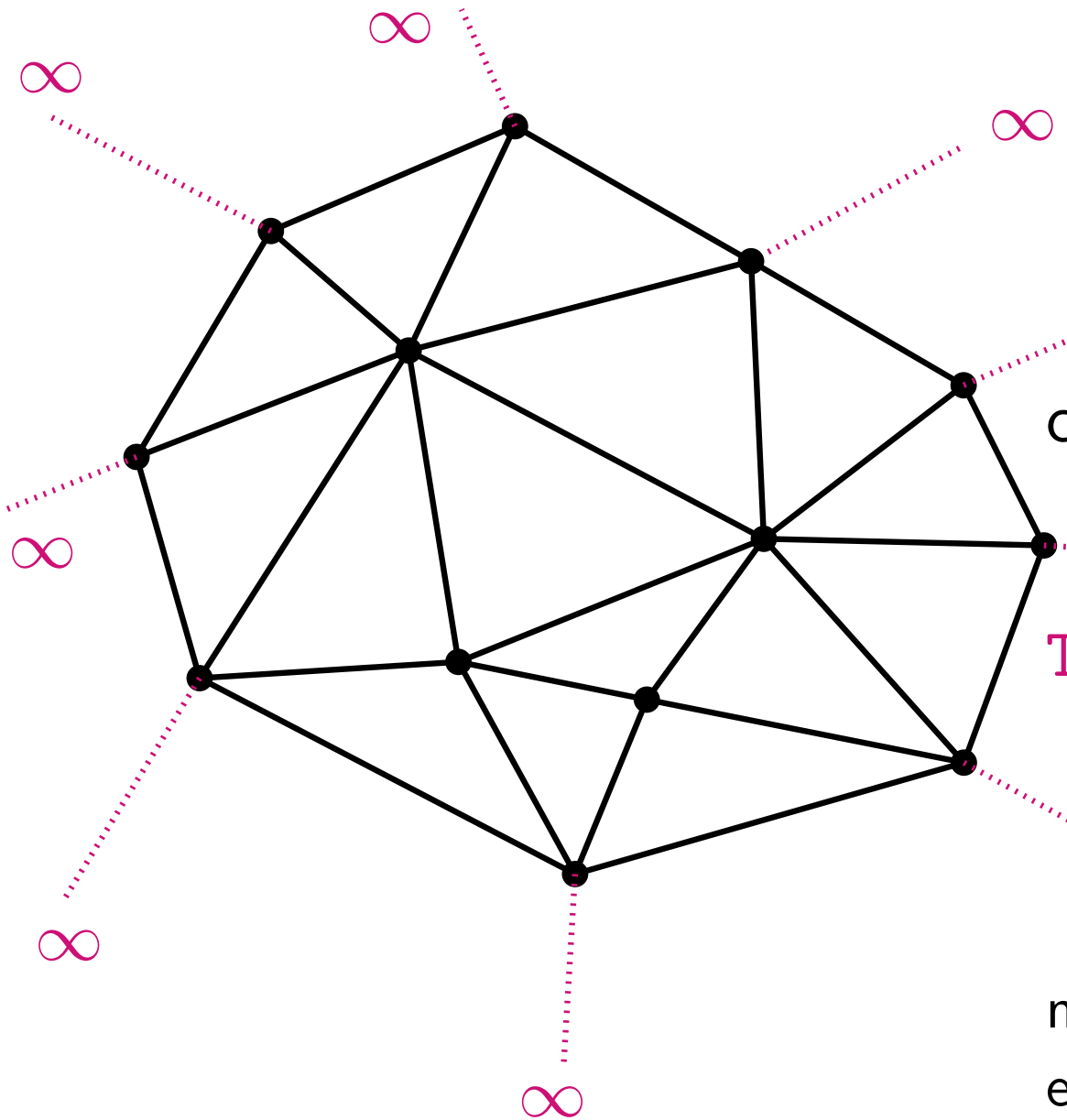
Bowyer's incremental algorithm



find simplices in conflict
`Triangulation`
using `Geom_traits`

create new simplices
`TDS`

Euclidean Delaunay Triangulations



compactification of $\mathbb{R}^d$

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

no boundary

each cell = d-simplex

Euclidean Delaunay Triangulations

Robustness

Arithmetic issues

Exact Geometric Computation paradigm [Yap]

exact predicates $\neq$ exact arithmetics

Filtering

easy cases are more frequent

$\Longrightarrow$ cost $\simeq$ cost of approximate (double) computation

Euclidean Delaunay Triangulations

Robustness

Arithmetic issues

Approximate evaluation $P^a(x)$
$+$ Error ϵ

Filtering

$P^a(x) > \epsilon$

?

Yes

No

Sign$(P(x))$ = Sign$(P^a(x))$

Exact computation

Euclidean Delaunay Triangulations

Robustness

Degenerate cases

symbolic perturbation

only perturbs the `in_sphere` predicate

[Devillers, T. SODA'03, CGTA'11]

Euclidean Delaunay Triangulations

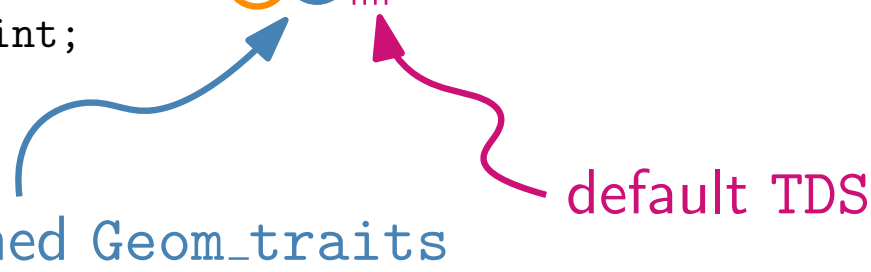
Tricks to improve efficiency

EuroCG'12 invited talk by Olivier Devillers

`https://hal.inria.fr/hal-00850561`

Euclidean Delaunay Triangulations

```
#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

Flexibility

```
#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

Euclidean Delaunay Triangulations

Flexibility

```
#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;
}
```

Euclidean Delaunay Triangulations

fully dynamic

fully robust

also weighted

2D (insertion with flips?)

$\simeq$ 10 M points / second

[Yvinec CGAL'97]

also constrained

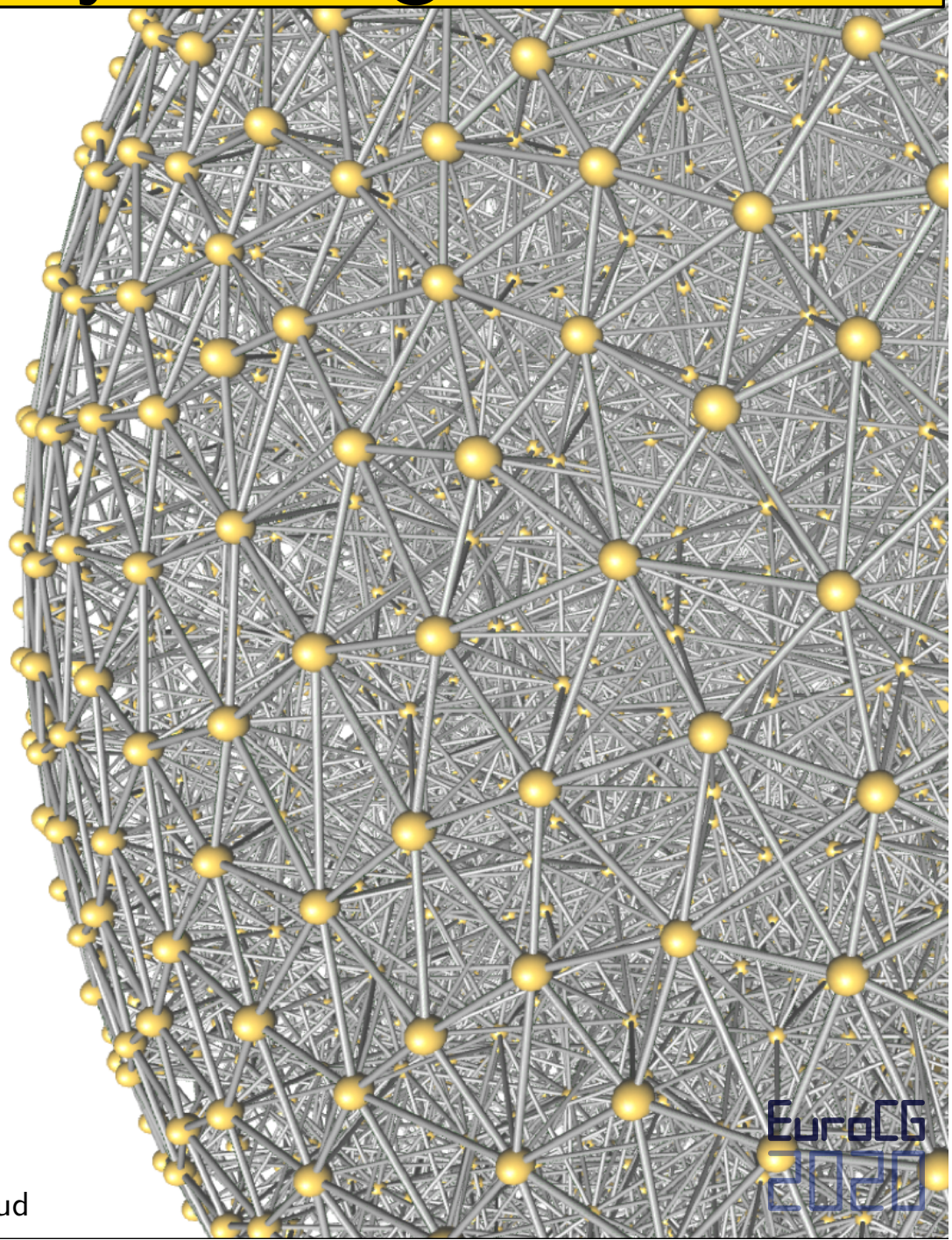
3D

$\simeq$ 1 M points / second

[T. CGAL'00] [Pion, T. CGAL'01*]

multicore version

[Jamin CGAL'14]



Hyperbolic triangulations

Hyperbolic triangulations

[Boissonnat 1988]

Once upon a time...

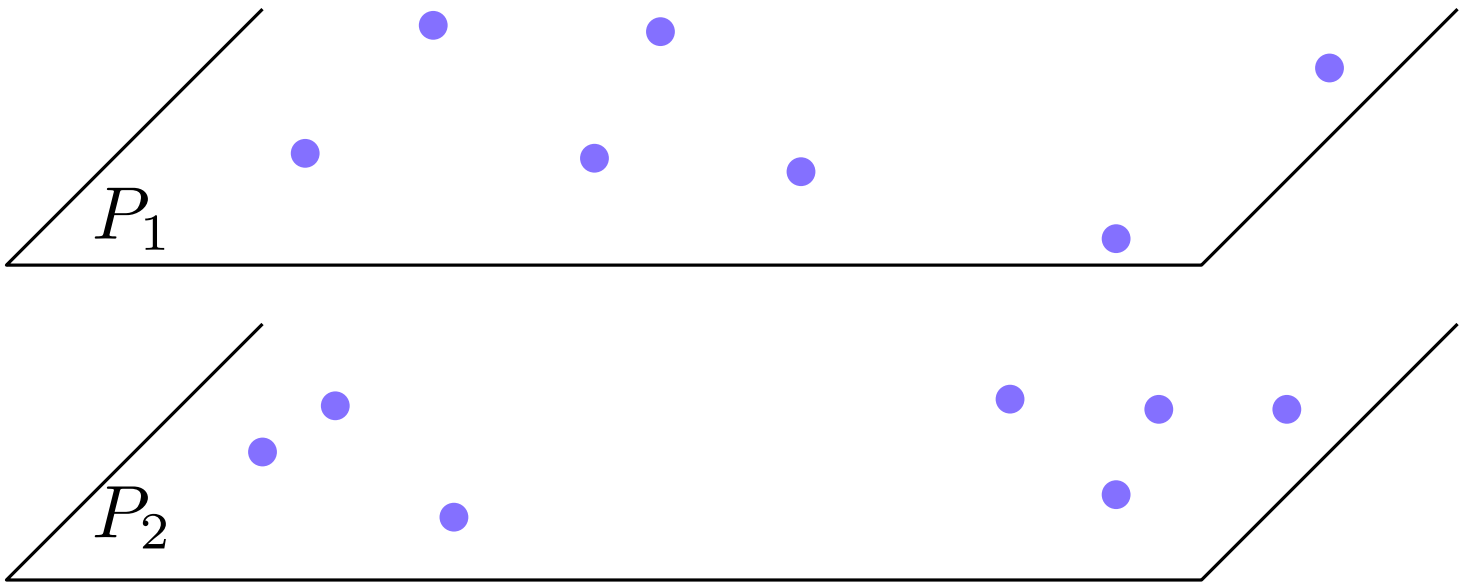
Compute the 3D Delaunay triangulation

of a set of points

lying in two parallel planes

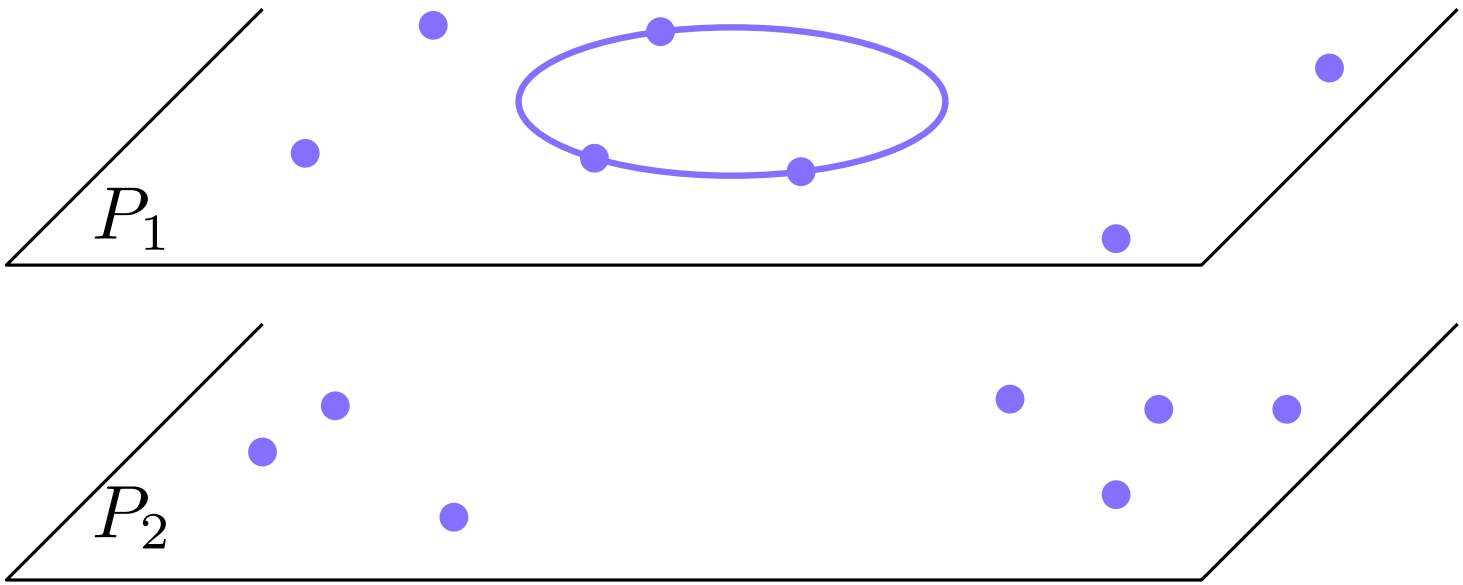
Hyperbolic triangulations

[Boissonnat 1988]



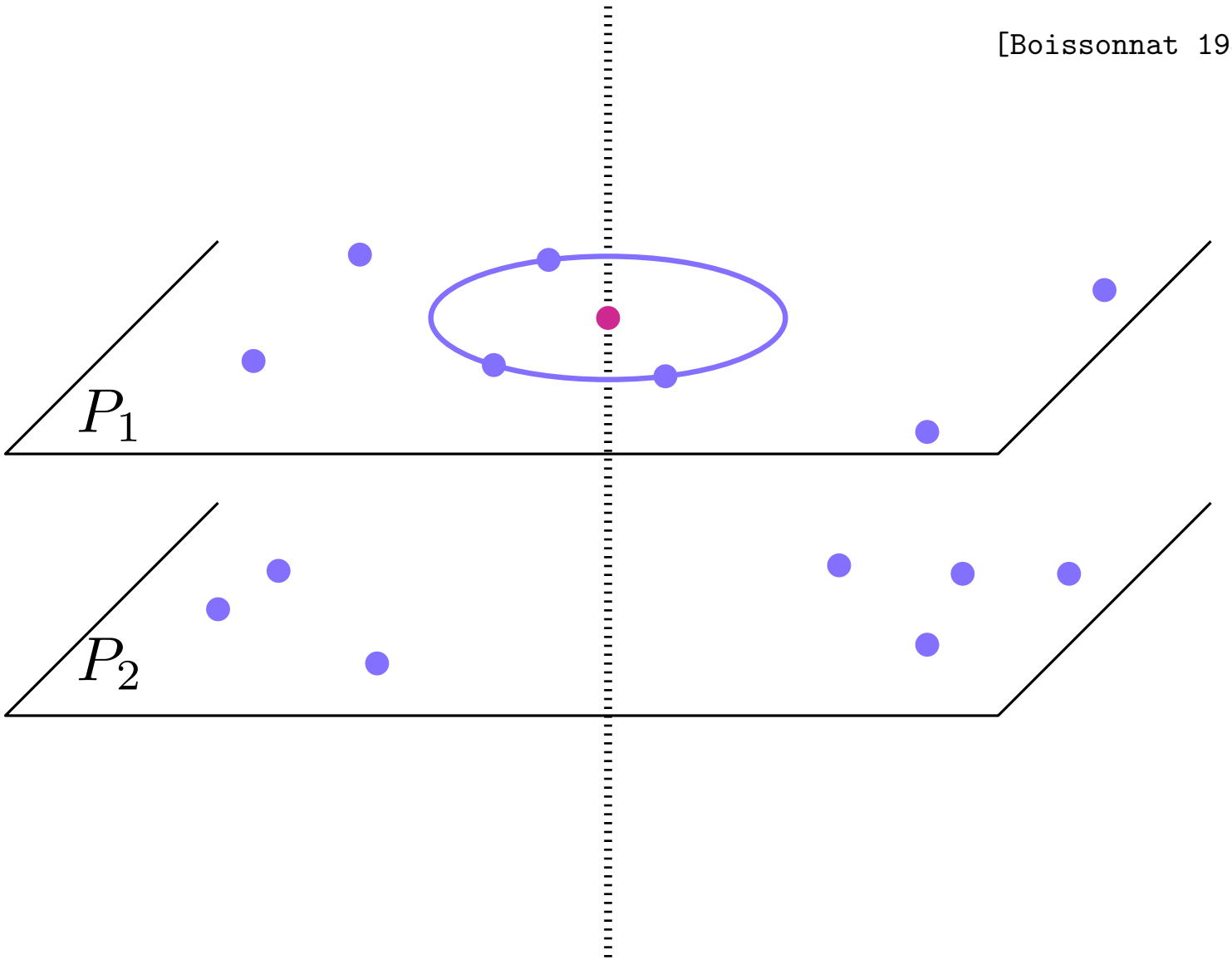
Hyperbolic triangulations

[Boissonnat 1988]



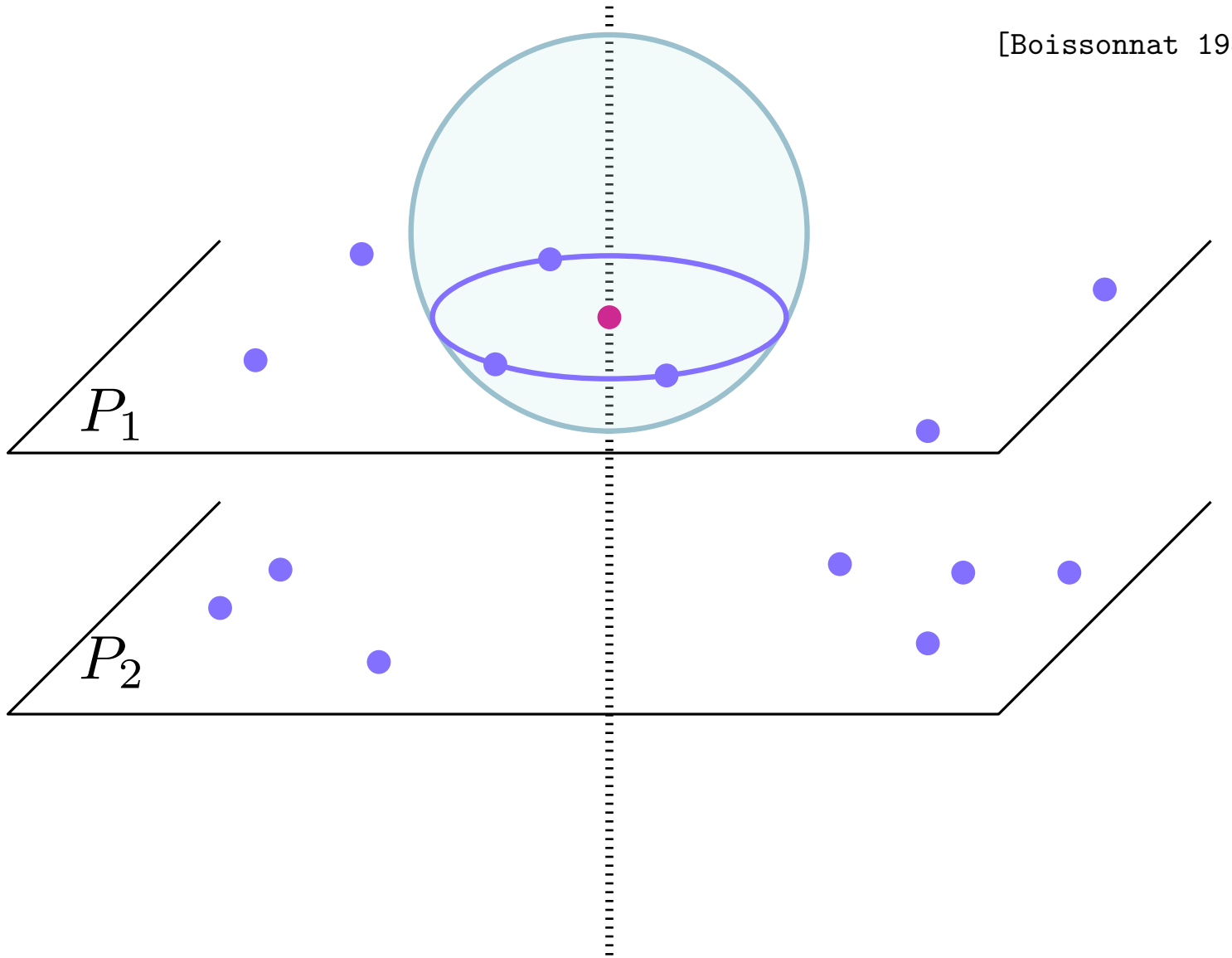
Hyperbolic triangulations

[Boissonnat 1988]



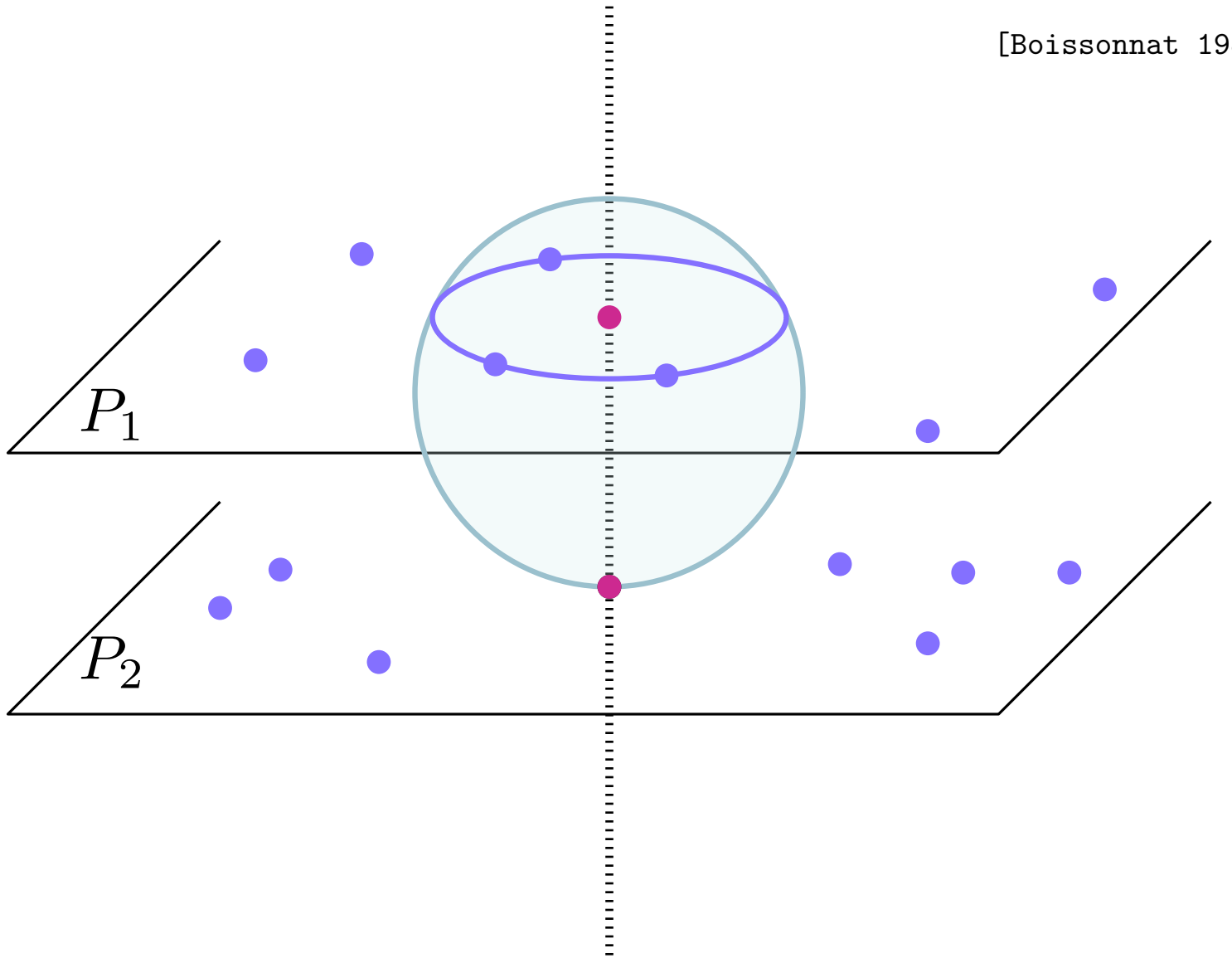
Hyperbolic triangulations

[Boissonnat 1988]



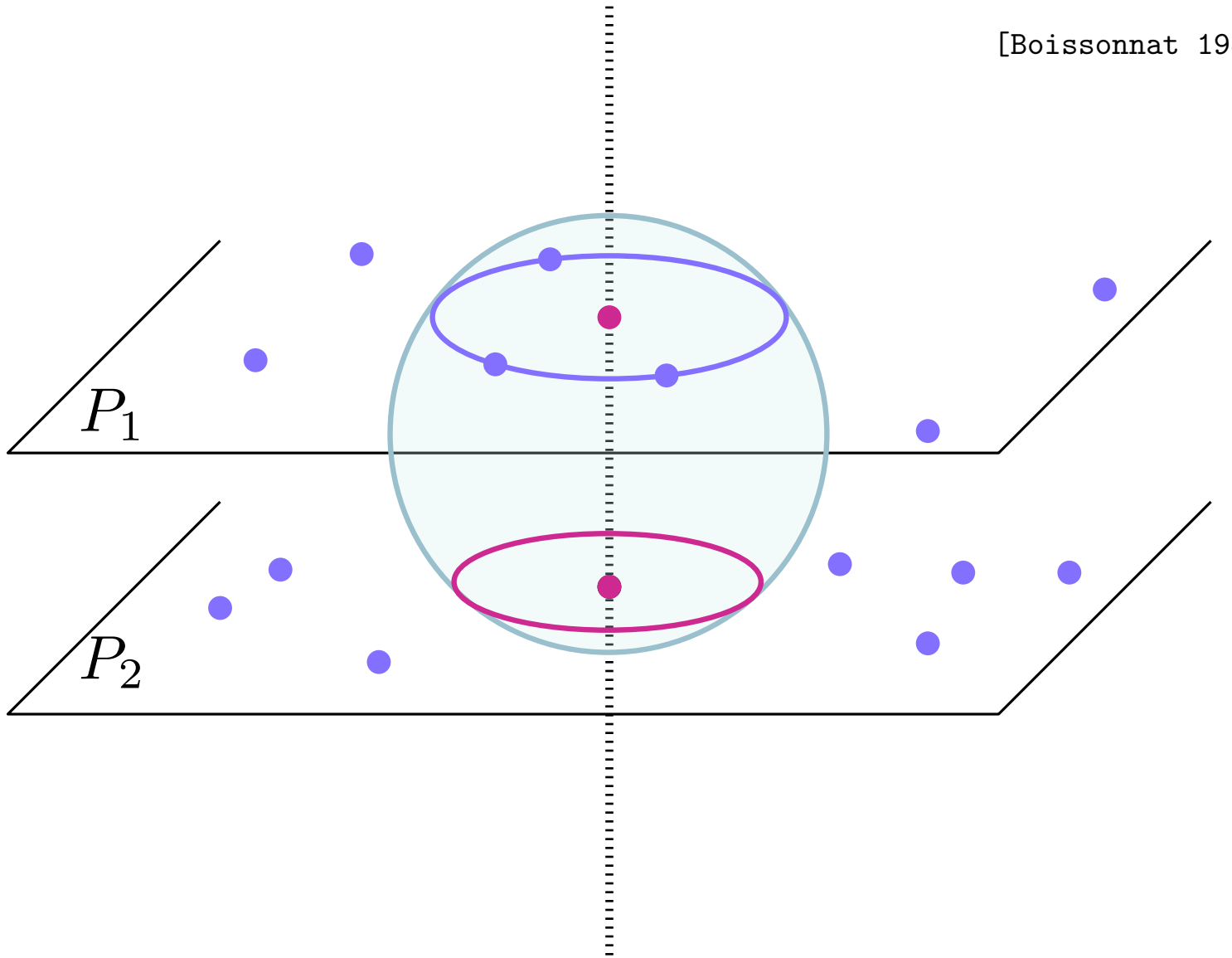
Hyperbolic triangulations

[Boissonnat 1988]



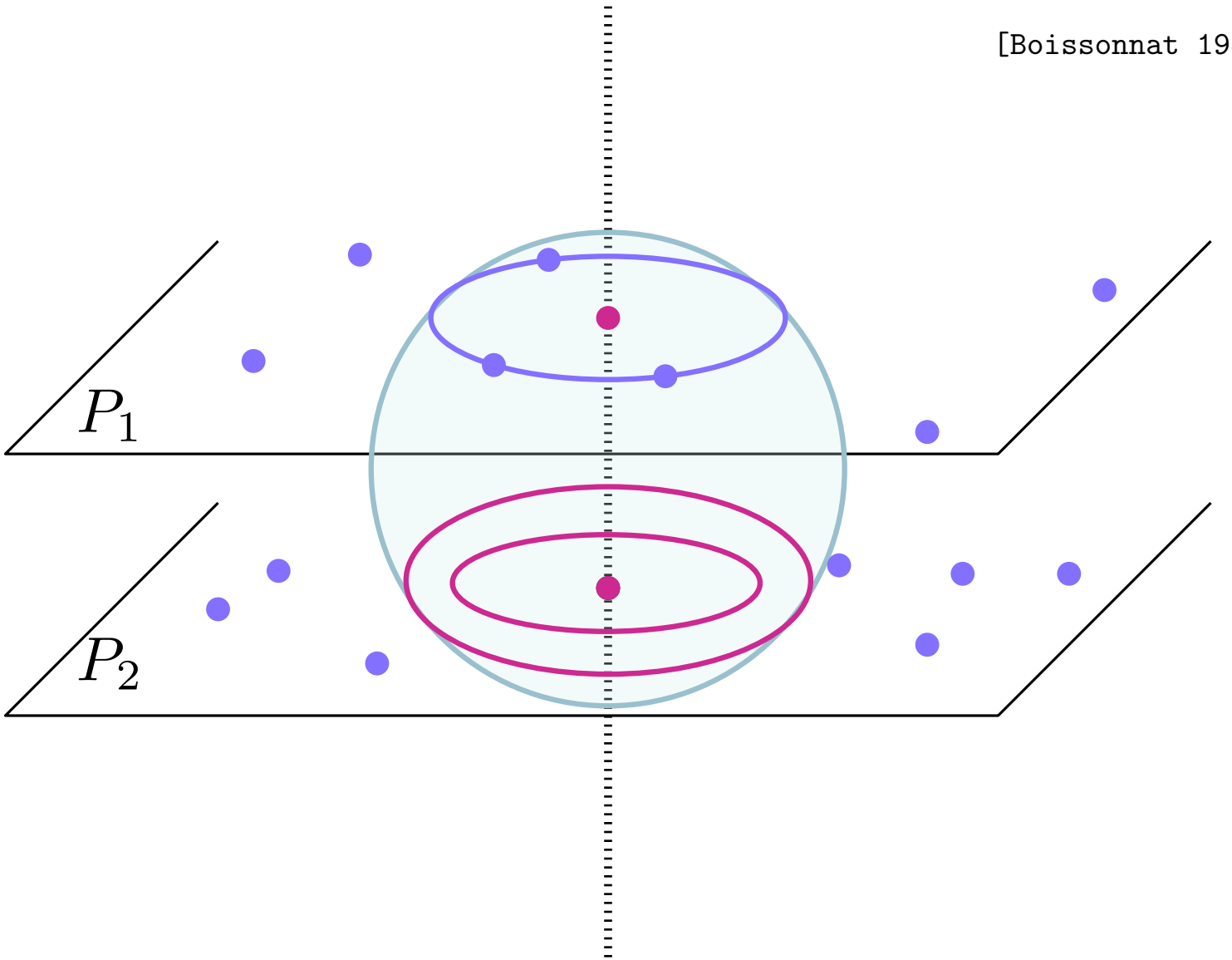
Hyperbolic triangulations

[Boissonnat 1988]



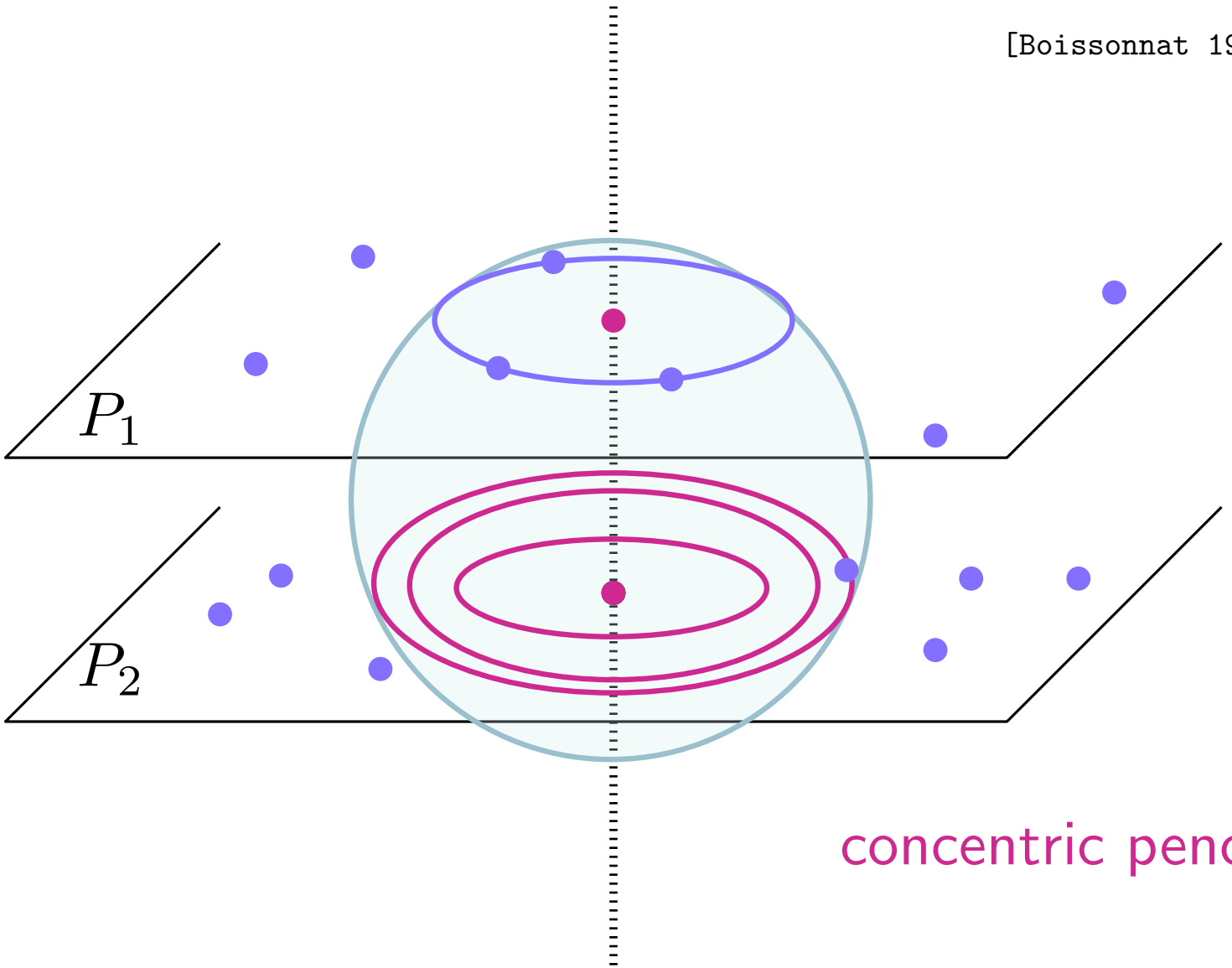
Hyperbolic triangulations

[Boissonnat 1988]



Hyperbolic triangulations

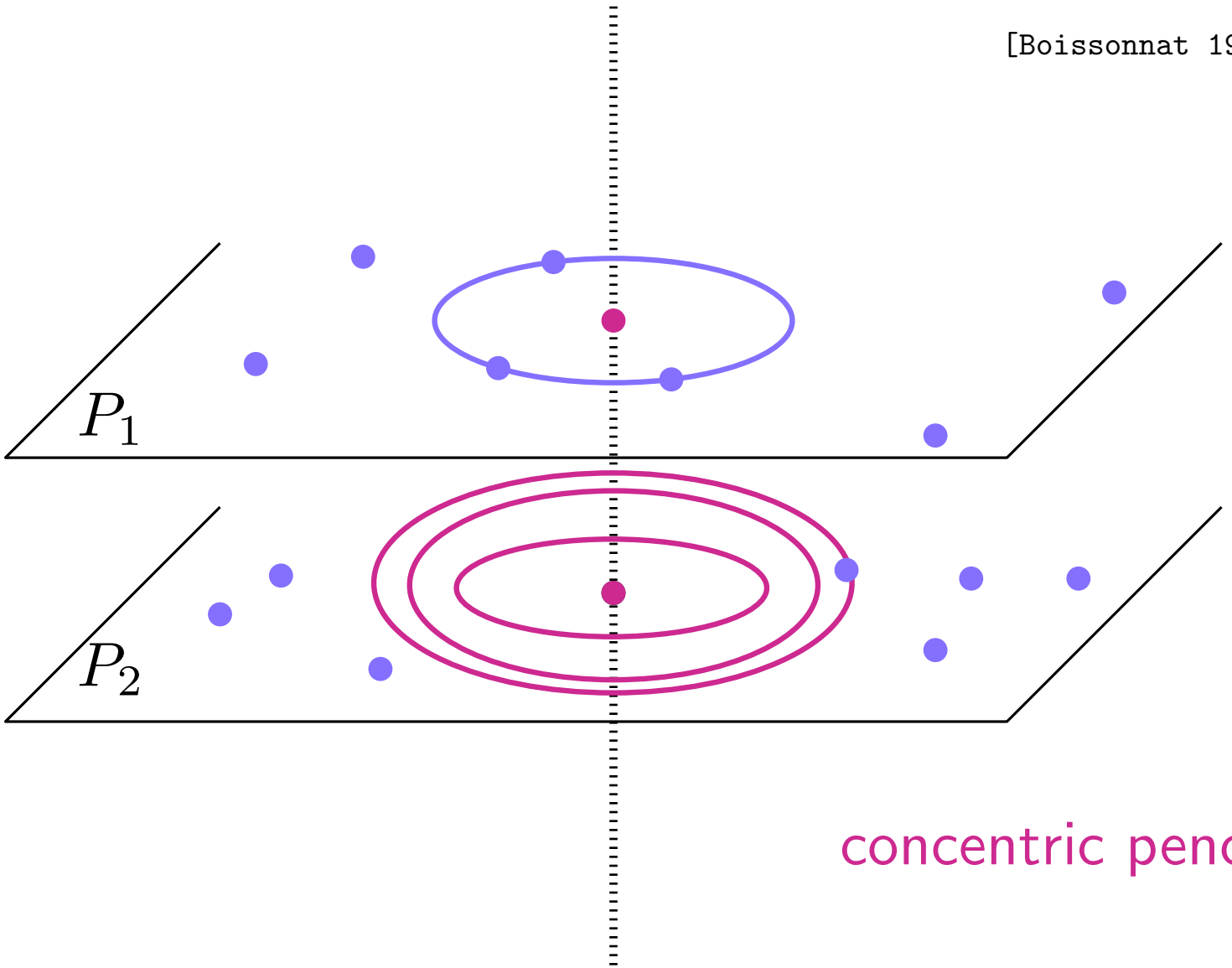
[Boissonnat 1988]



concentric pencil

Hyperbolic triangulations

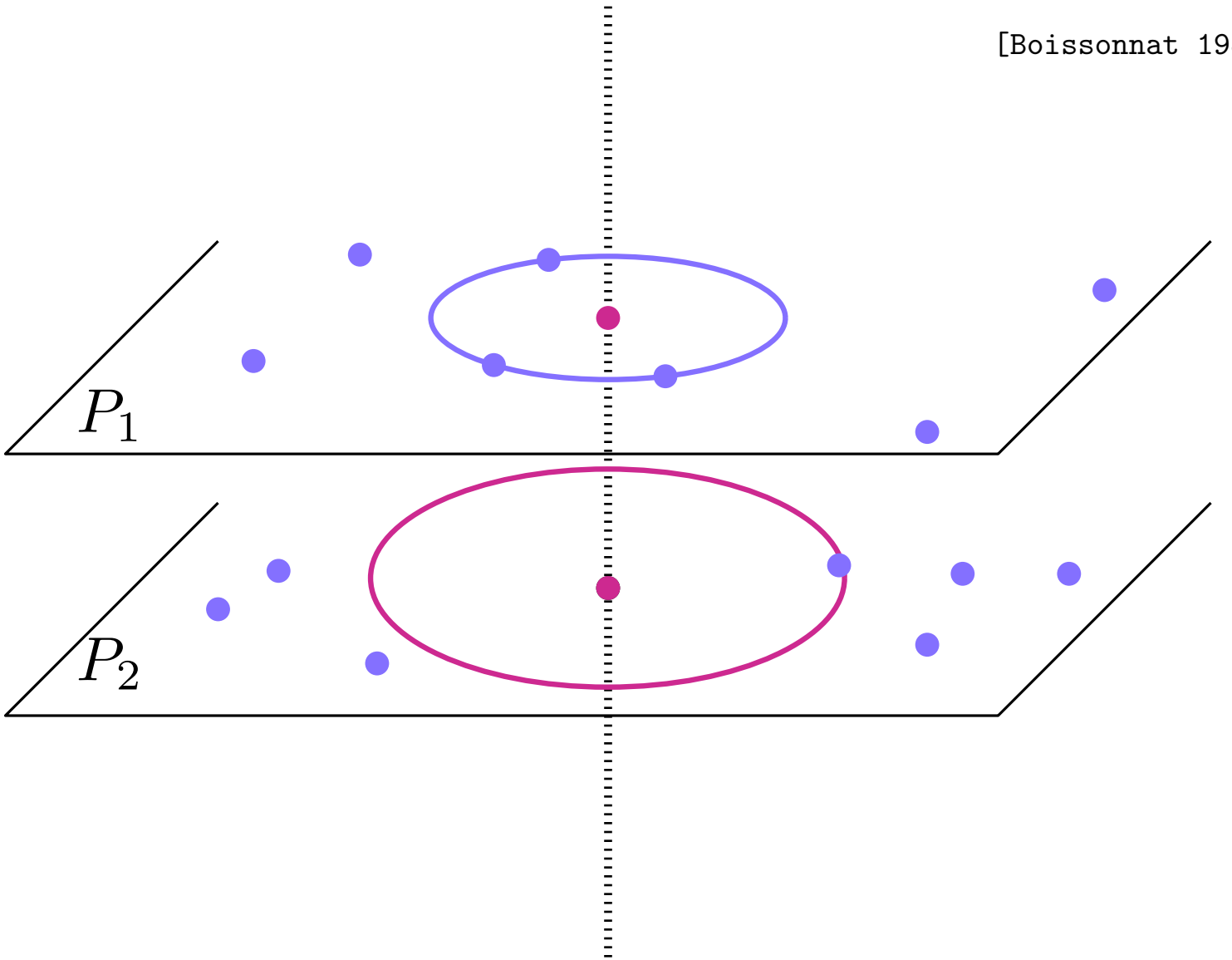
[Boissonnat 1988]



concentric pencil

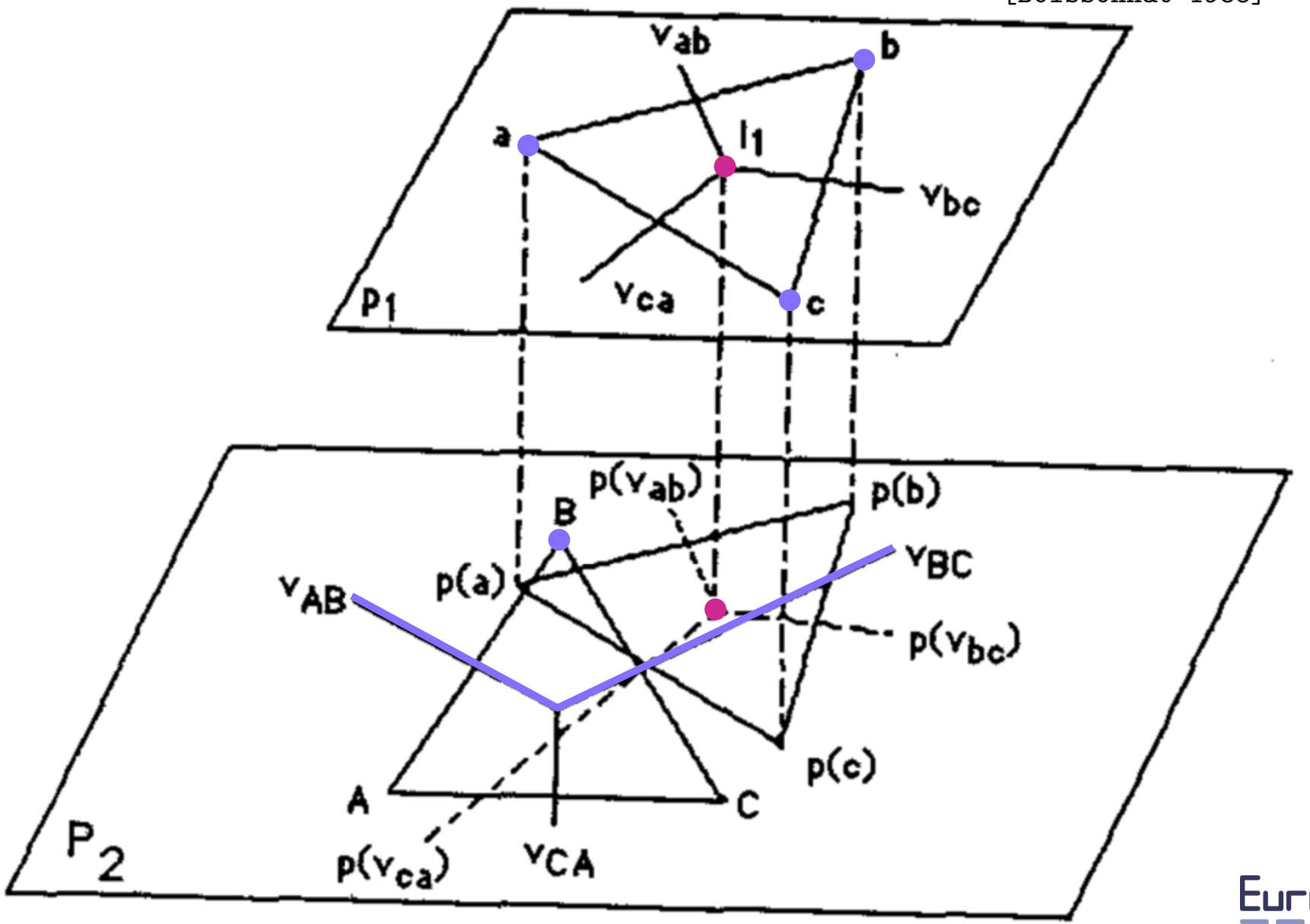
Hyperbolic triangulations

[Boissonnat 1988]



Hyperbolic triangulations

[Boissonnat 1988]



Hyperbolic triangulations

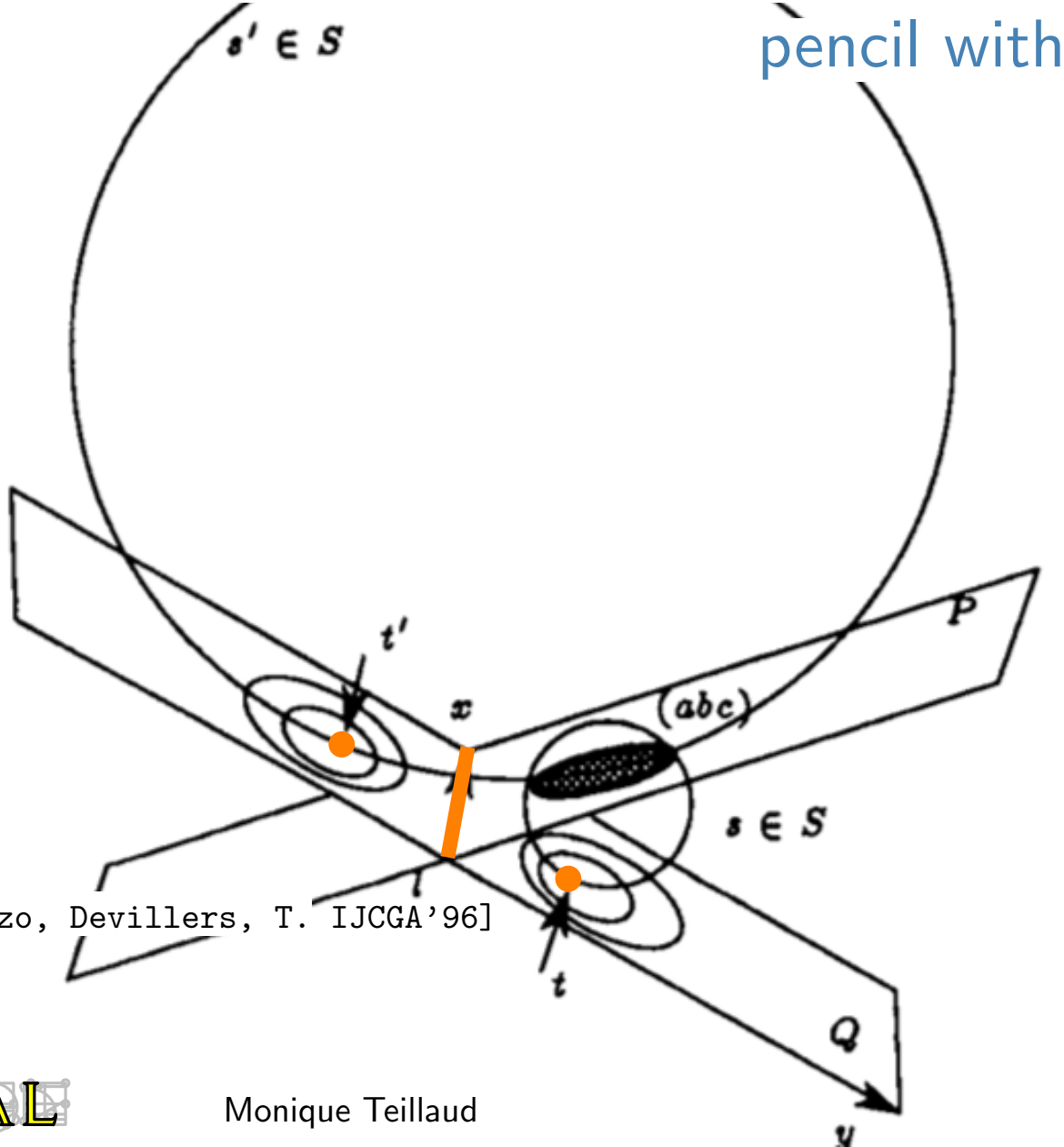
What if the planes are not parallel?

[Boissonnat, Cérézo, Devillers, T. IJCGA'96]

Hyperbolic triangulations

What if the planes are not parallel?

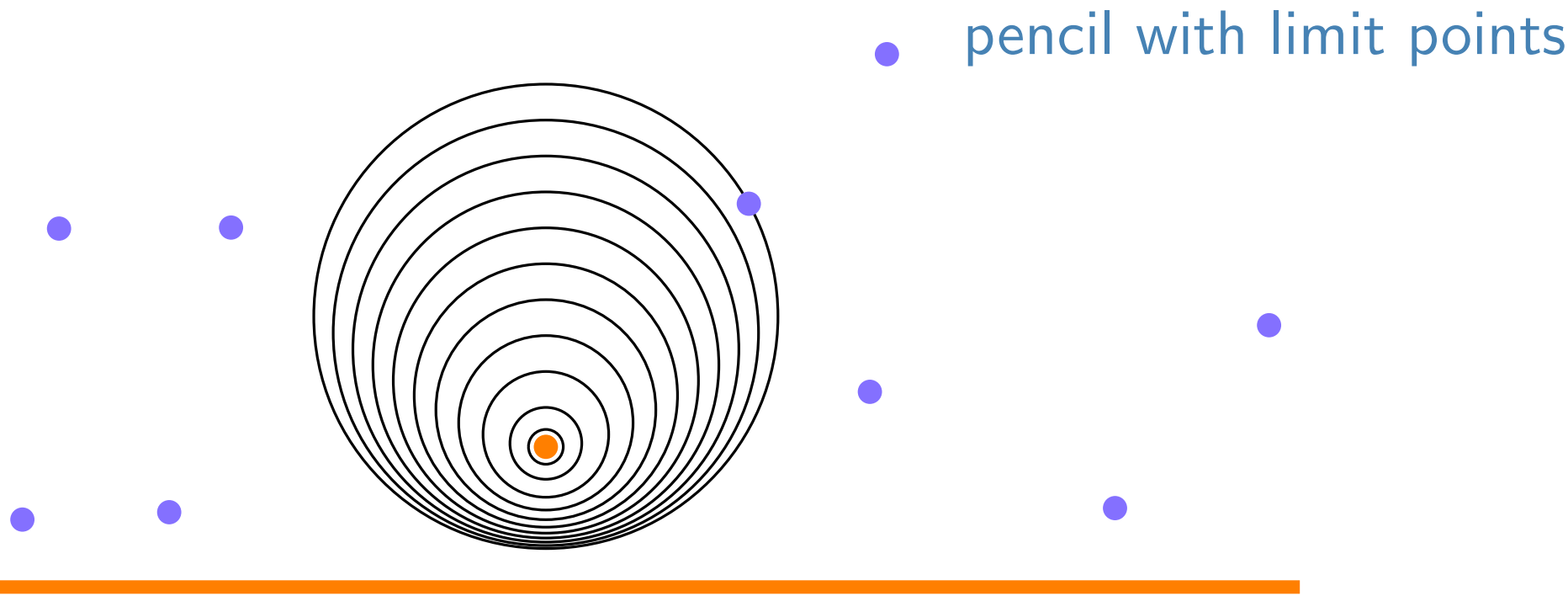
pencil with limit points



[Boissonnat, Cérézo, Devillers, T. IJCGA'96]

Hyperbolic triangulations

What if the planes are not parallel?

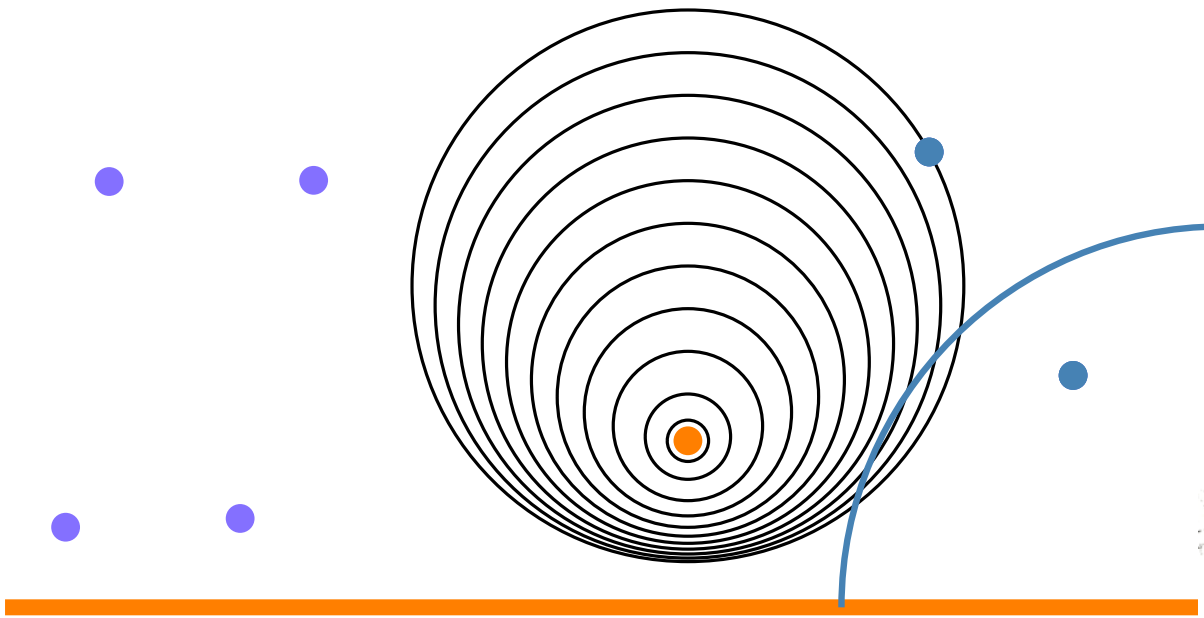


[Boissonnat, Cérézo, Devillers, T. IJCGA'96]

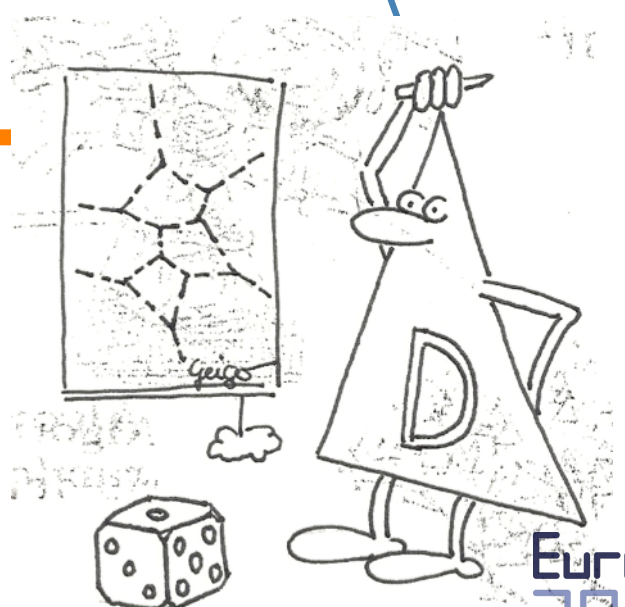
Hyperbolic triangulations

What if the planes are not parallel?

pencil with limit points

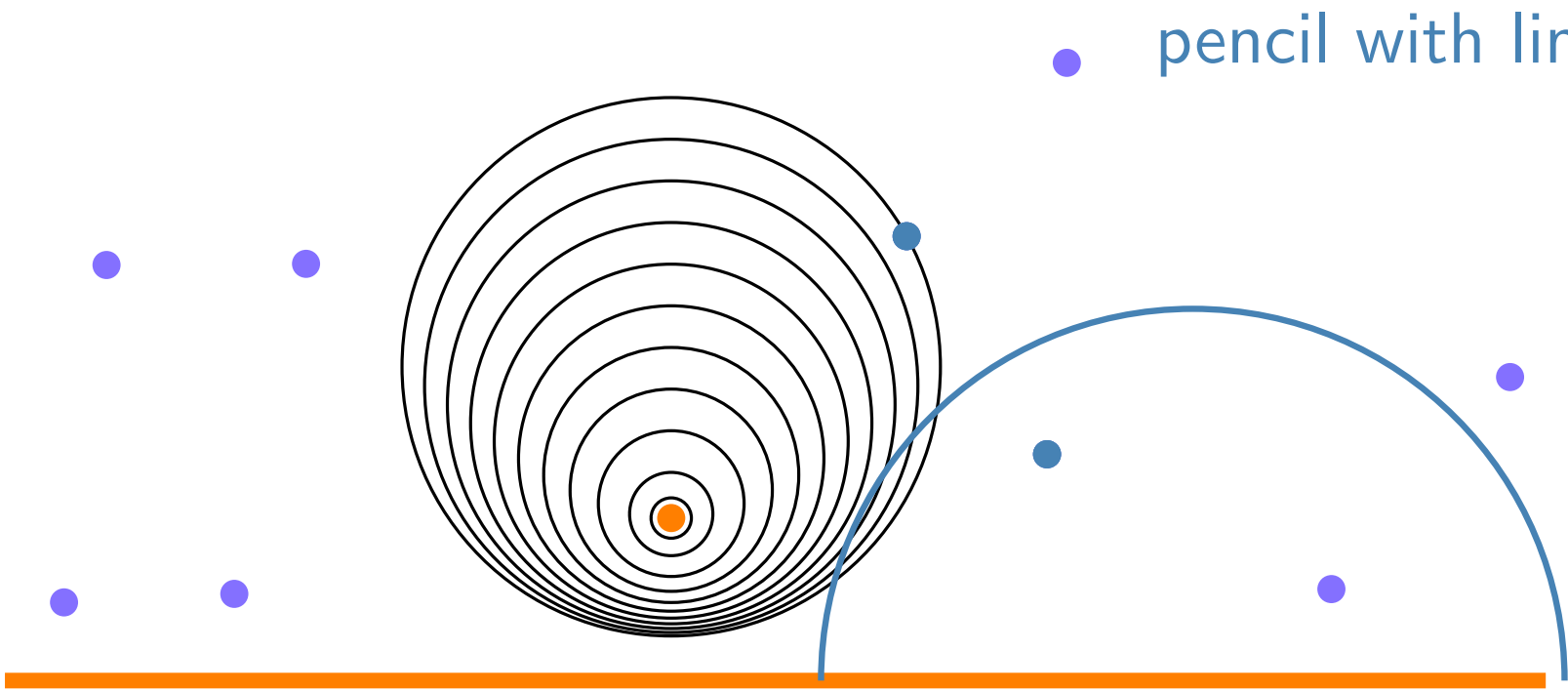


[Boissonnat, Cérézo, Devillers, T. IJCGA'96]



Hyperbolic triangulations

What if the planes are not parallel?

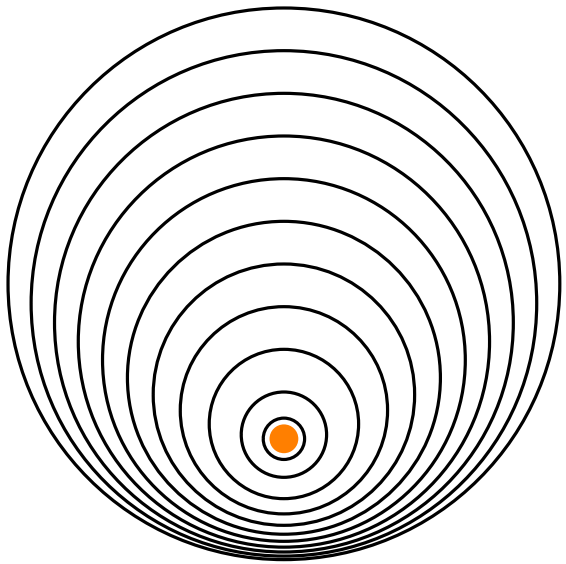


[Boissonnat, Cérézo, Devillers, T. IJCGA'96]

Hyperbolic triangulations

What if the planes are not parallel?

pencil with limit points

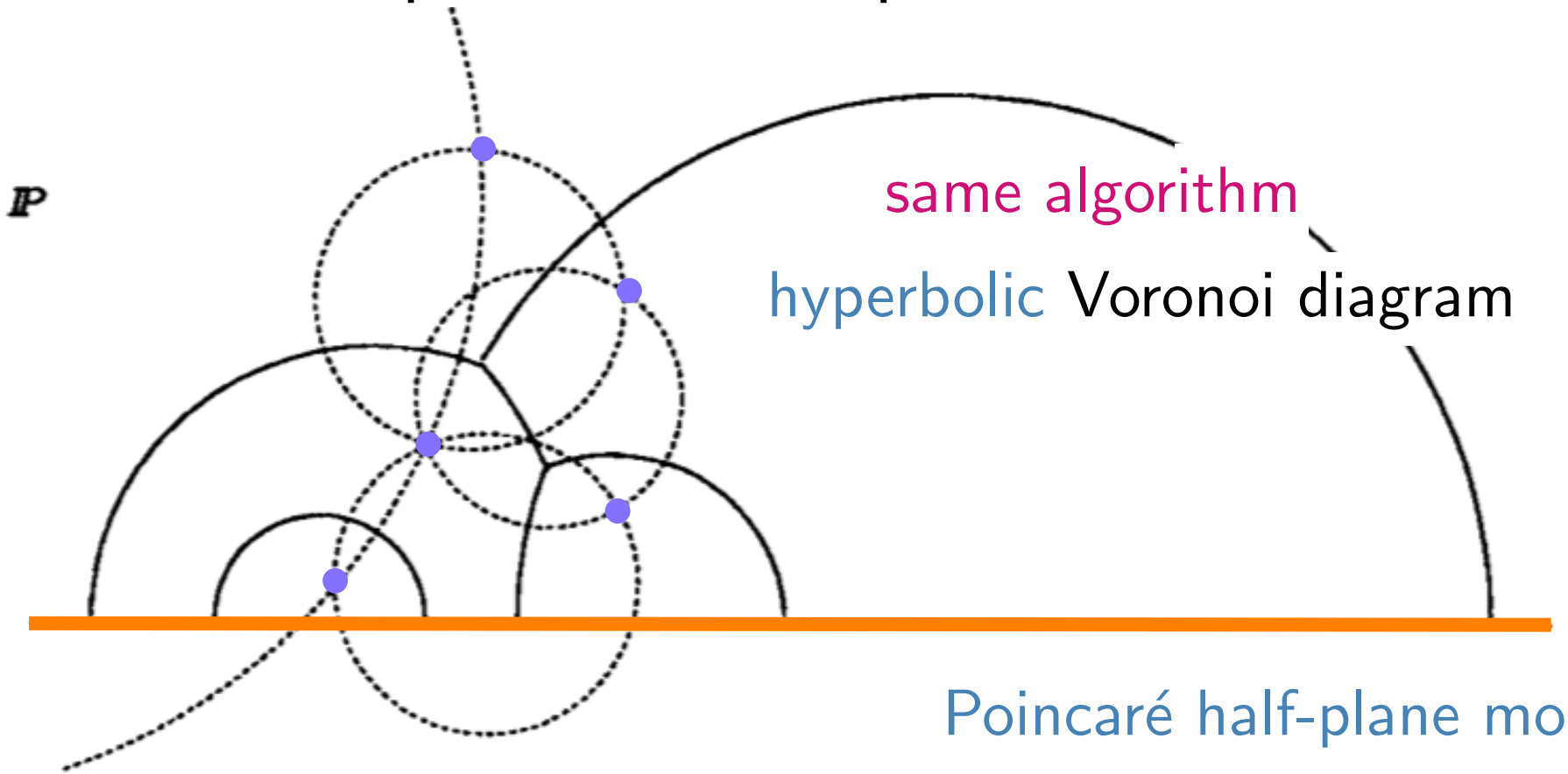


concentric pencil
in the Poincaré half-plane

[Boissonnat, Cérézo, Devillers, T. IJCGA'96]

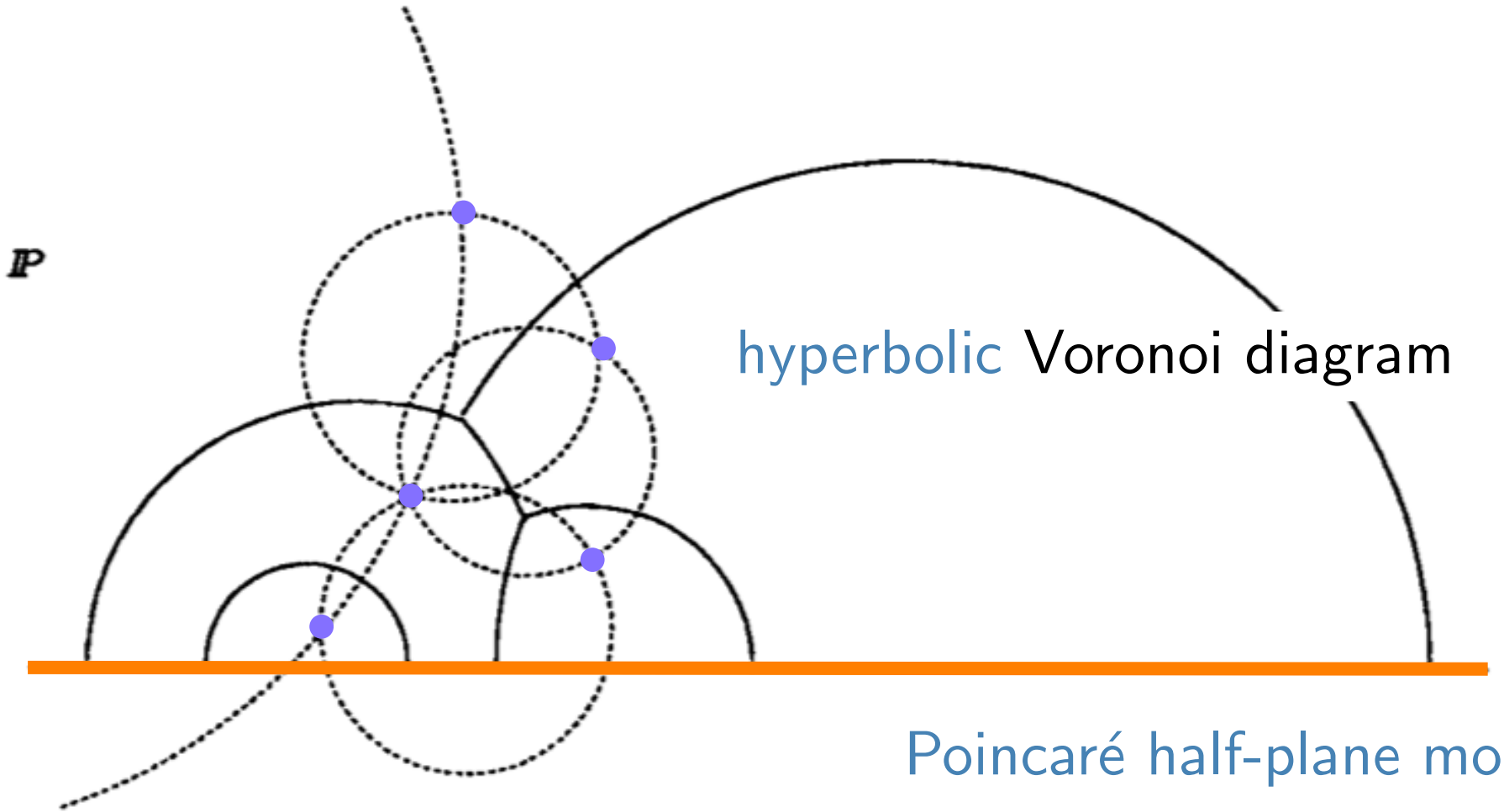
Hyperbolic triangulations

What if the planes are not parallel?



[Boissonnat, Cérézo, Devillers, T. IJCGA'96]

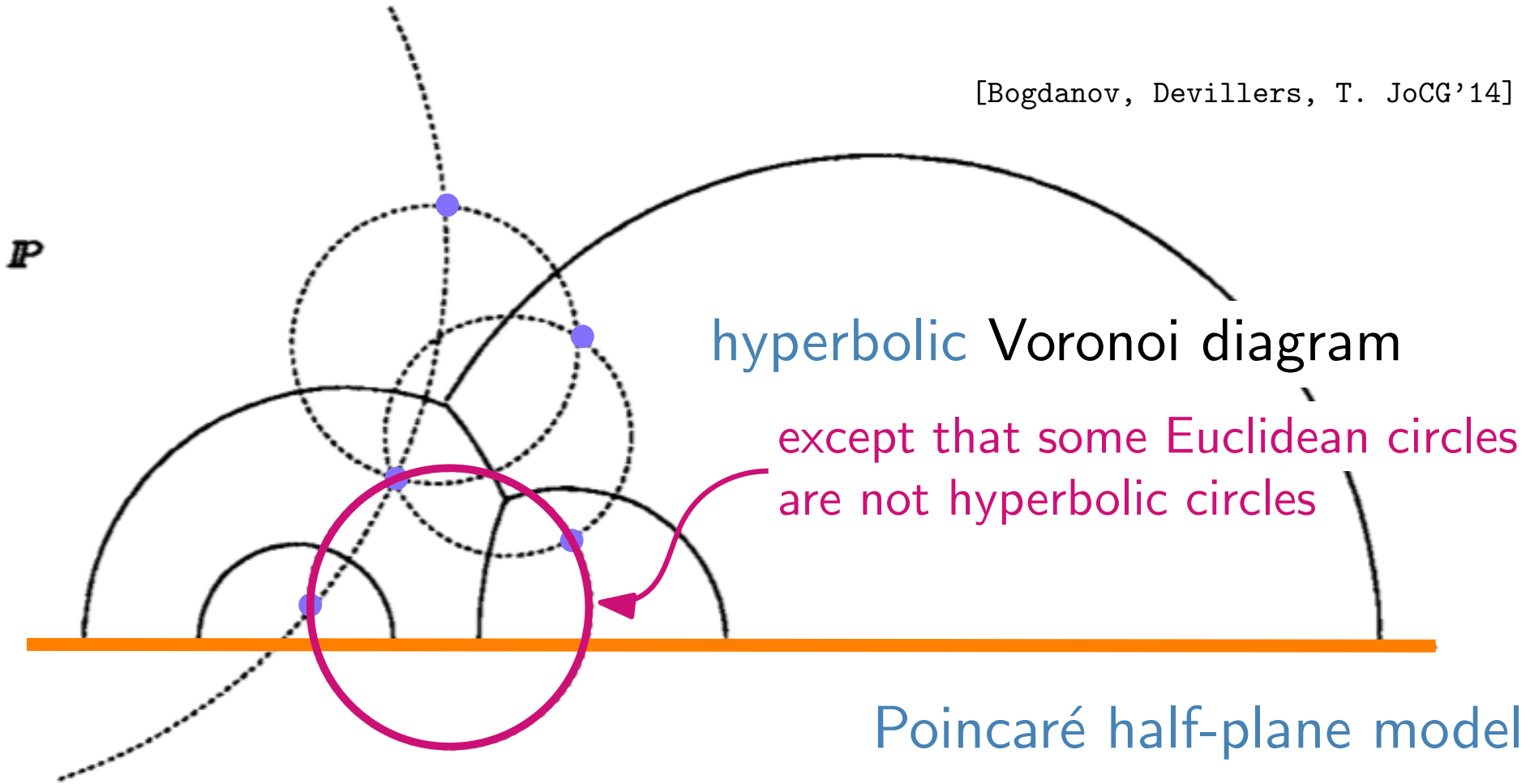
Hyperbolic triangulations



hyperbolic circles are Euclidean circles

$\mapsto$ same combinatorics as Euclidean Voronoi diagram

Hyperbolic triangulations



hyperbolic circles are Euclidean circles

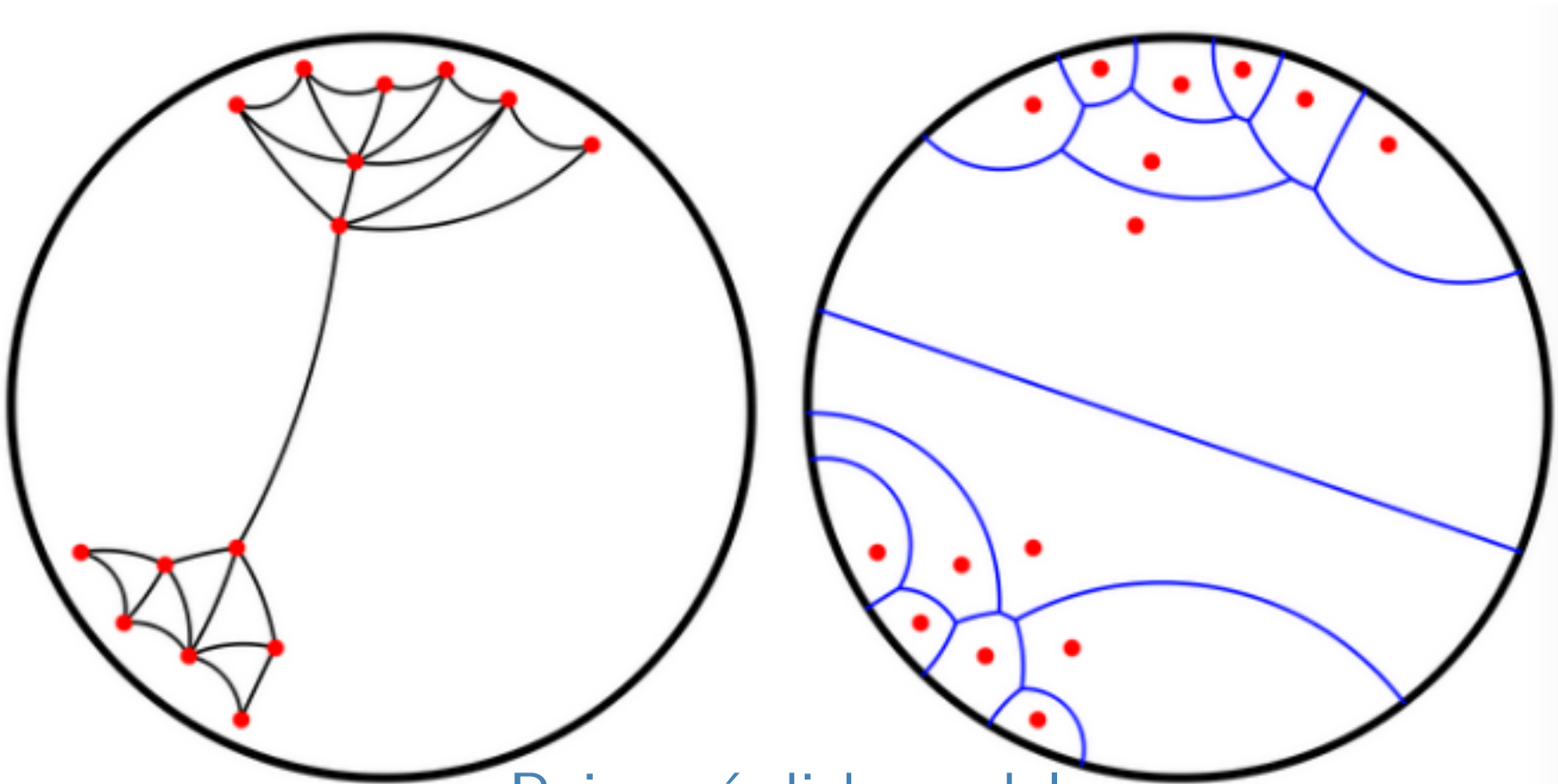
$\mapsto$ same combinatorics as Euclidean Voronoi diagram

Hyperbolic triangulations

`Hyperbolic_Delaunay_triangulation_2`
inherits from
`Delaunay_triangulation_2`

[Bogdanov, Devillers, T. JoCG'14]

[Bogdanov, Iordanov, T. CGAL'19]



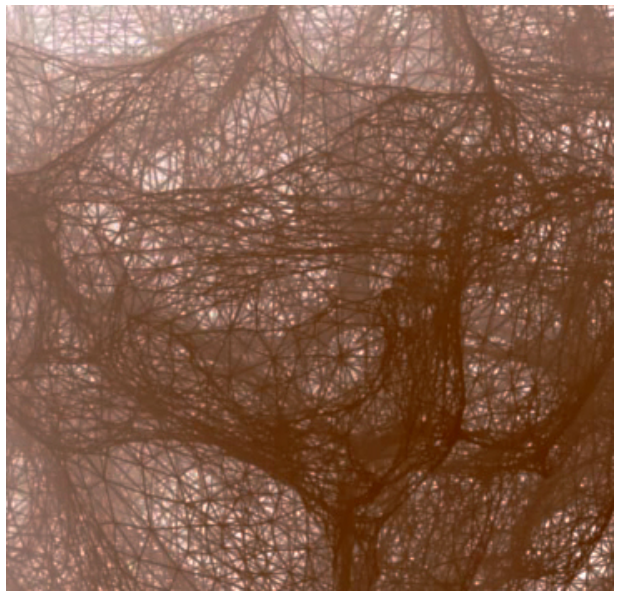
Poincaré disk model

Hyperbolic triangulations

```
#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

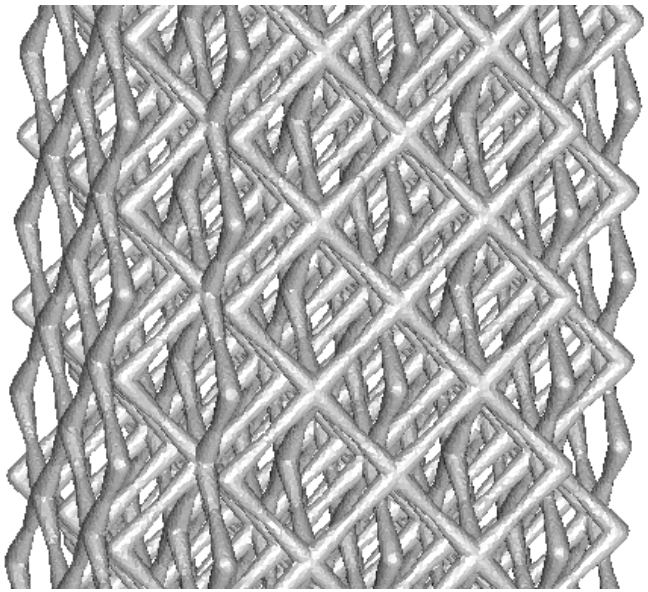
Periodic Delaunay triangulations



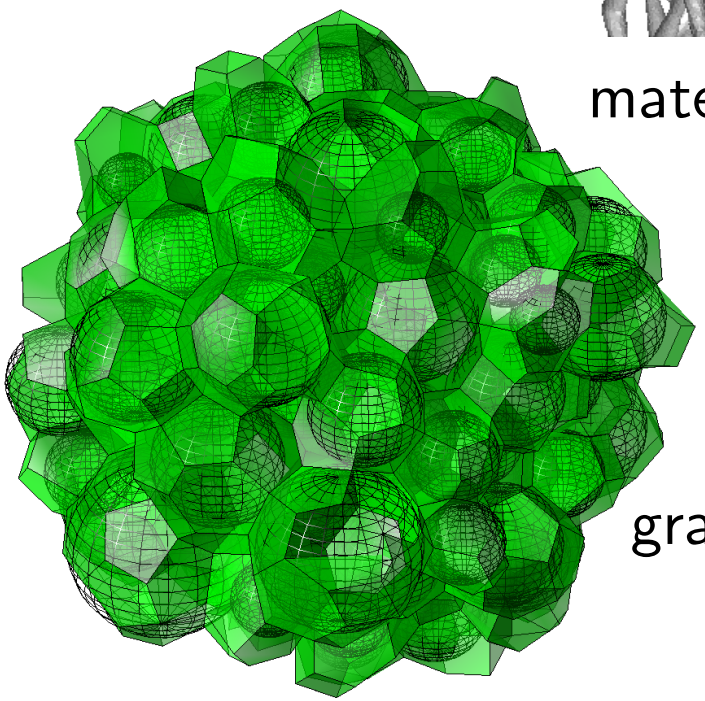
cosmic web
[v.d. Weijgaert, Groningen]

motivation:

CGAL
users

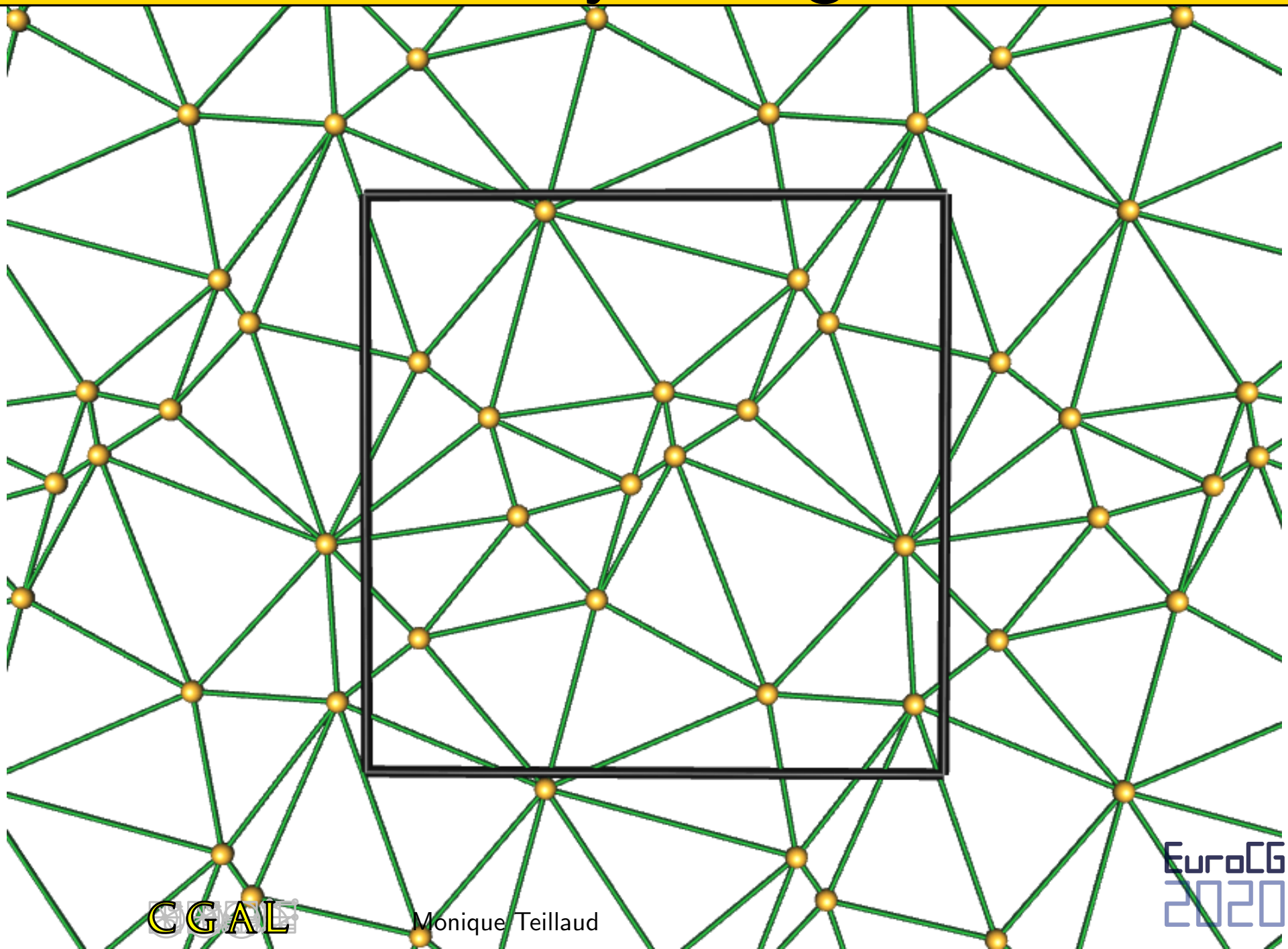


material for bone scaffold
[Moesen+, Leuven]

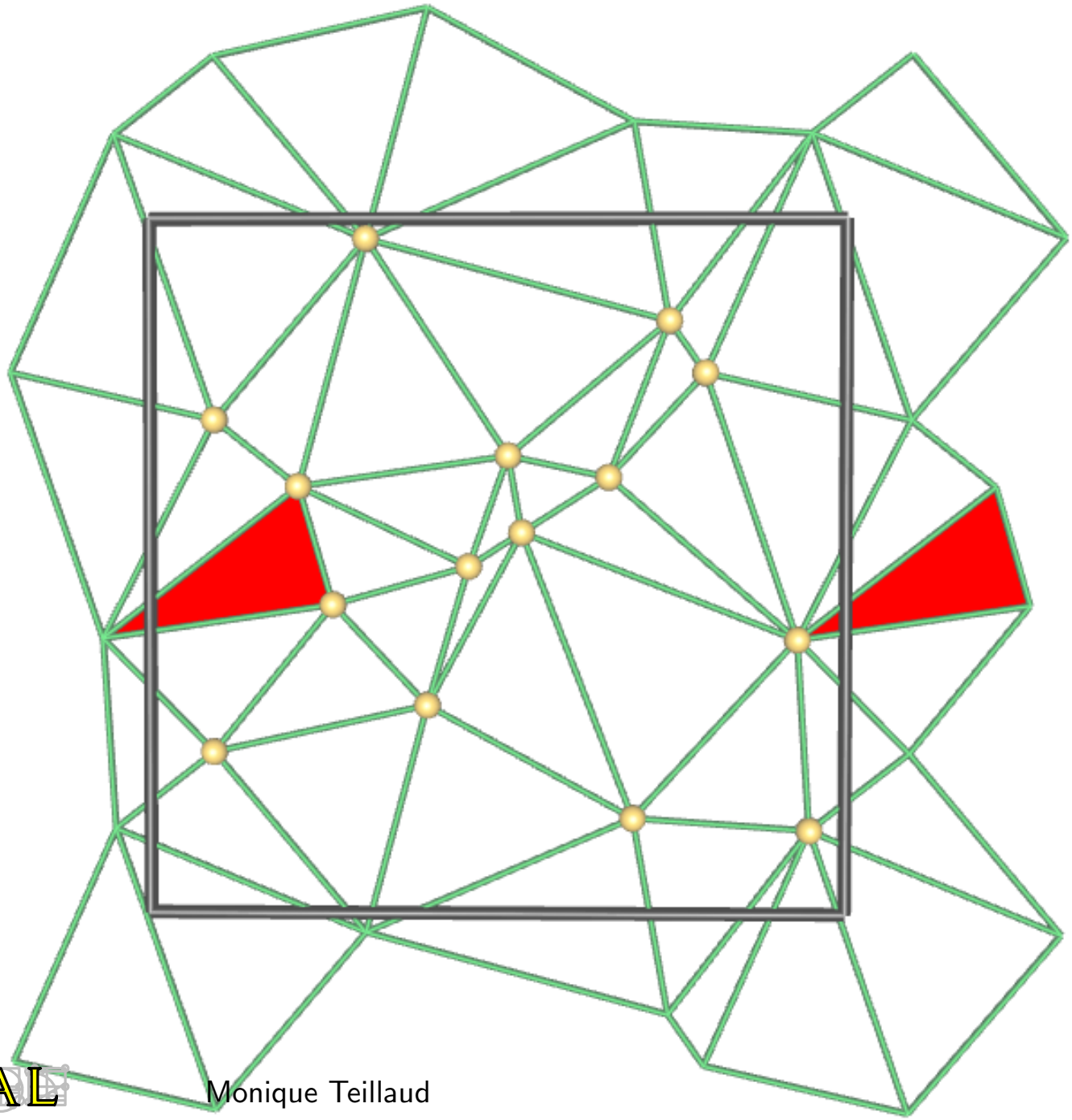


granular material
[Ludig+, Twente]

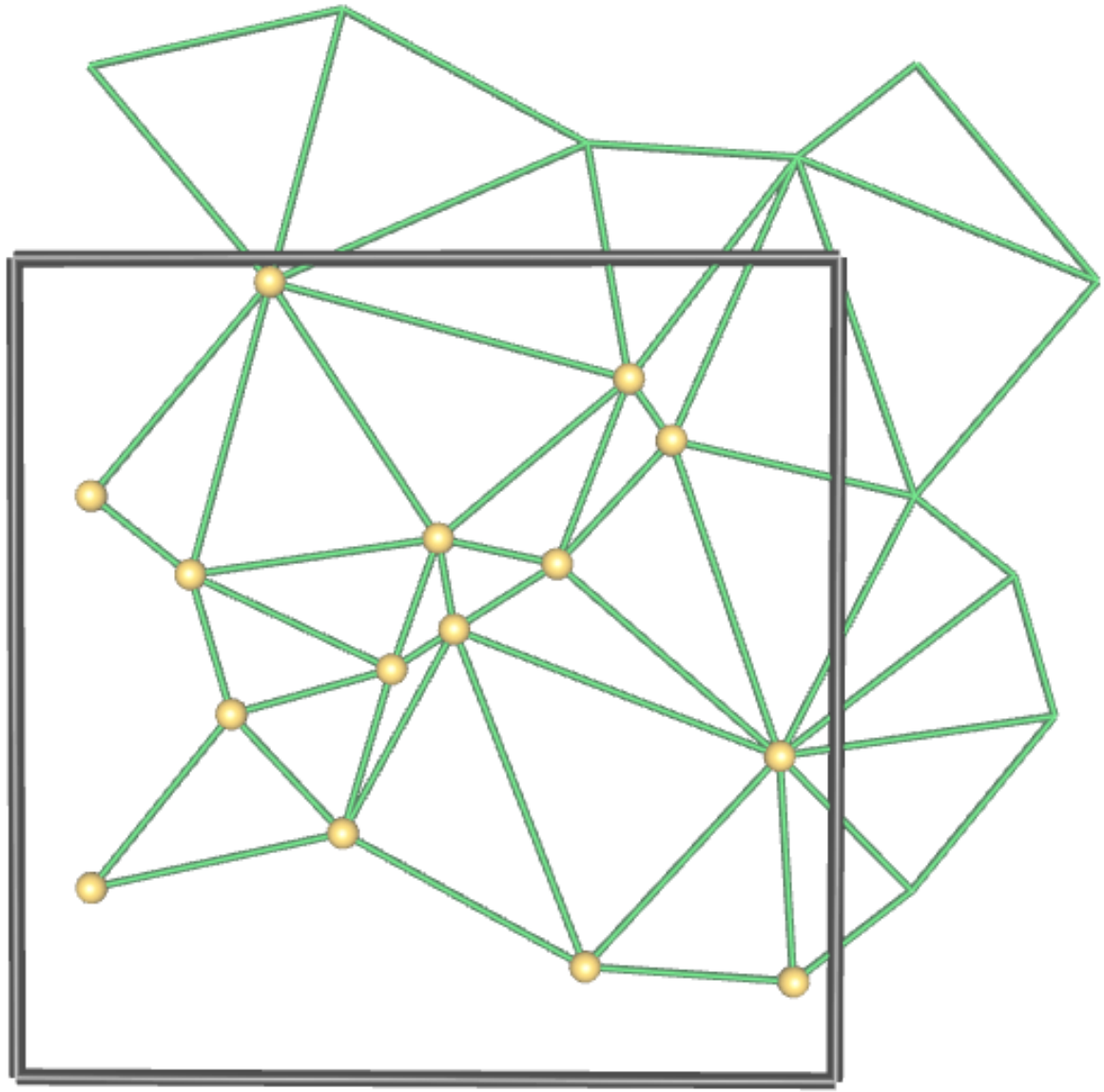
Periodic Delaunay triangulations



Periodic Delaunay triangulations



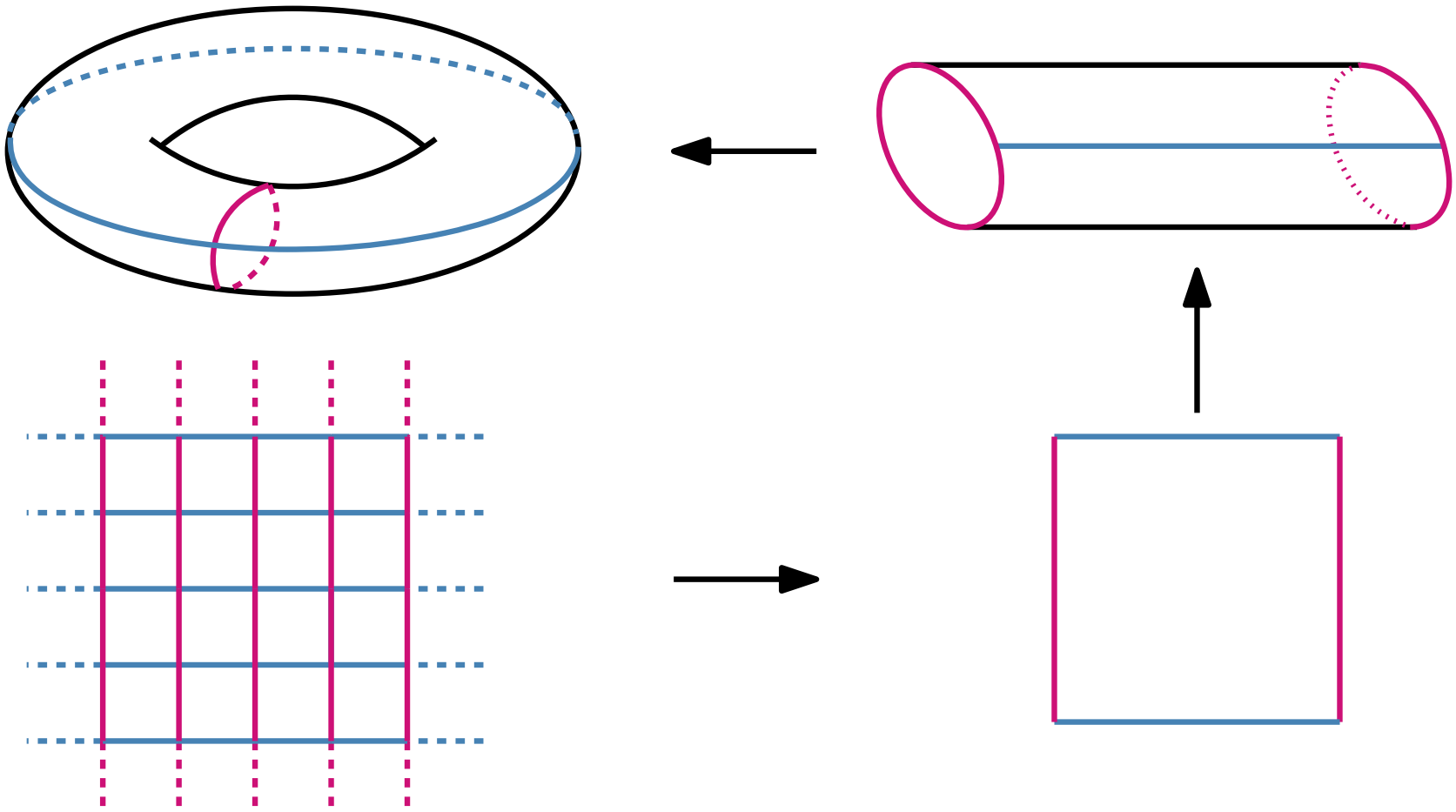
Periodic Delaunay triangulations



[Caroli, T. SoCG'08 video]

Periodic Delaunay triangulations

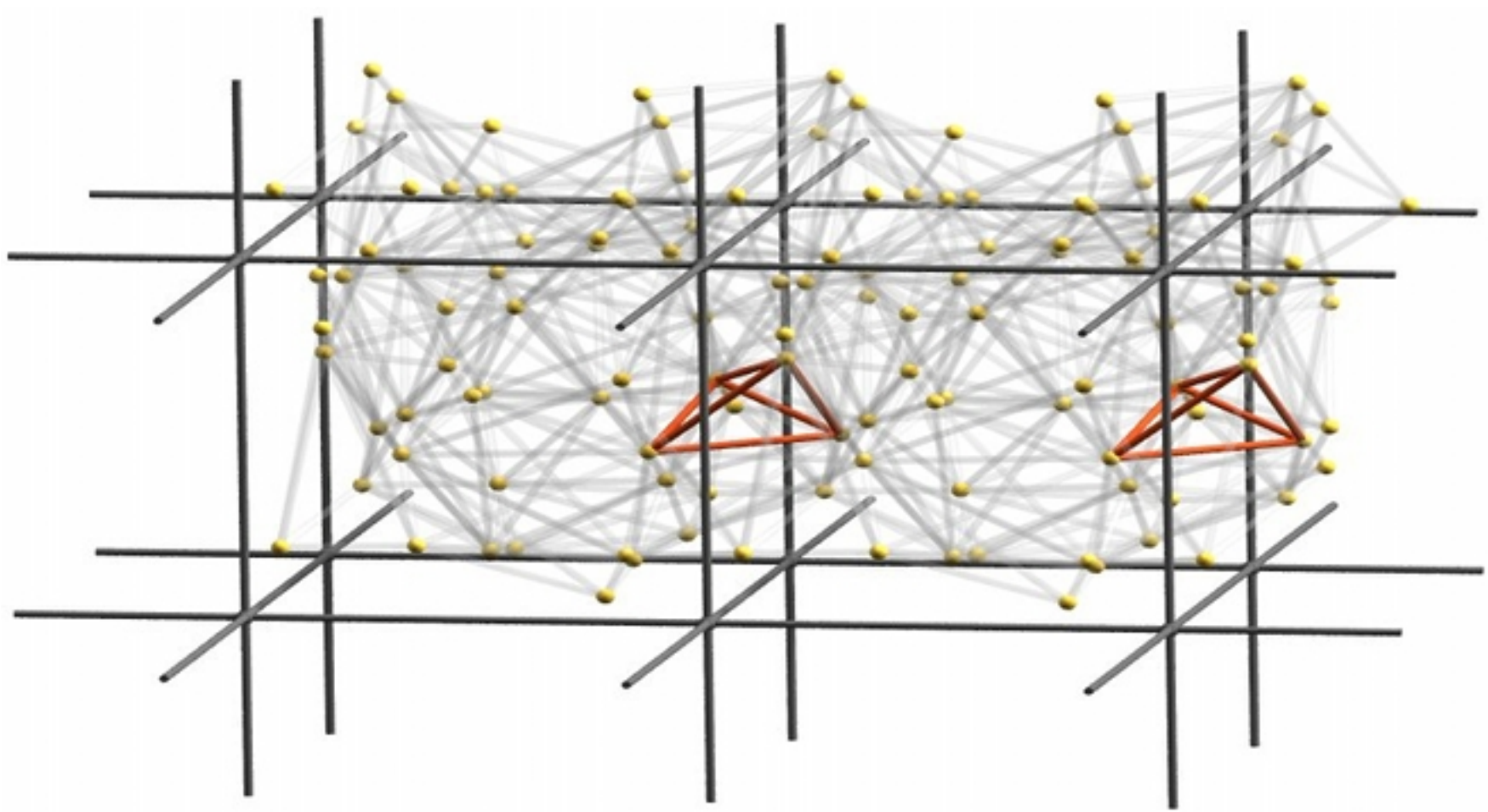
flat torus $\mathbb{T}^2 \sim \mathbb{R}^2/G$ $G =< t_x, t_y >$



universal covering space

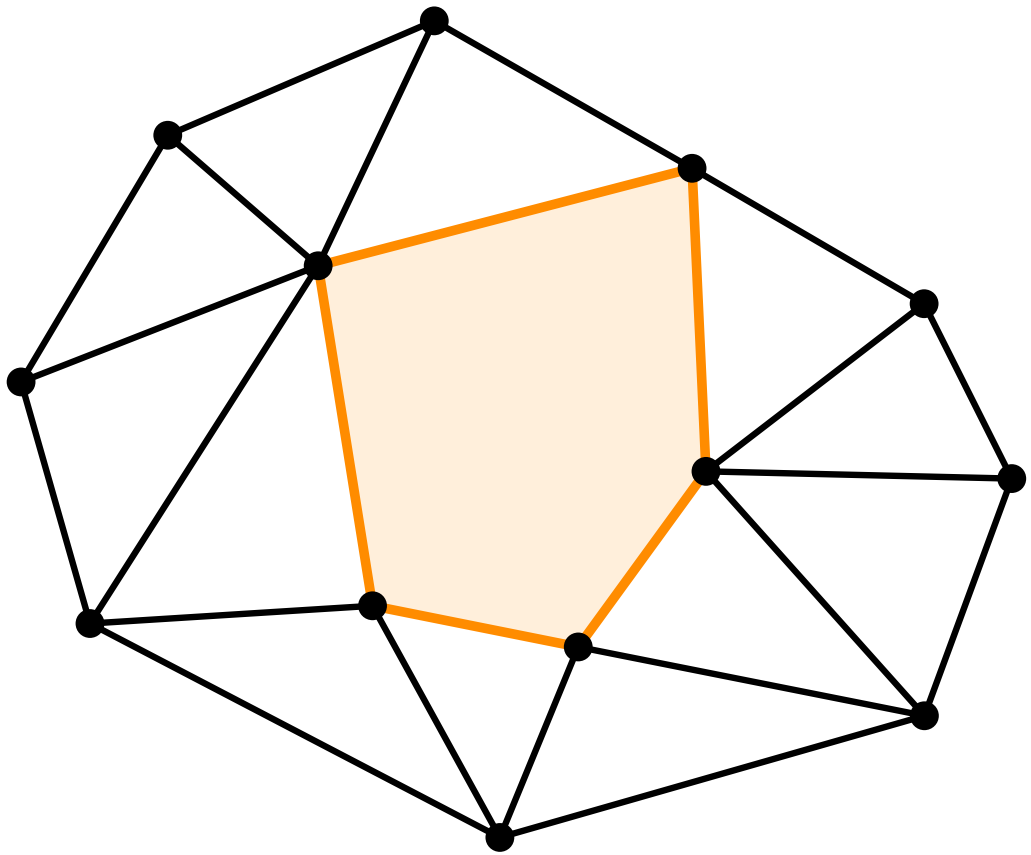
Periodic Delaunay triangulations

flat torus $\mathbb{T}^3 \sim \mathbb{R}^3/G$ $G =< t_x, t_y, t_z >$



Periodic Delaunay triangulations

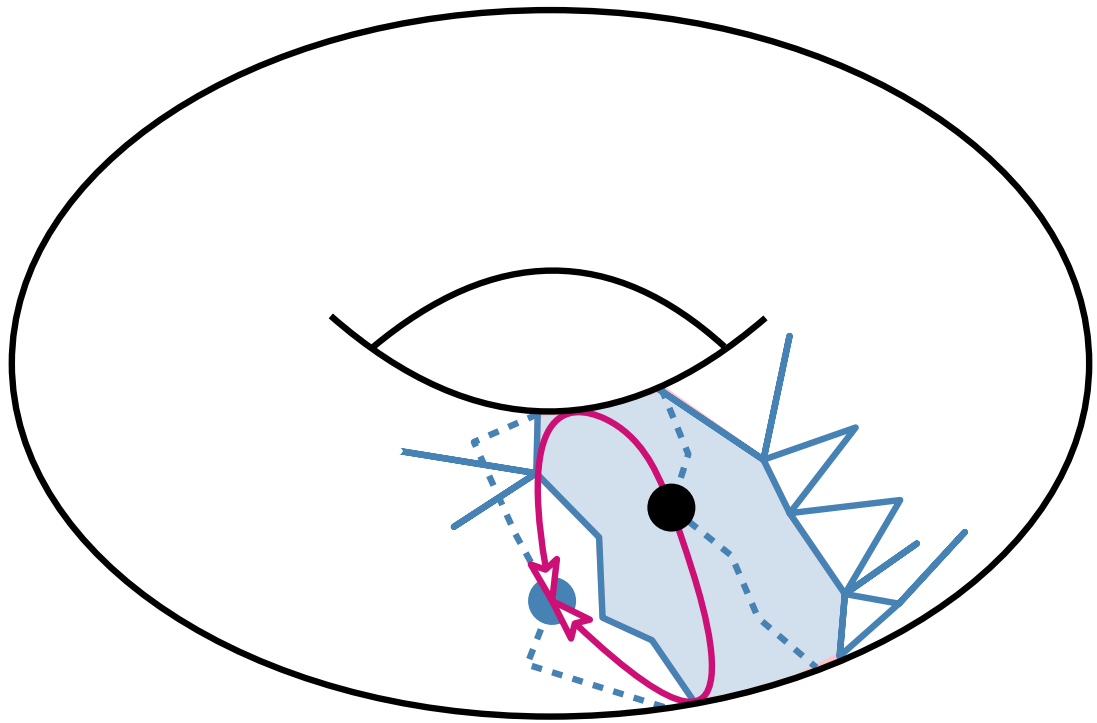
Bowyer's incremental algorithm



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)

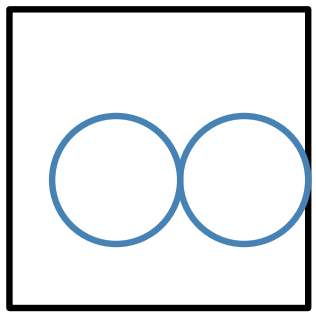
and remove them asap

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



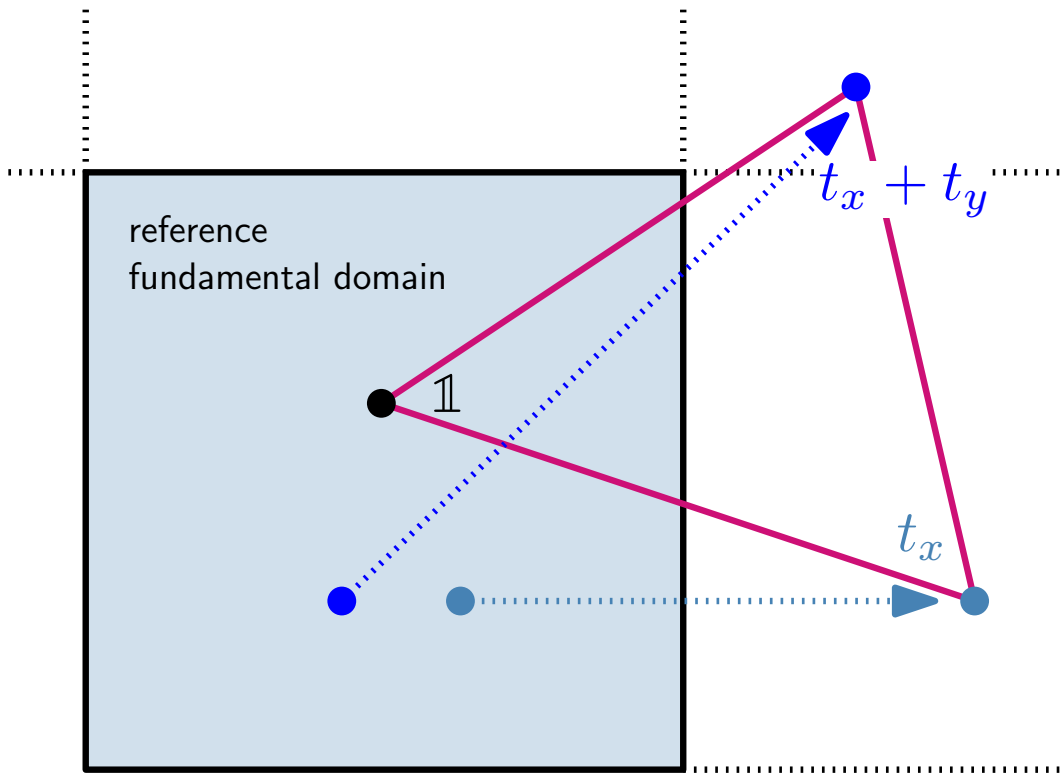
[Caroli, T. ESA'09]

Φ(largest empty disk) $<$ systole$/2$

Periodic Delaunay triangulations

`Periodic_Delaunay_triangulation< TDS, Geom_traits >`

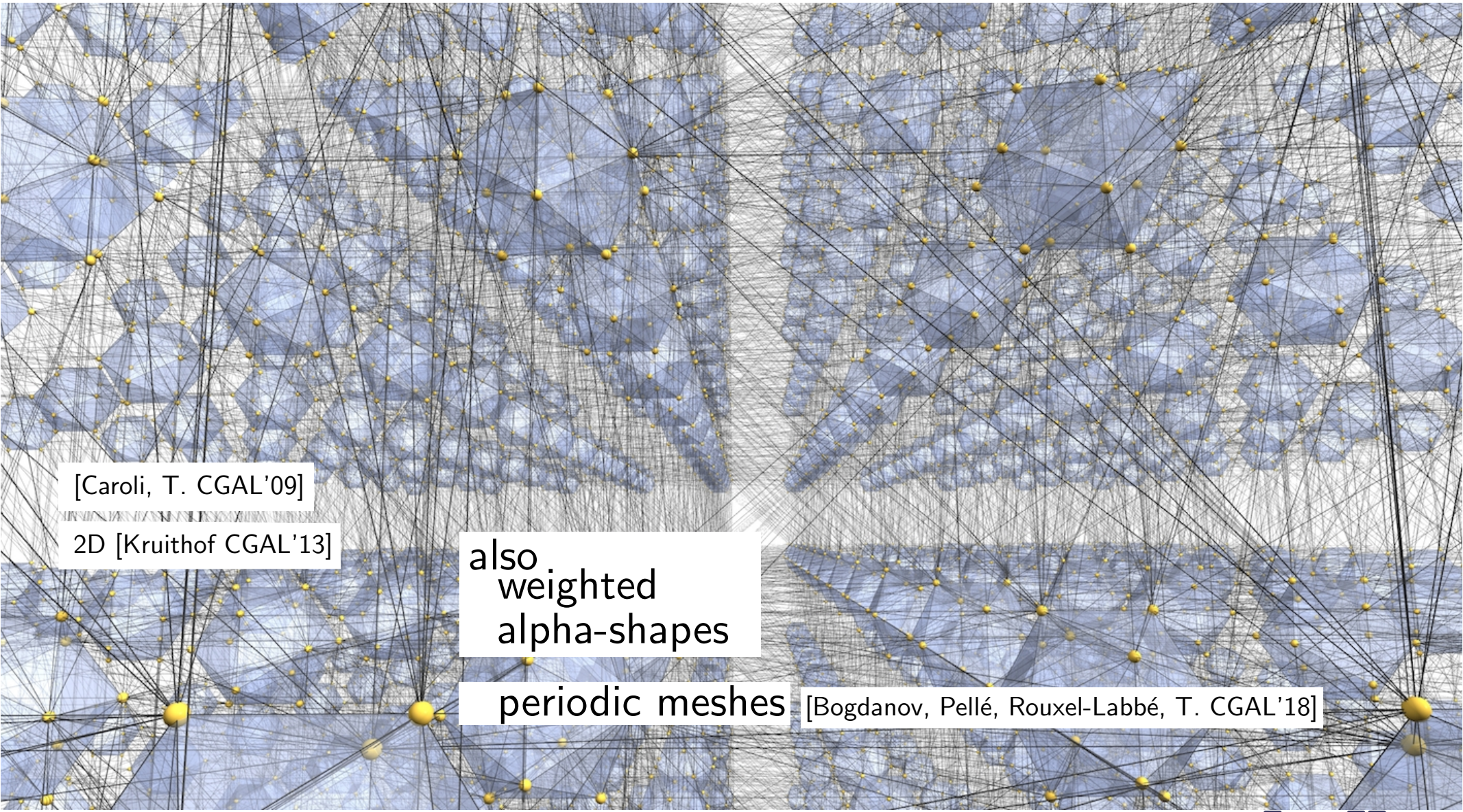
combinatorial triangulation of $\mathbb{S}^3$ reused for $\mathbb{T}^3$



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

Vertex
canonical point
$\longrightarrow$ one incident cell

Periodic Delaunay triangulations



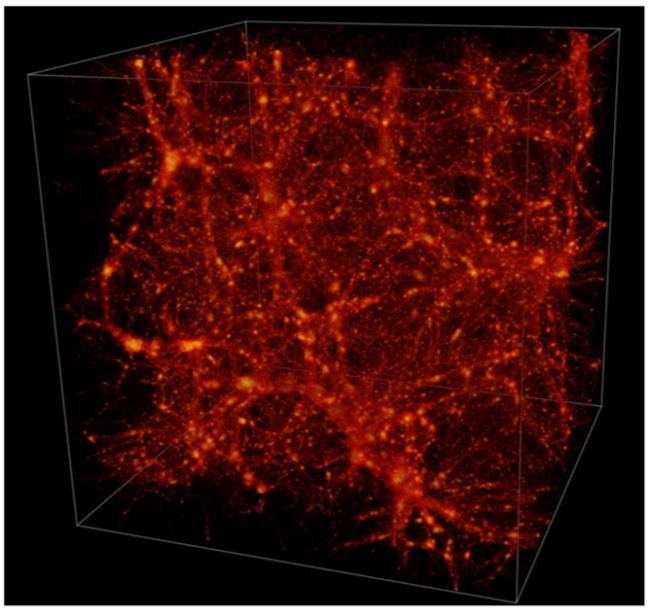
Periodic Delaunay triangulations

```
#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

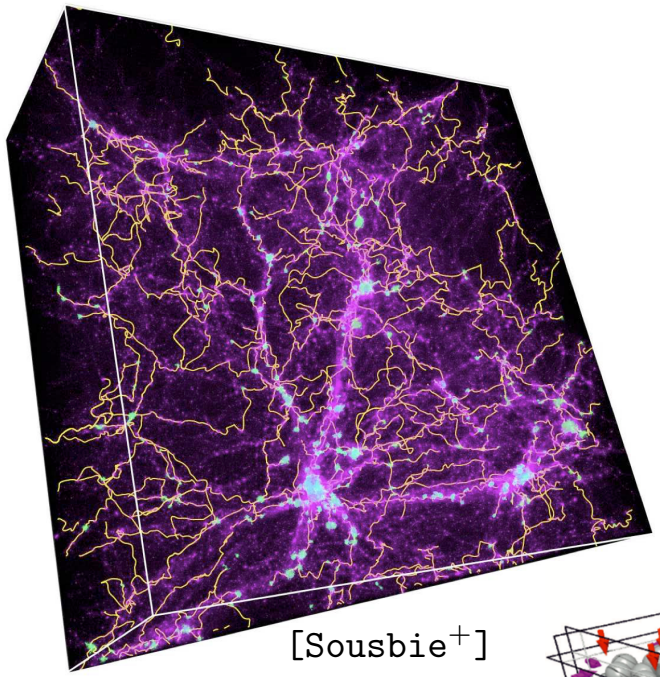
Users

Astrophysics



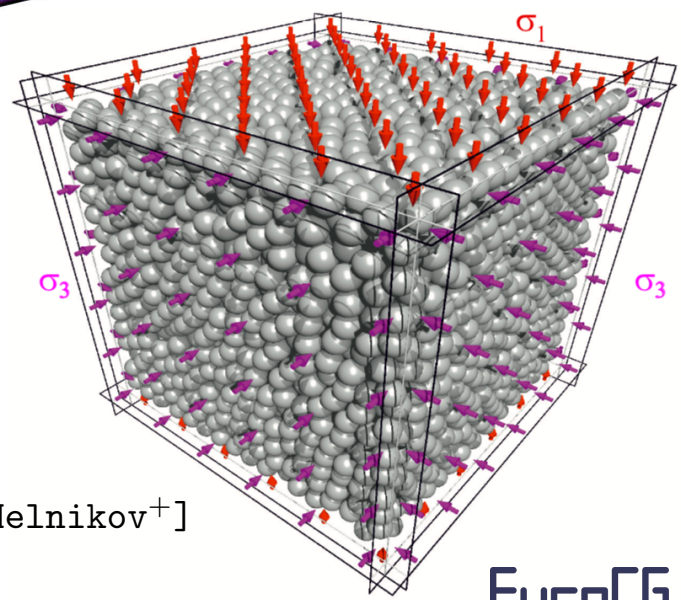
[v.d. Weijgaert^{+}]

[SoCG'12 video]



[Sousbie^{+}]

Granular materials



[Melnikov^{+}]

Particle physics
Nanostructures
etc

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

Φ(largest empty disk) $<$ systole$/2$

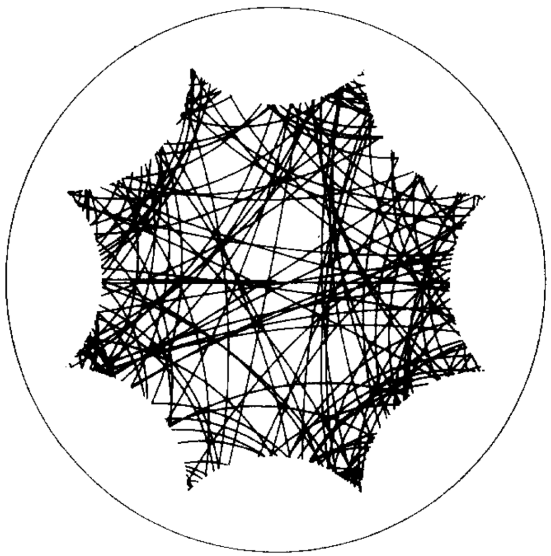
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

Motivation: e.g., in mathematical physics

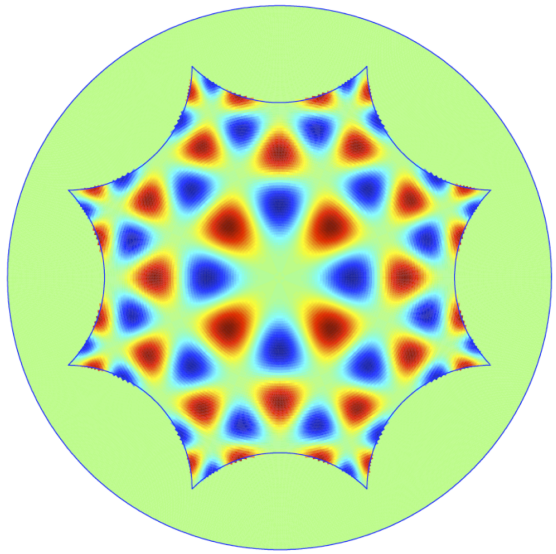


[Balazs, Voros '86]

Chaotic motion

[Sausset, Tarjus, Viot '08]

Glass-forming liquid

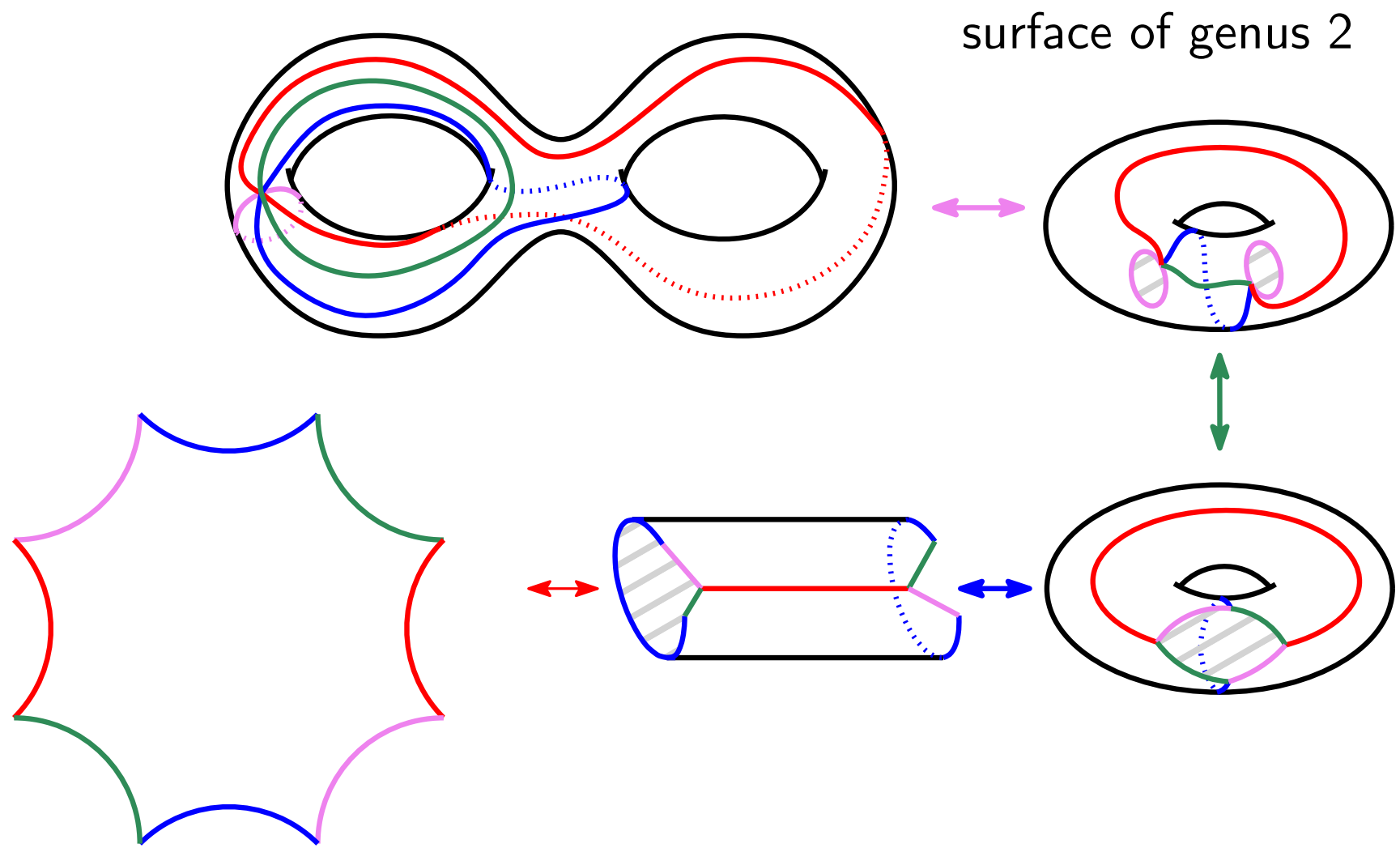


[Chossat, Faye, Faugeras '11]

Visual perception of textures

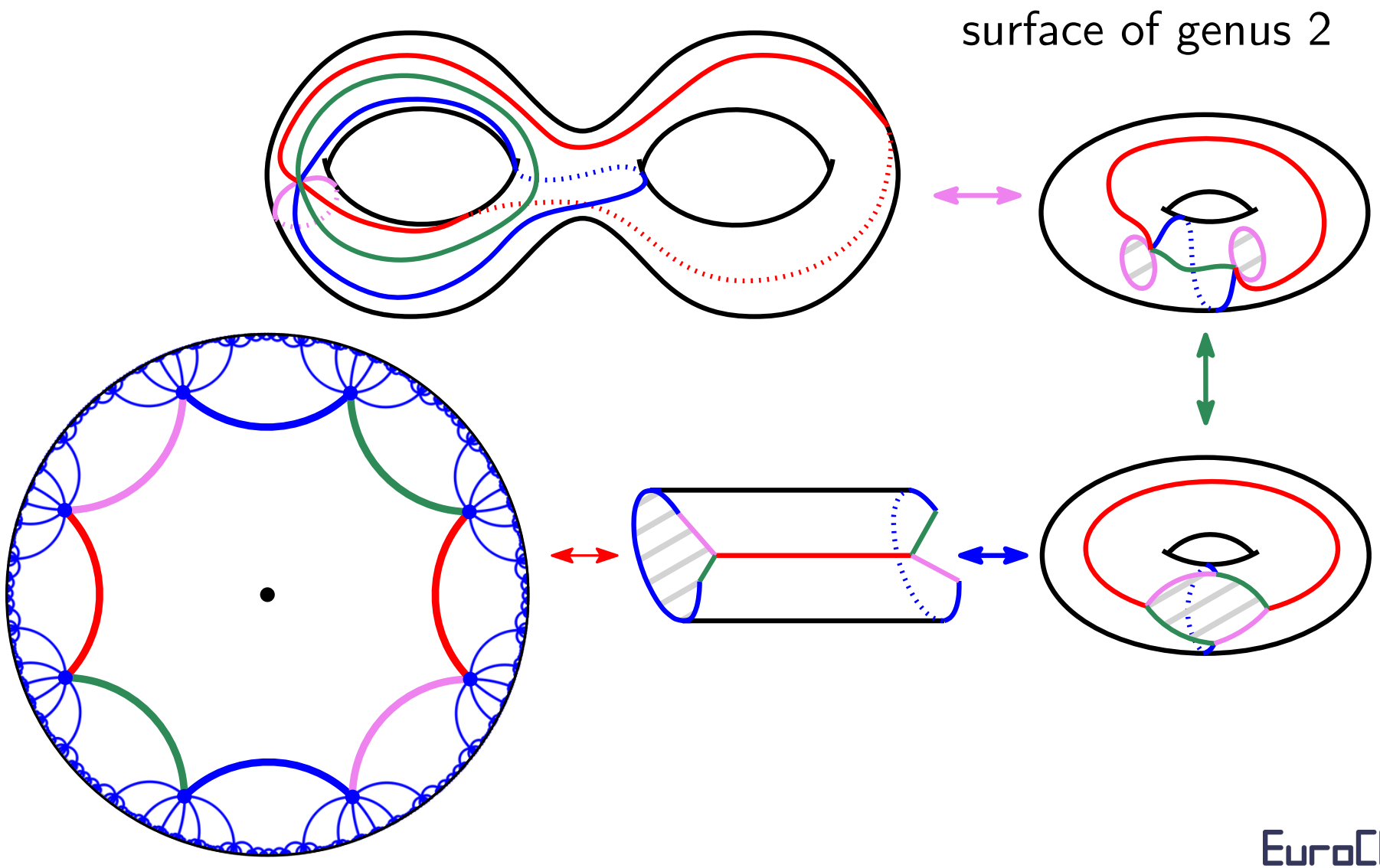
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface



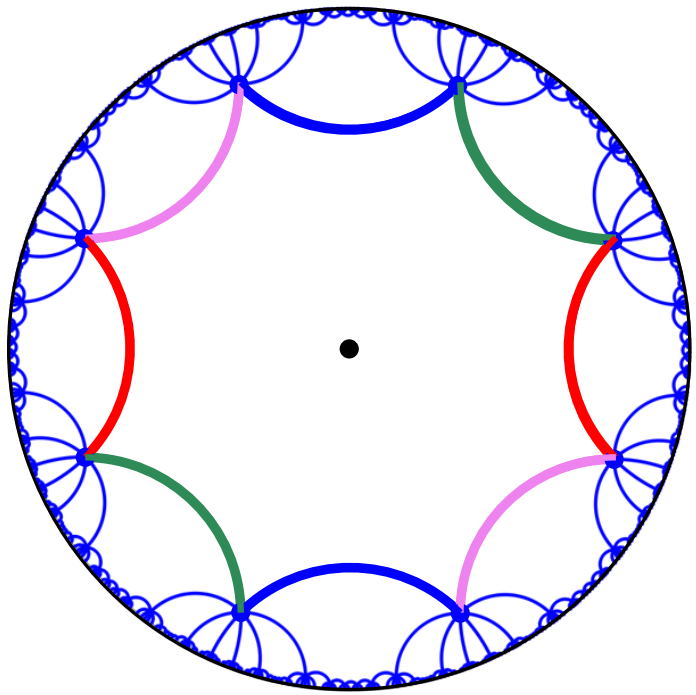
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface



Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface



the regular octagon tiles the hyperbolic plane = covering space

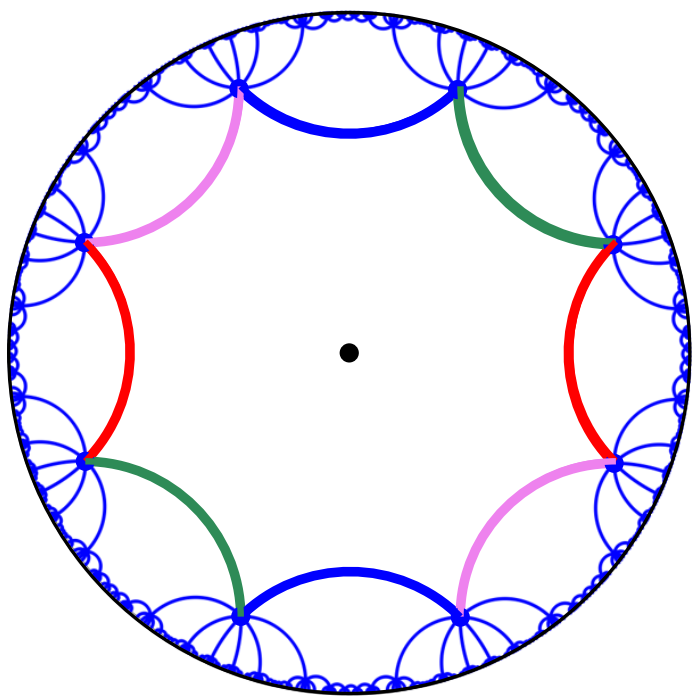
Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

$\mathbb{M}_2 \sim \mathbb{H}^2/G$ $\quad$ $G =< a, b, c, d \mid relation >$

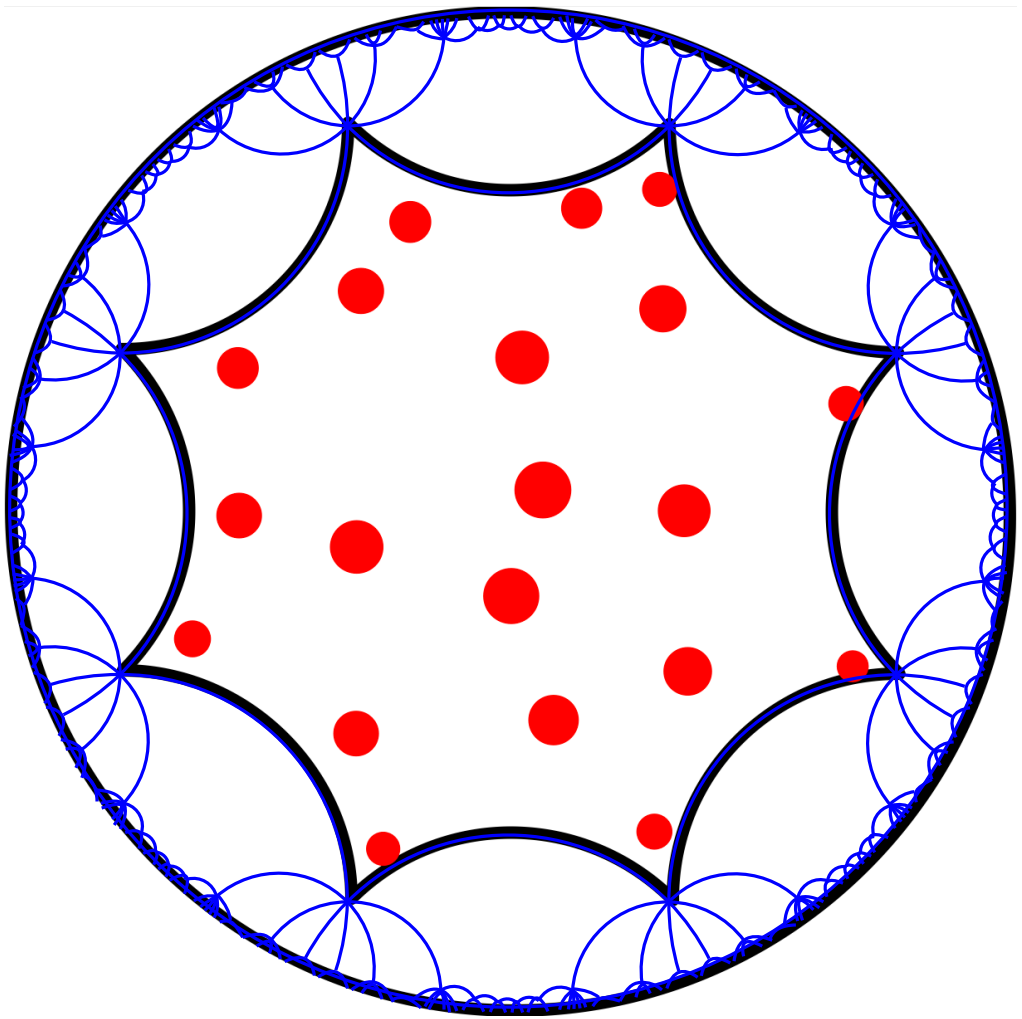
a, b, c, d hyperbolic translations

do not commute



Periodic hyperbolic triangulations

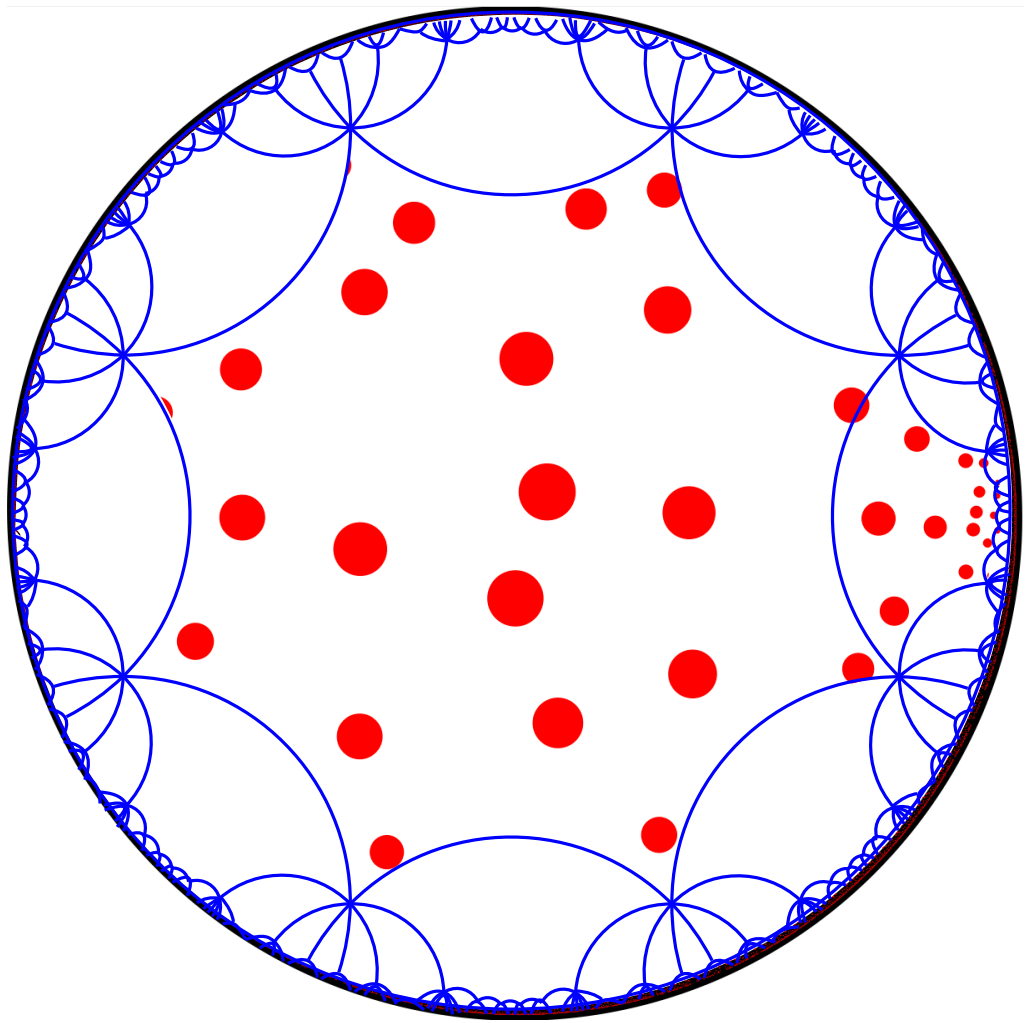
Delaunay triangulation of the Bolza surface



points on the surface

Periodic hyperbolic triangulations

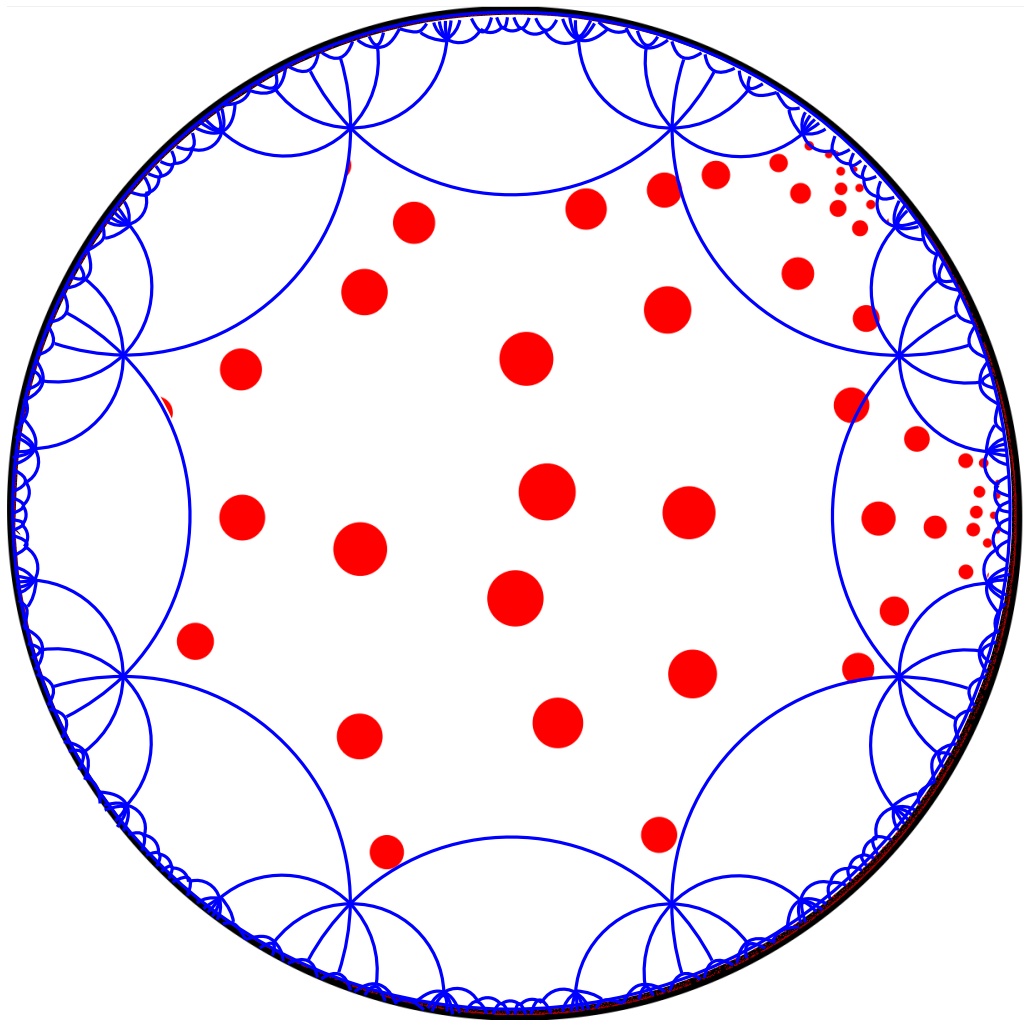
Delaunay triangulation of the Bolza surface



translated images

Periodic hyperbolic triangulations

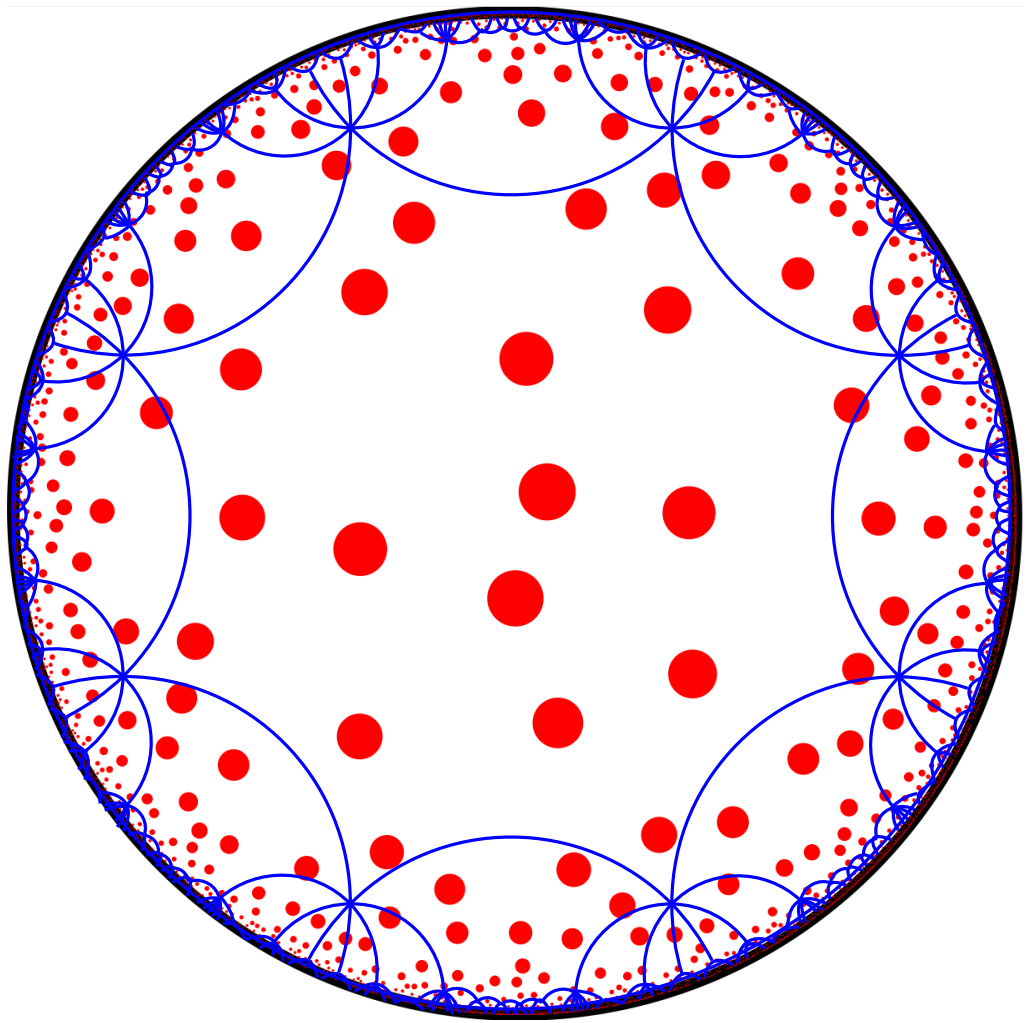
Delaunay triangulation of the Bolza surface



translated images

Periodic hyperbolic triangulations

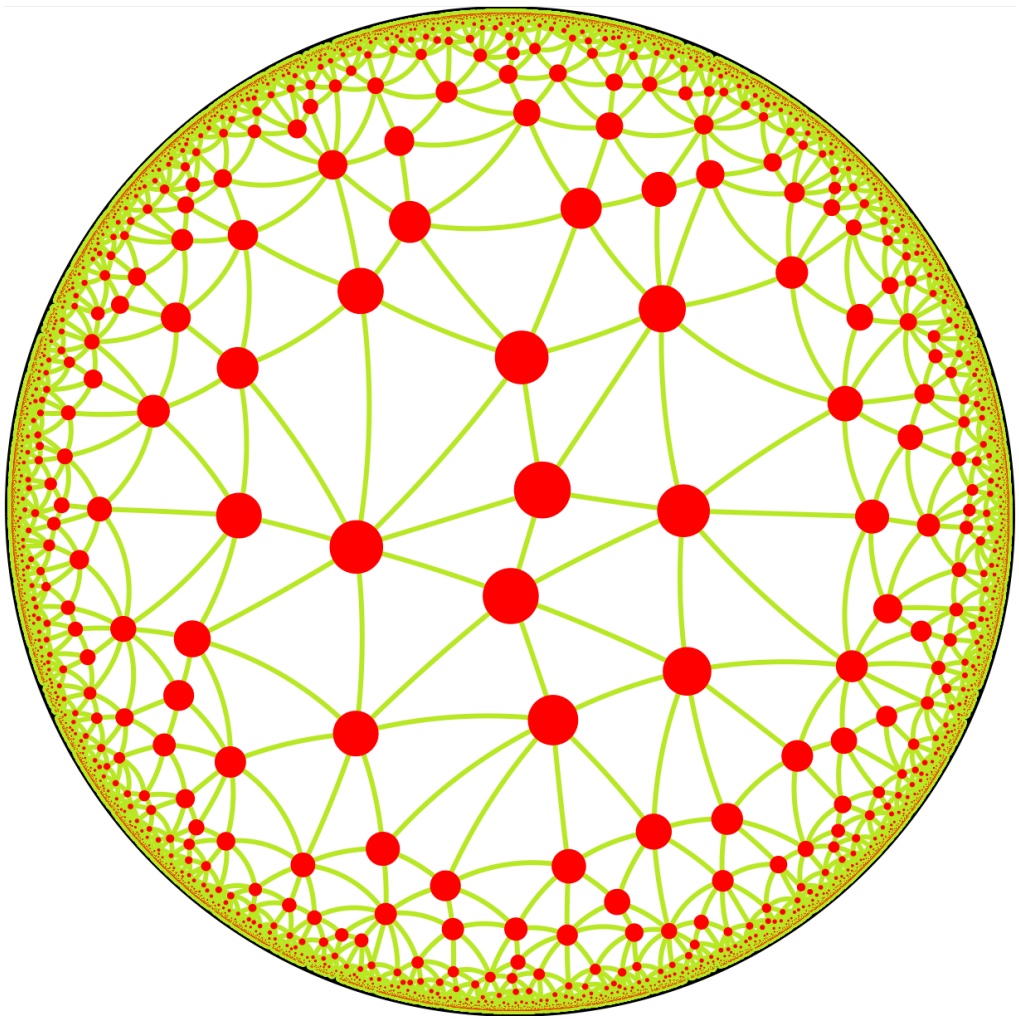
Delaunay triangulation of the Bolza surface



translated images

Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

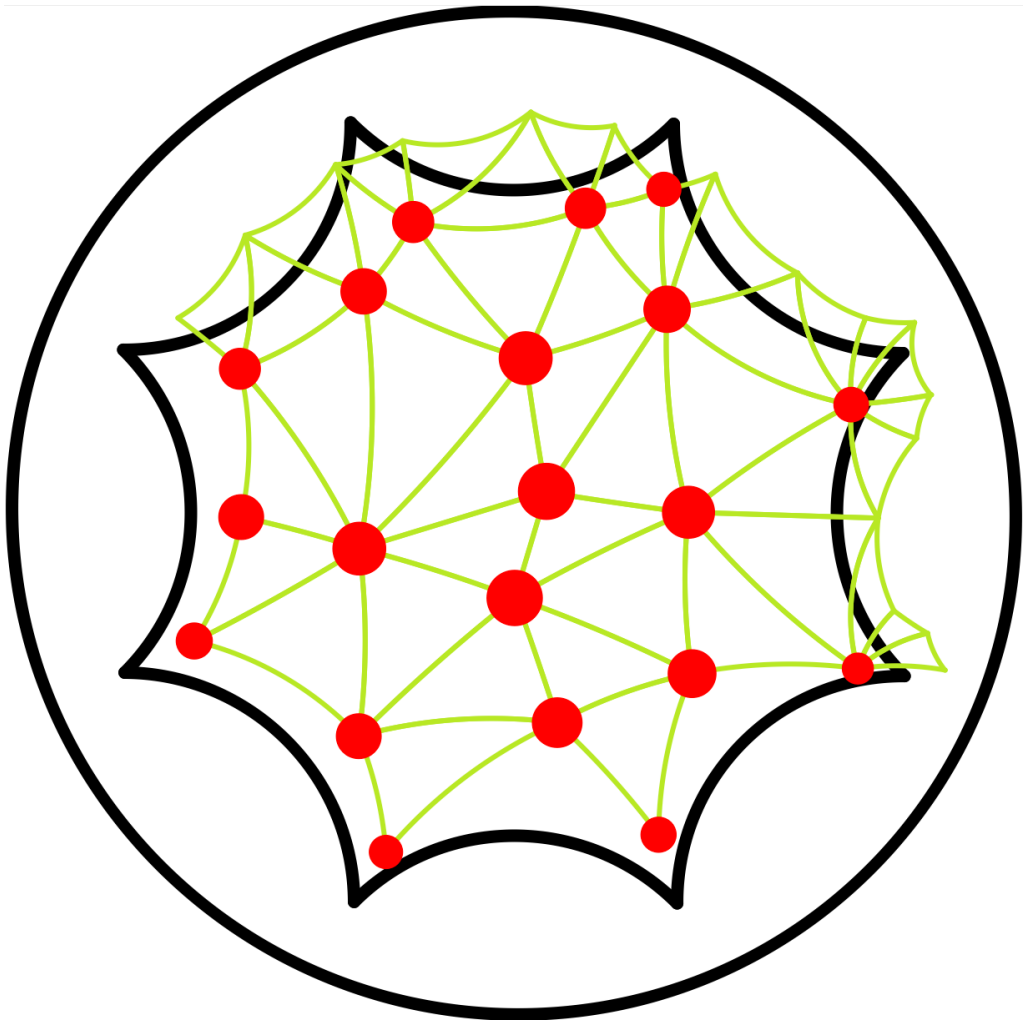


Delaunay triangulation

of the infinite point set

Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface

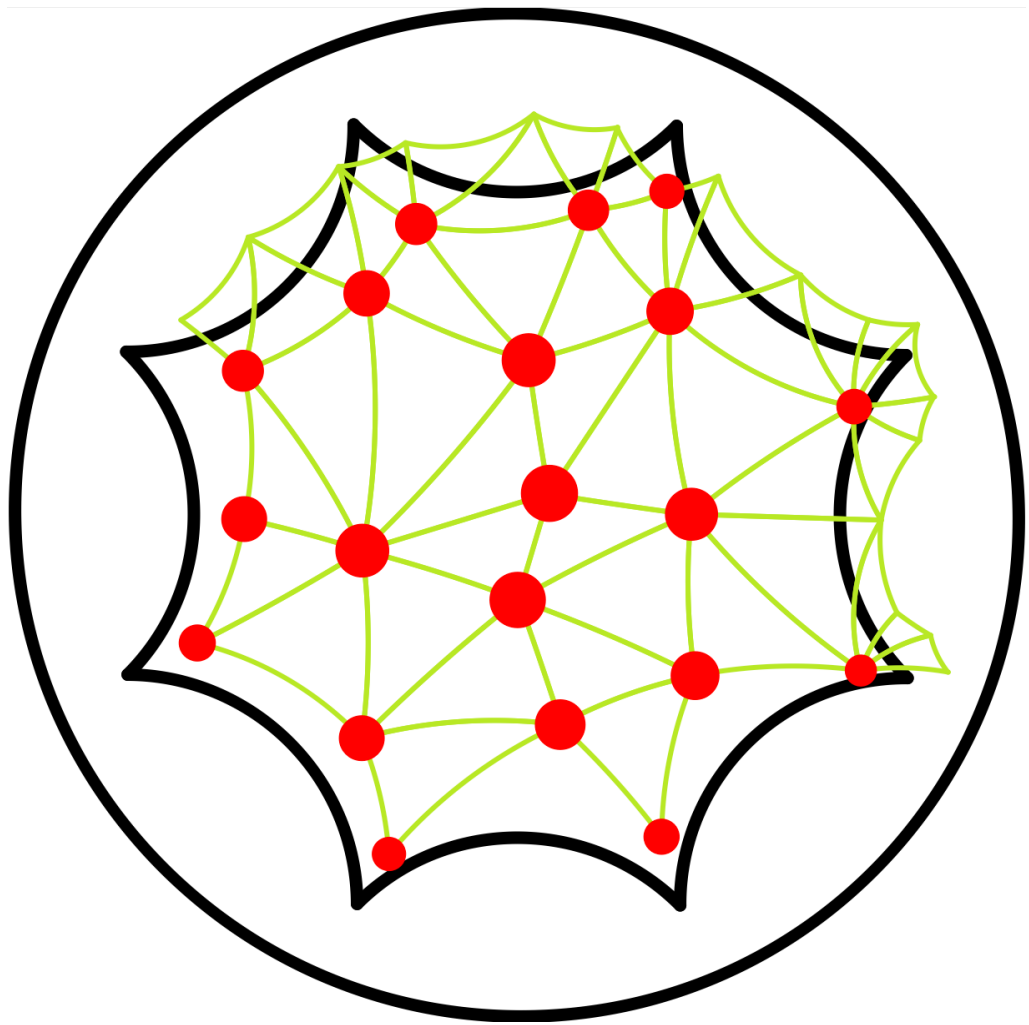


Delaunay triangulation

of the surface

Periodic hyperbolic triangulations

Delaunay triangulation of the Bolza surface



Delaunay triangulation

of the surface

(14 "dummy" points)

Φ(largest empty disk) $<$ systole$/2$

[Bogdanov, T., Vegter SoCG'16]

[Iordanov, T. SoCG'17]

[Iordanov, T. CGAL'19]

The CGAL project

Becoming a CGAL 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 CGAL user

http://www.cgal.org

Follow "Getting started"

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

Read, compile and run **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!

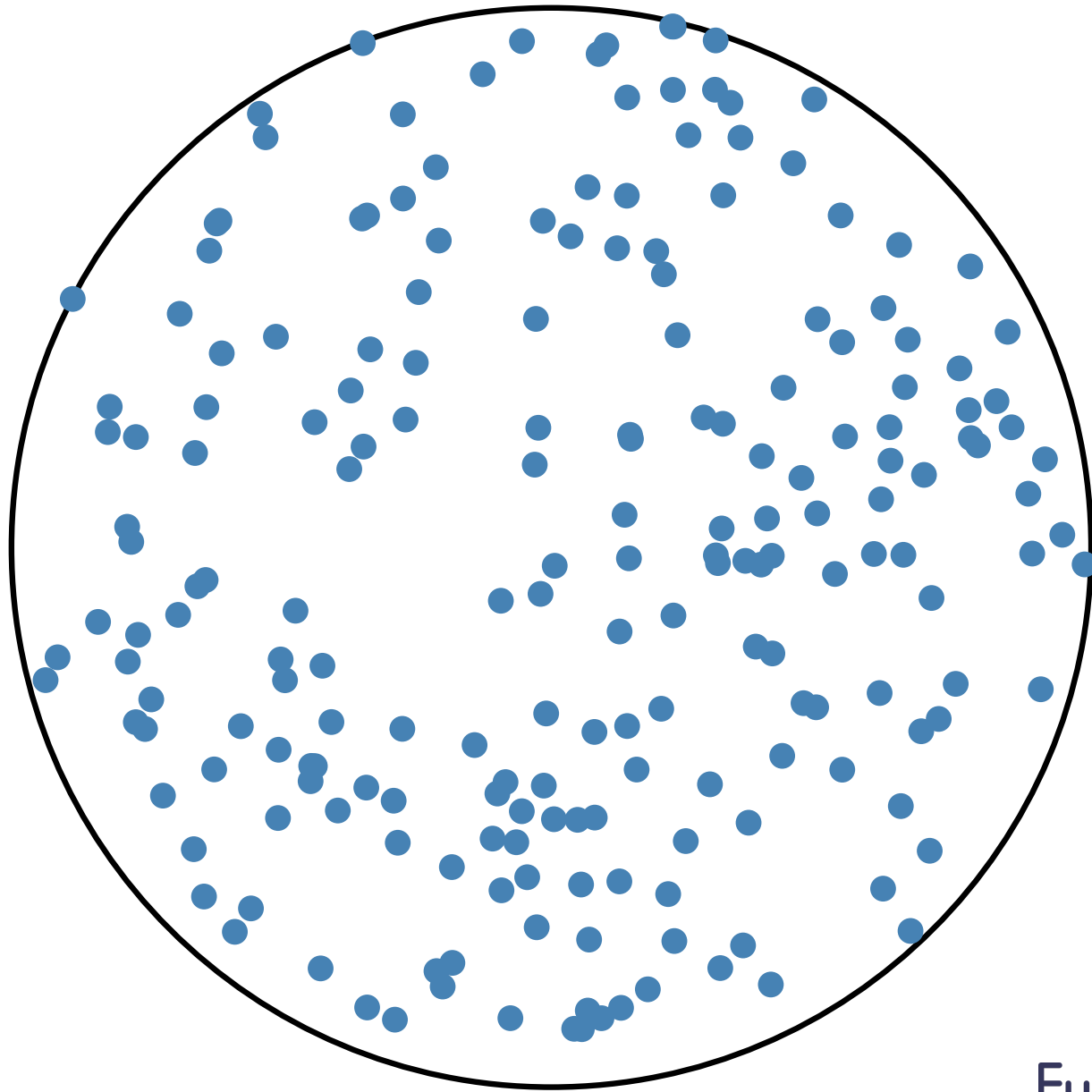
Becoming a CGAL user

Ipelets!

Generators

e.g. 200 points in a disk

$\mathbb{R}^2$



Becoming a CGAL user

Ipelets!

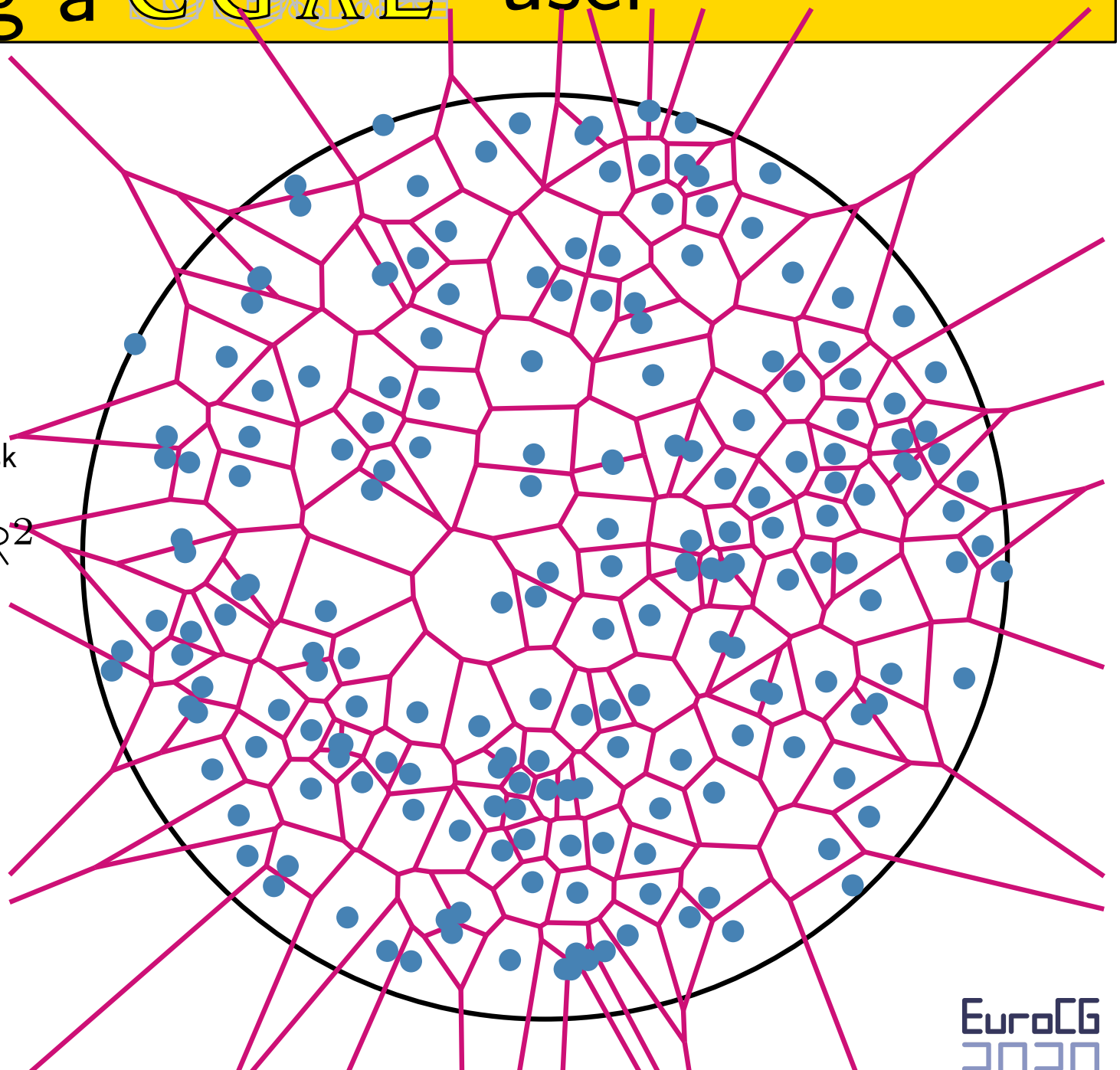
Generators

e.g. 200 points in a disk

$\mathbb{R}^2$

"Diagrams"

$\longrightarrow$ Voronoi



Becoming a CGAL user

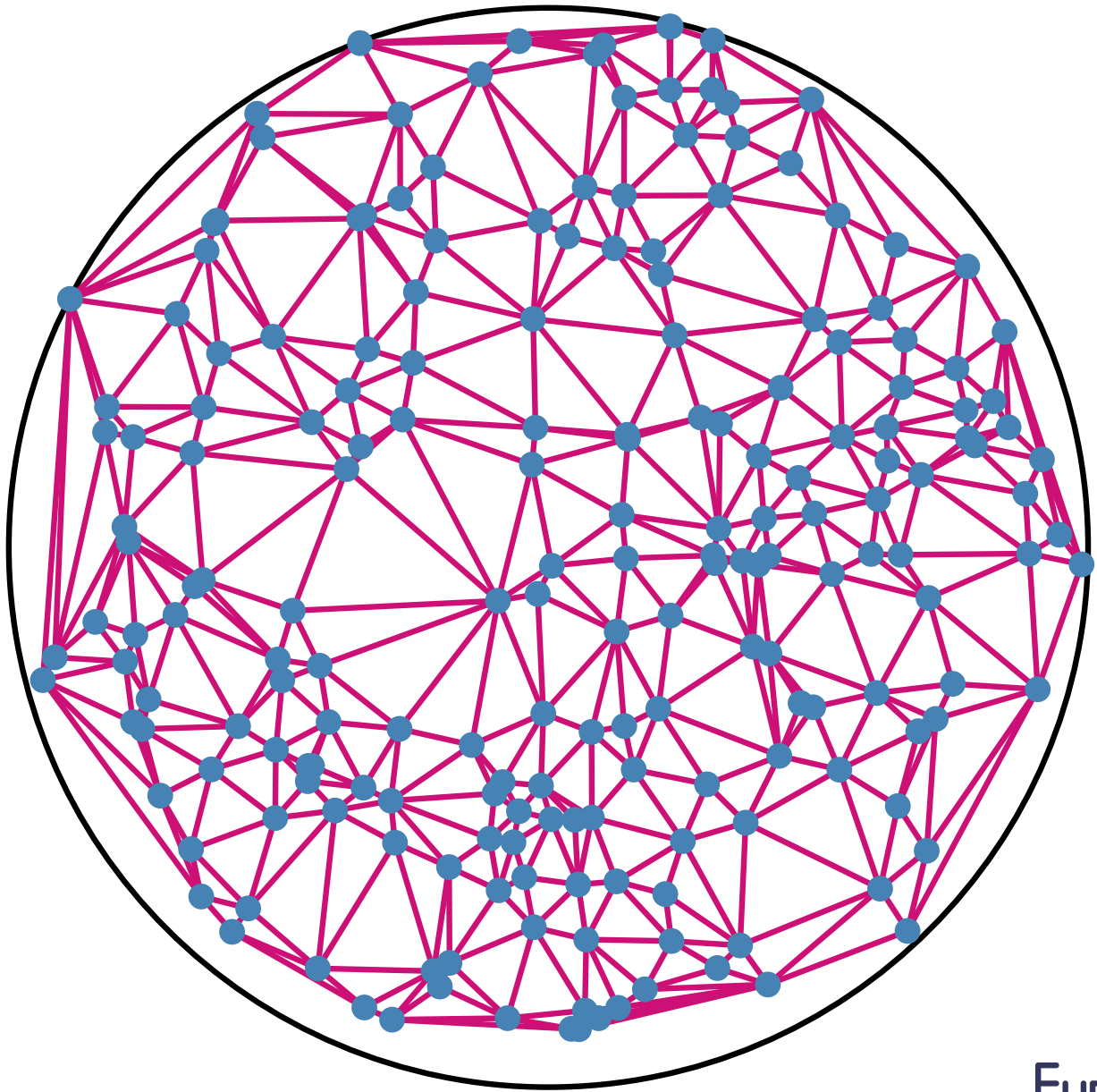
Ipelets!

Generators

e.g. 200 points in a disk

$\mathbb{R}^2$

"Triangulations"
$\longrightarrow$ 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

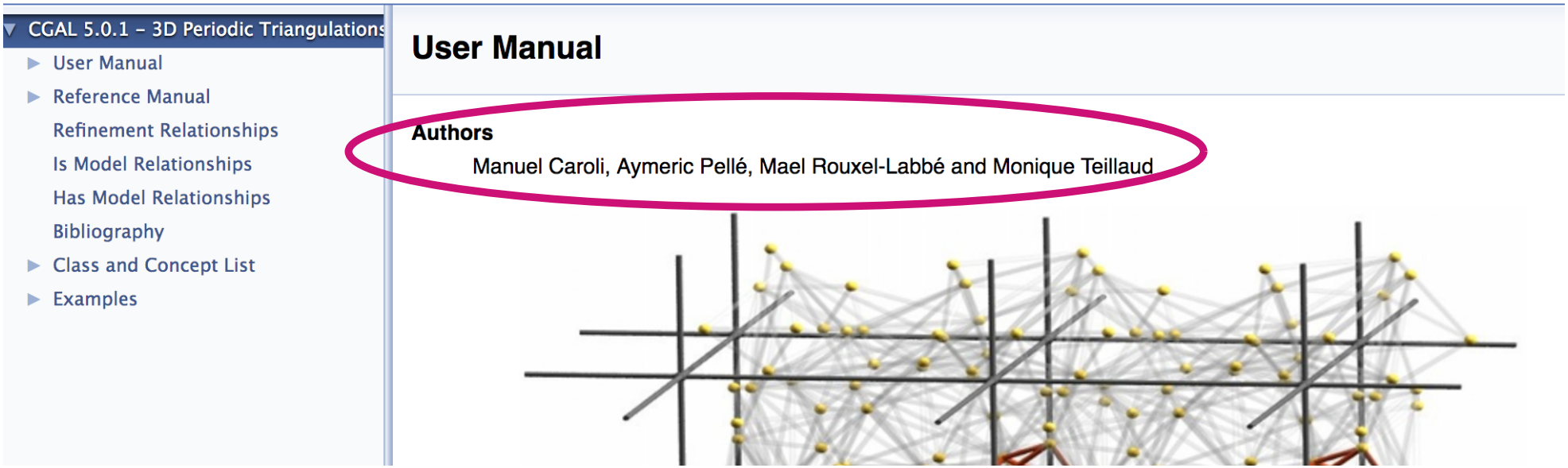
Review and integration process

Becoming a CGAL developer

Contributors are clearly acknowledged — Visibility

- Appear as Authors in manual chapters

CGAL 5.0.1 - 3D Periodic Triangulations



- Listed in the "People" web page

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 **Visibility**

[The institution of the] authors keep the copyright

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

Becoming a CGAL developer

Key for success = Diversity of its members

gather many skills:

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

Join the crowd!

Becoming a CGAL developer


CGAL 20 JAHRE
Datum/Date: 10.09.2016 Zug/Train: 571 (12:33) Position: -1b
09.09.2016 15:27:10
Reserviert Réservé
Bitte Sitzplätze für Gruppen freihalten
Veuillez laisser les places libre pour les groupes
Per favore tenere liberi i posti per gruppi
CGAL-ETH Zürich (Salow) 20
Zürich HB (13:37) - Landquart (14:41)
September 2016


CGAL 20 JAHRE
DE PLANIS
September 2016
EuroCG
2020

Thanks to

past,
present,
future

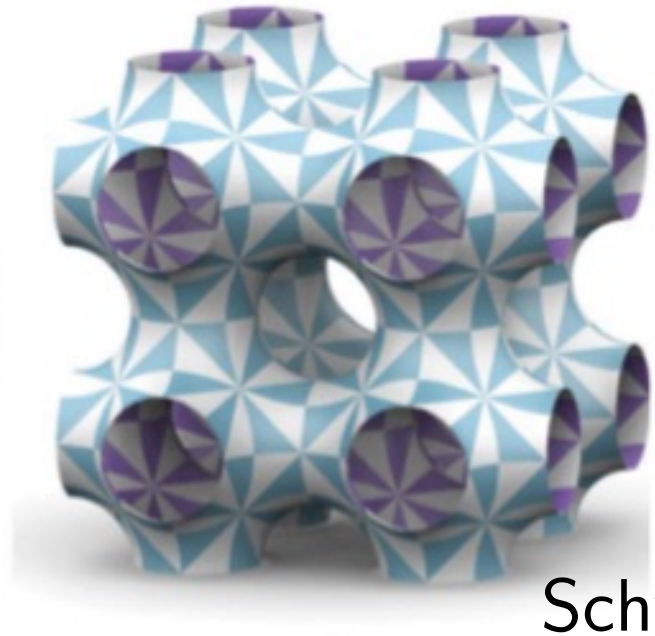
developers
and
users
of

you all!

Otfried for Ipe

Triply periodic minimal surfaces

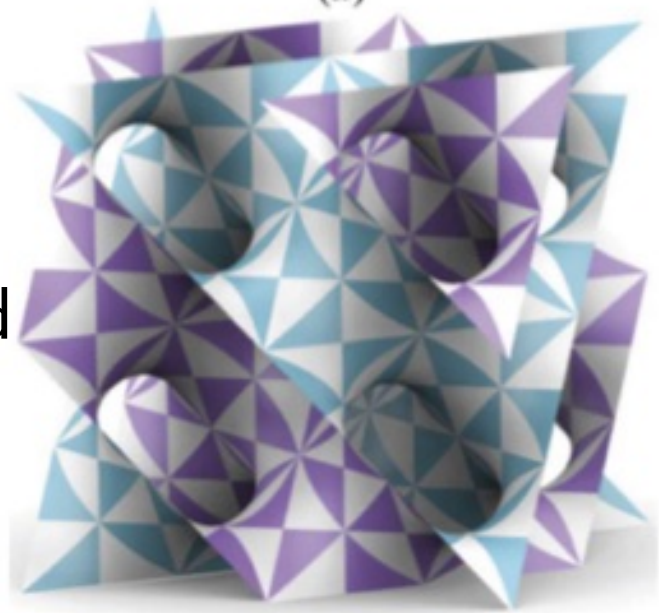
(a)



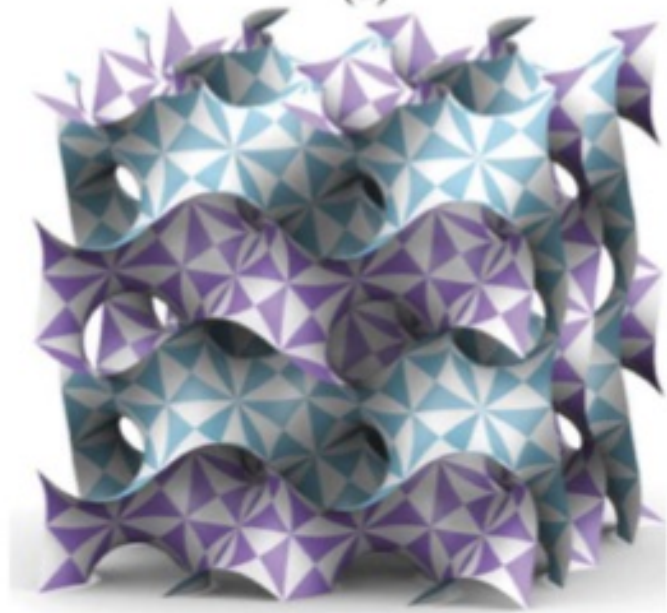
(b)

Schwartz P

Diamond



(c)



(d)

Gyroid