Computational Geometry Algorithms Library

Pierre Alliez
INRIA

Andreas Fabri
GeometryFactory

http://www.cgal.org/siggraph2009

The updated handout has

- extensive comments in the "Notes" part of Powerpoint.
- hyperlinks to the CGAL User and Reference Manual (~3500 pages).
- hyperlinks to precompiled demos illustrating the algorithms. They will be made available online.

Course Outline

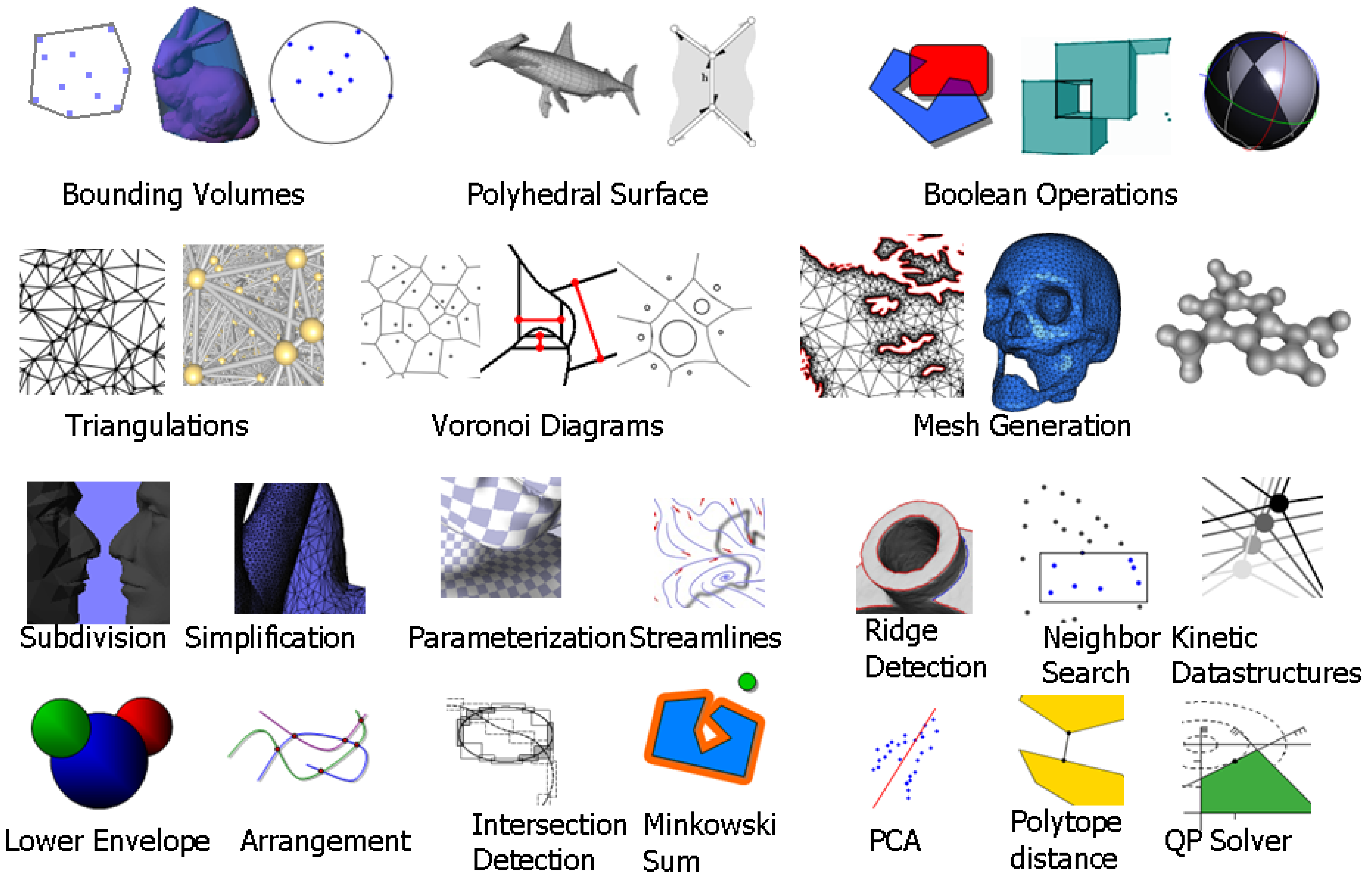
- General Introduction
- CGAL for 2D Vector Graphics
- CGAL for Point Sets
- CGAL for Modeling and Processing of Polyhedral Surfaces
- CGAL for Mesh Generation
- Questions and Answers

Mission Statement

"Make the large body of geometric algorithms developed in the field of computational geometry available for industrial applications"

CGAL Project Proposal, 1996

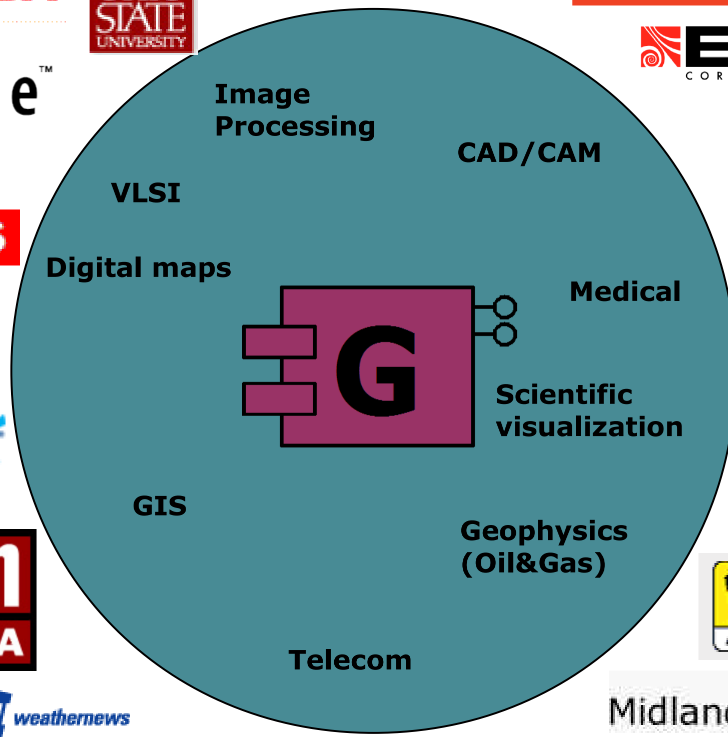
Algorithms and Datastructures



CGAL in Numbers

500,000	lines of C++ code
10,000	downloads/year (+ Linux distributions)
3,500	manual pages
3,000	subscribers to cgal-announce
1,000	subscribers to cgal-discuss
120	packages
90	commercial users
20	active developers
12	months release cycle
2	licenses: Open Source and commercial

Some Commercial Users



VDRC
pulsic
Orbotech
In-Three
NOESIS
protomold
ARADEX
Dassault Systemes
ECL
Wihofszky Software
TOSHIBA
The Ohio State University
cadence
Exa Corporation
ZWCAD
MPC
Katholieke Universiteit Leuven
Agilent Technologies
BAE SYSTEMS
schaerermayfield
TomTec Imaging Systems
Leica Geosystems
QinetiQ
St. Jude Medical
GE Healthcare
Virtualwind
School of Biomedical Sciences
The MathWorks
Safe Software
rm DATA
Archivideo
Polytec
Agip
TGS
Industrial Research Limited
weathernews
Midland Valley
ESRI
TruePosition
france telecom
BT
IFP
BSAP
Saudi Aramco
Total
brgm
ENVITIA

Why They Use CGAL

" I recommended to the senior management that we start a policy of buying-in as much functionality as possible to reduce the quantity of code that our development team would have to maintain.

This means that we can concentrate on the application layer and concentrate on our own problem domain."

Senior Development Engineer
& Structural Geologist

Midland Valley Exploration

Why They Use CGAL

" My research group JYAMITI at the Ohio State University uses CGAL because it provides an efficient and robust code for Delaunay triangulations and other primitive geometric predicates. Delaunay triangulation is the building block for many of the shape related computations that we do. [...]

Without the robust and efficient codes of CGAL, these codes could not have been developed. "

Tamal Dey
Professor, Ohio State University

CGAL Open Source Project

Project = « Planned Undertaking »

- Institutional members make a long term commitment: Inria, MPI, Tel-Aviv U, Utrecht U, Groningen U, ETHZ, GeometryFactory, FU Berlin, Forth, U Athens
- Editorial Board
  - Steers and animates the project
  - Reviews submissions
- Development Infrastructure
  - Gforge: svn, tracker, nightly testsuite,...
  - 120p developer manual and mailing list
  - Two 1-week developer meetings per year

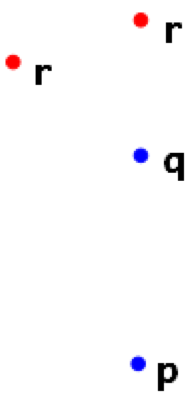
Contributions

- Submission of specifications of new contributions
- Review and decision by the Editorial Board

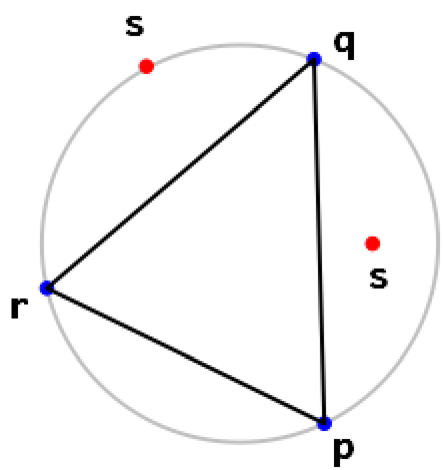
- Value for contributor
  - Integration in the CGAL community
  - Gain visibility in a mature project
  - Publication value for accepted contributions

Exact Geometric Computing

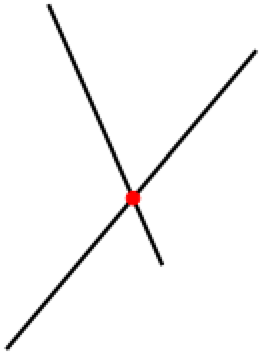
Predicates and Constructions



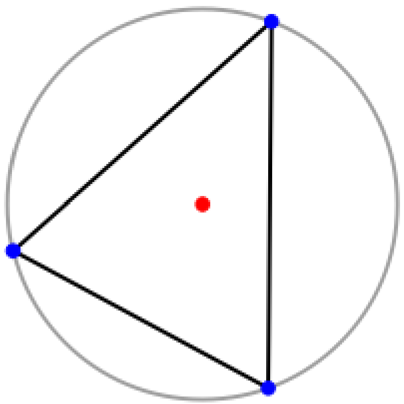
orientation



in_circle



intersection



circumcenter

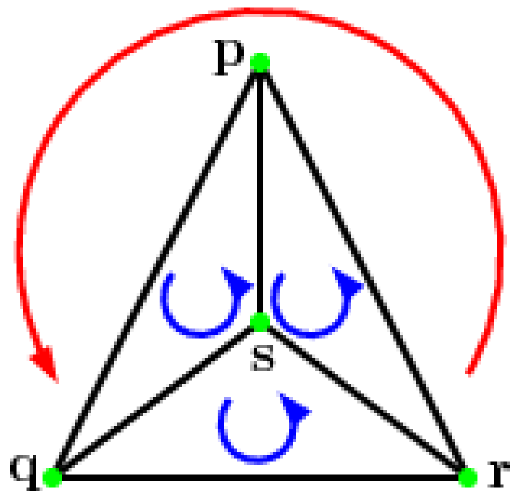
Robustness Issues

- Naive use of floating-point arithmetic causes geometric algorithms to:
  - Produce [slightly] wrong output
  - Crash after invariant violation
  - Infinite loop

- There is a gap between
  - Geometry in theory
  - Geometry with floating-point arithmetic

Geometry in Theory

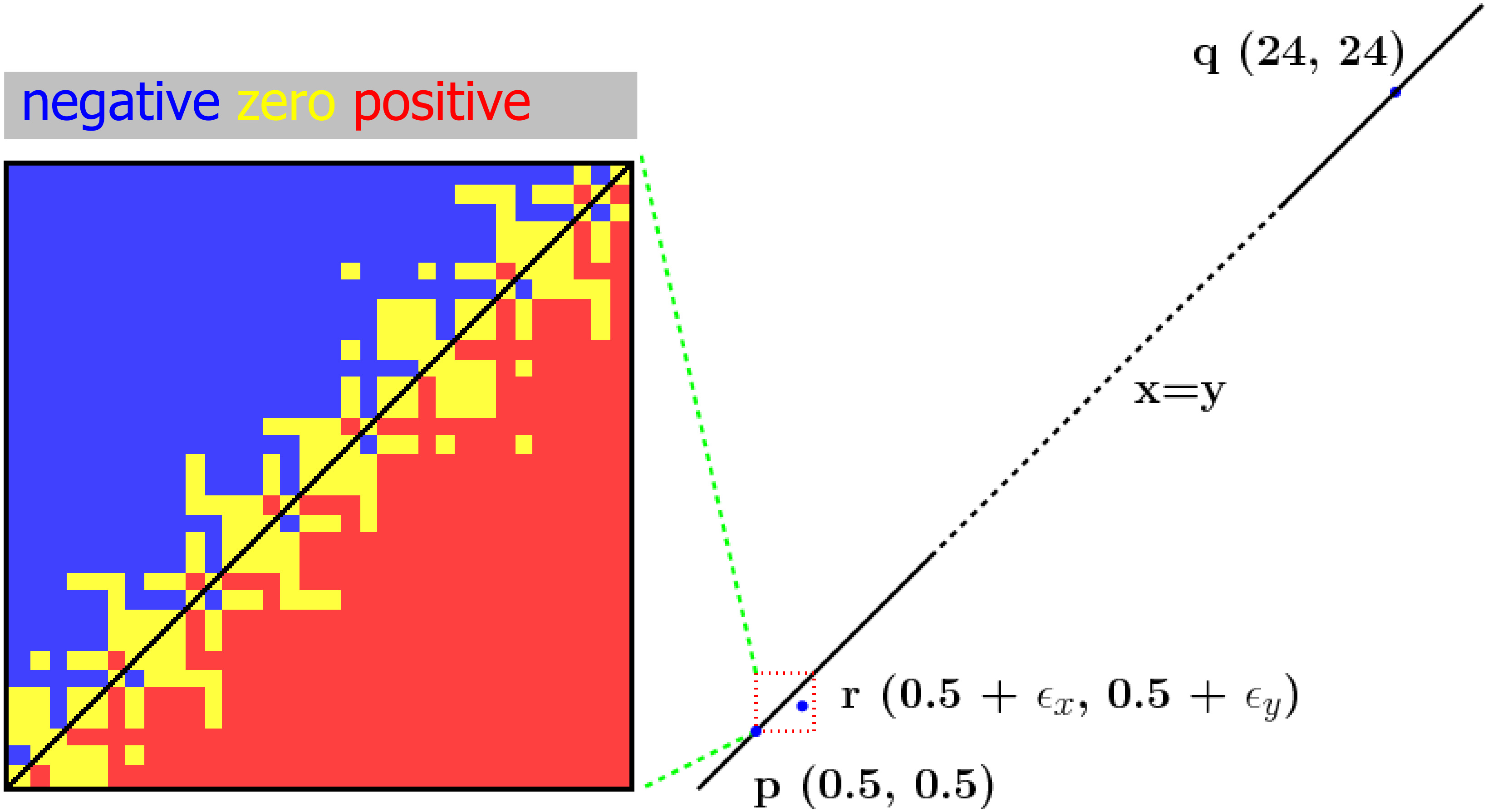
ccw(s,q,r) & ccw(p,s,r) & ccw(p,q,s) ⇒ ccw(p,q,r)



Correctness proofs of algorithms rely on such theorems

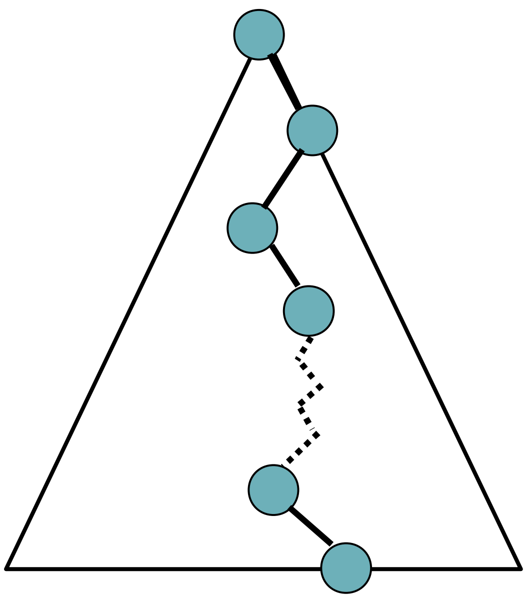
Demo: The Trouble with Double

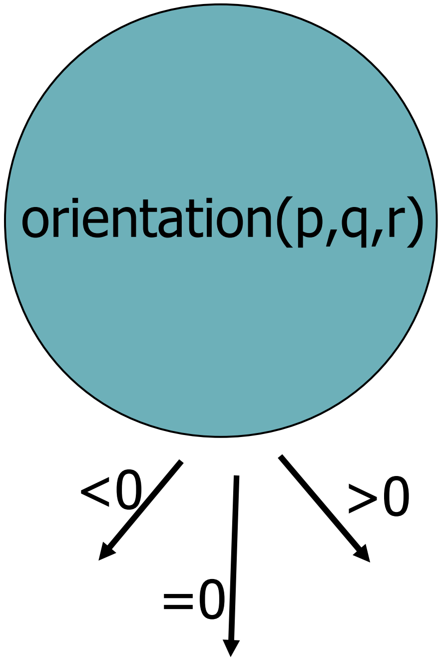
orientation(p,q,r) = sign($(p_x - r_x)(q_y - r_y) - (p_y - r_y)(q_x - r_x)$)



Exact Geometric Computing [Yap]

Make sure that the control flow in the implementation corresponds to the control flow with exact real arithmetic





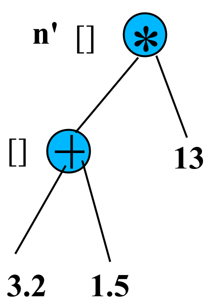
Filtered Predicates

- Generic functor adaptor Filtered_predicate<>
  - Try the predicate instantiated with intervals
  - In case of uncertainty, evaluate the predicate with multiple precision arithmetic
- Refinements:
  - Static error analysis
  - Progressively increase precision

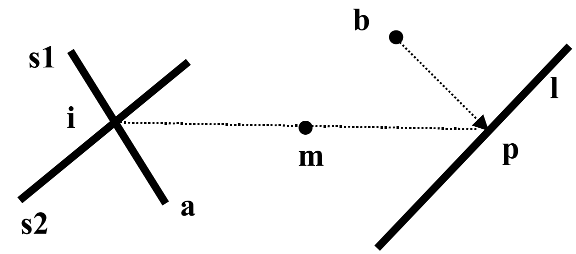
Filtered Constructions

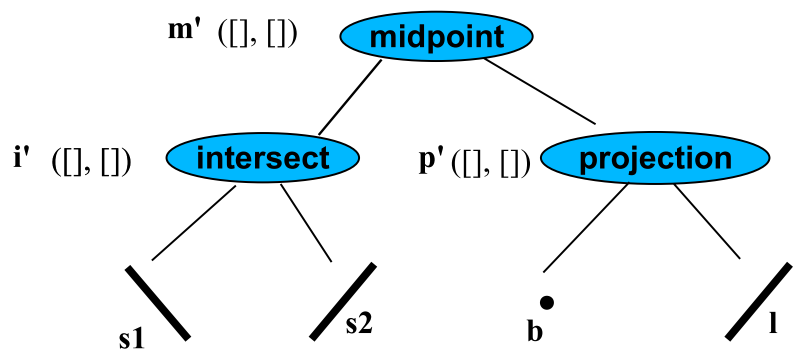
Lazy number = interval and arithmetic expression tree

$(3.2 + 1.5) * 13$



Lazy object = approximated object and geometric operation tree





Test that may trigger an exact re-evaluation:

if (n' < m')

if (collinear(a',m',b'))

The User Perspective

- Convenience Kernels
  - **Exact_predicates_inexact_constructions_kernel**
  - **Exact_predicates_exact_constructions_kernel**
  - **Exact_predicates_exact_constructions_kernel_with_sqrt**
- Number Types
  - **double, float**
  - **CGAL::Gmpq (rational), Core (algebraic)**
  - **CGAL::Lazy_exact_nt<ExactNT>**
- Kernels
  - **CGAL::Cartesian<NT>**
  - **CGAL::Filtered_kernel<Kernel>**
  - CGAL::Lazy_kernel<NT>

CGAL manual

Merits and Limitations

- Ultimate robustness inside the black box
- The time penalty is reasonable, e.g. 10% for 3D Delauny triangulation of 1M random points
- Limitations of Exact Geometric Computing
  - Topology preserving rounding is non-trivial
  - Construction depth must be reasonable
  - Cannot handle trigonometric functions

Generic Programming

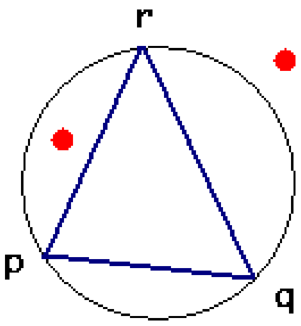
STL Genericity

```
template <class Key, class Less>
class set {
  Less less;

  insert(Key k)
  {
     if (less(k, treenode.key))
       insertLeft(k);
     else
       insertRight(k);
  }
};
```

STL manual

CGAL Genericity

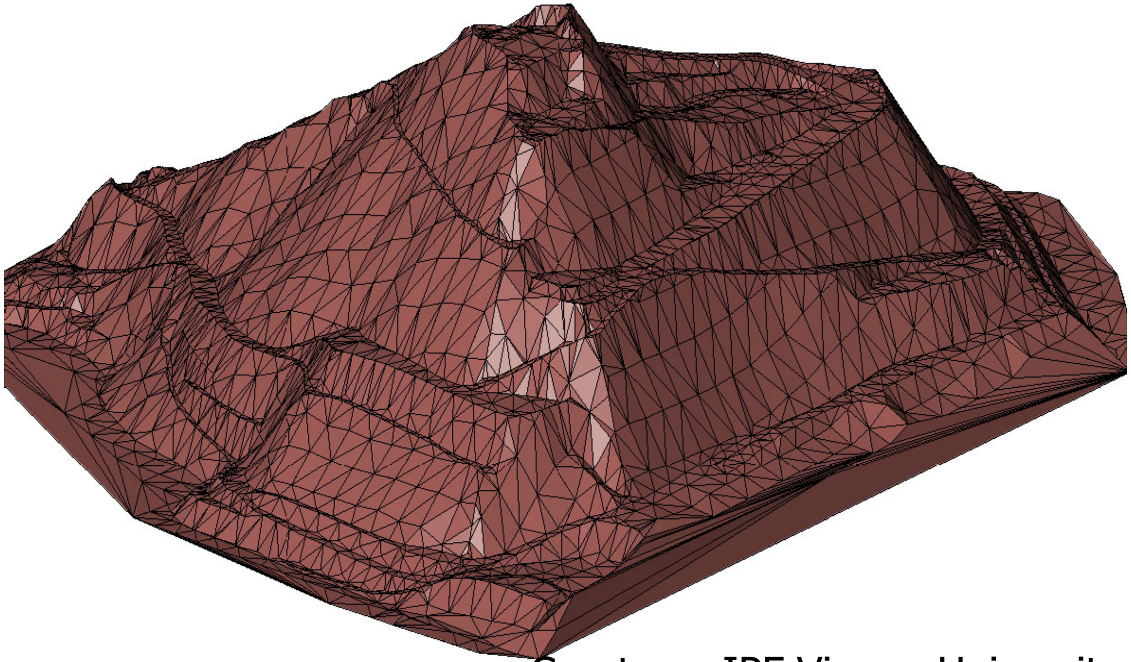
```
template < class Geometry >
class Delaunay_triangulation_2 {
    Geometry::Orientation orientation;
    Geometry::In_circle in_circle;

    void insert(Geometry::Point t) {
        ...
        if(in_circle(p,q,r,t)) {...}
        ...
        if(orientation(p,q,r){...}
    }
};
```



CGAL Genericity Demo

Without explicit conversion to points in the plane

- Triangulate the terrain in an xy-plane
- Triangulate the faces of a Polyhedron



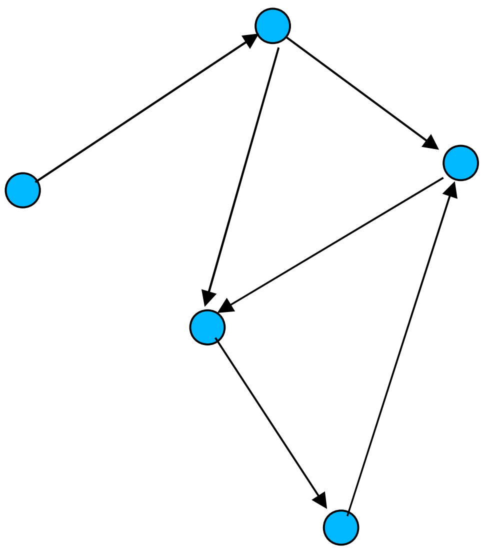
Courtesy: IPF,Vienna University
of Technology & Inpho GmbH

Boost Graph Library (BGL)

- Rich collection of graph algorithms:
  shortest paths, minimum spanning tree, flow, etc.
- Design that
  - decouples data structure from algorithm
  - links them through a thin glue layer
- BGL and CGAL
  - Provide glue layer for CGAL
  - Extension to embedded graphs
    inducing the notion of faces

BGL manual

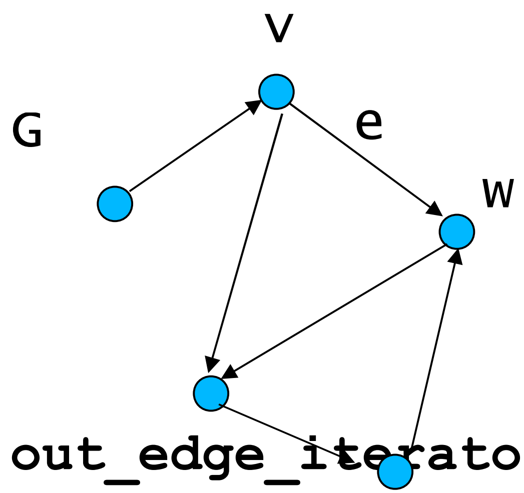
BGL Glue Layer: Traits Class



```
template <typename Graph >
struct boost::graph_traits {
     typedef ... vertex_descriptor;
     typedef ... edge_descriptor;
     typedef ... vertex_iterator;
     typedef ... out_edge_iterator;
};
```

BGL Glue Layer: Free Functions

```
vertex_descriptor v, w;
edge_descriptor e;

v = source(e,G);
w = target(e,G);

std::pair<out_edge_iterator, out_edge_iterator> ipair;

ipair = out_edges(v,G);
```



BGL Glue Layer for CGAL

CGAL provides partial specializations:

```
template <typename T>
graph_traits<Polyhedron<T>>;
```

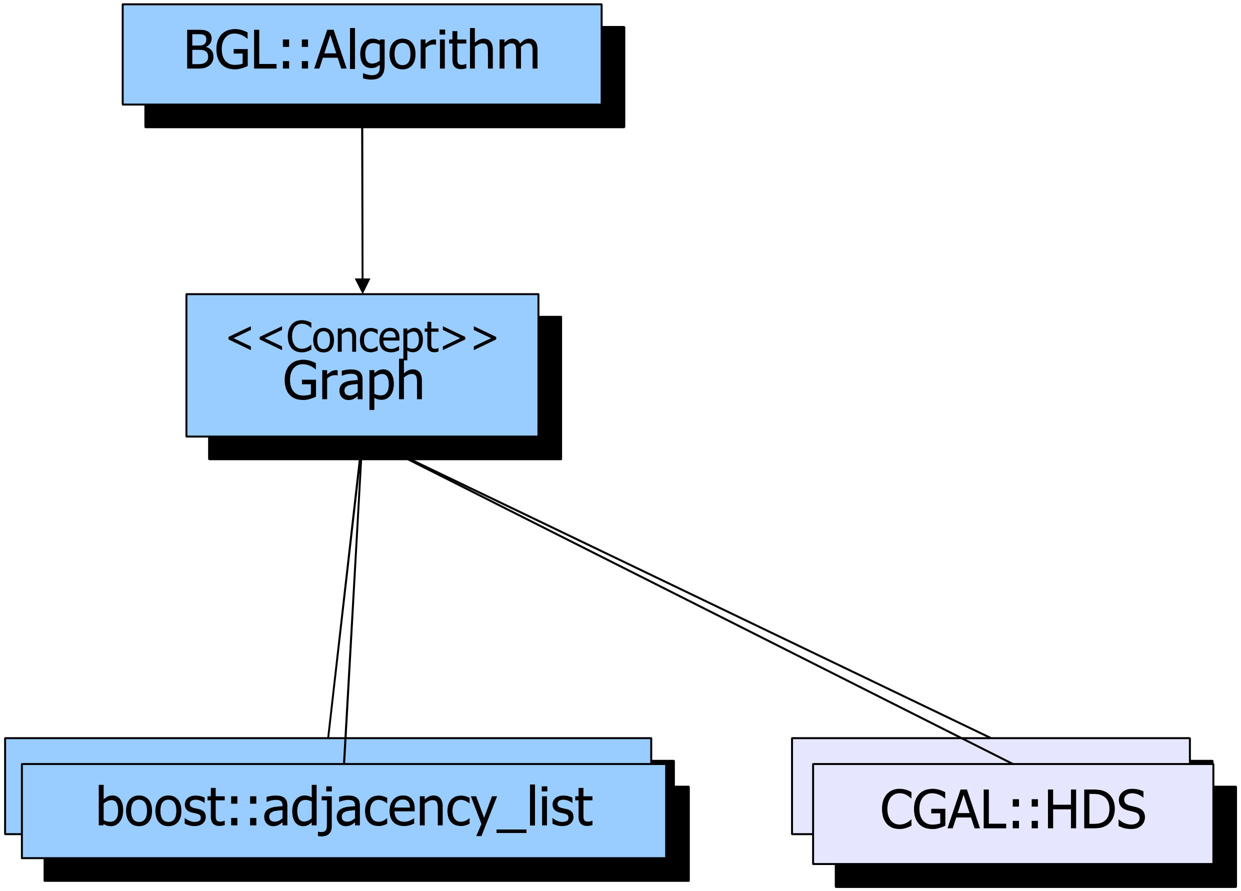
```
template <typename T>
Polyhedron<T>::Vertex
source(Polyhedron<T>::Edge);
```

Users can run:

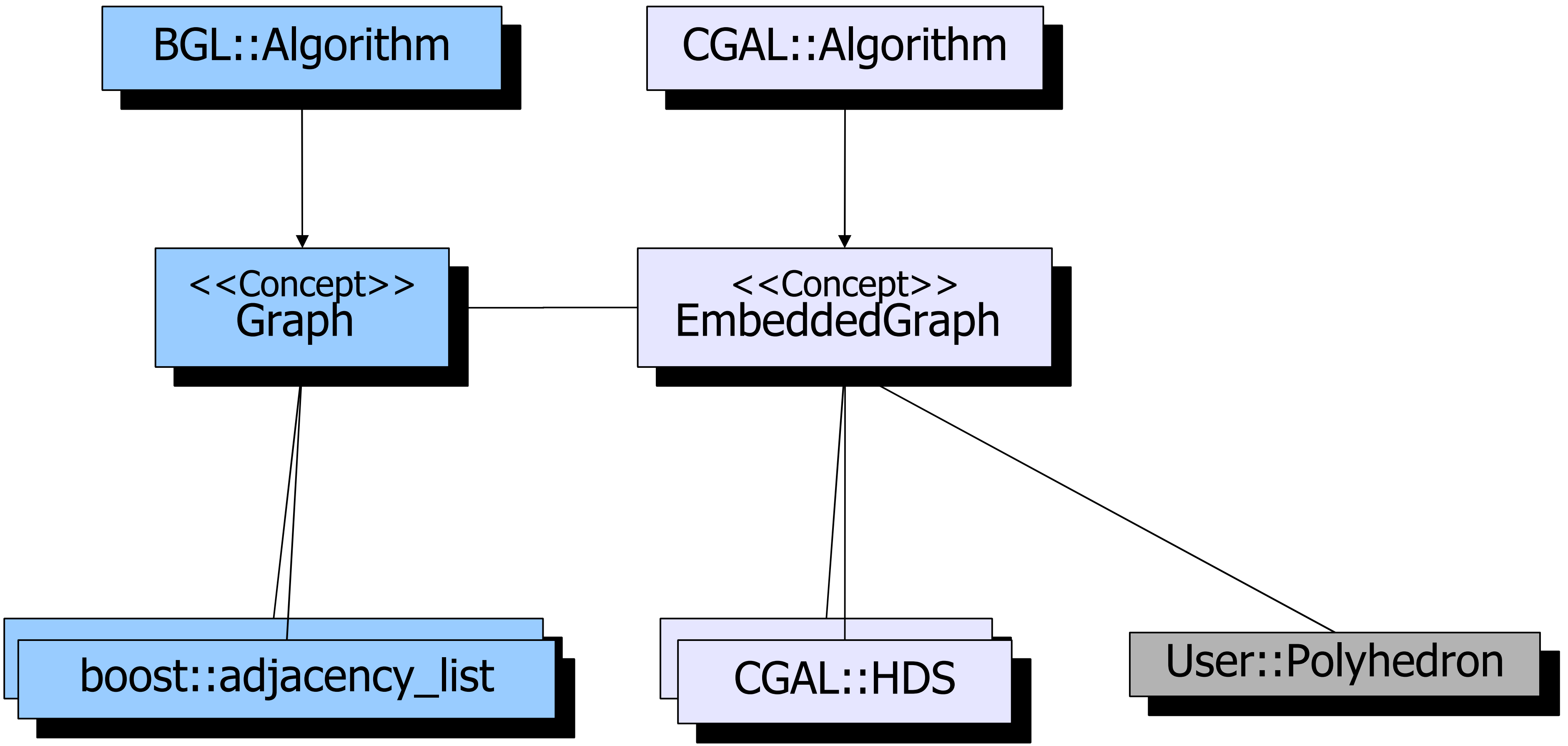
```
boost::kruskal_mst(P);
```

Courtesy: P.Schroeder, Caltech

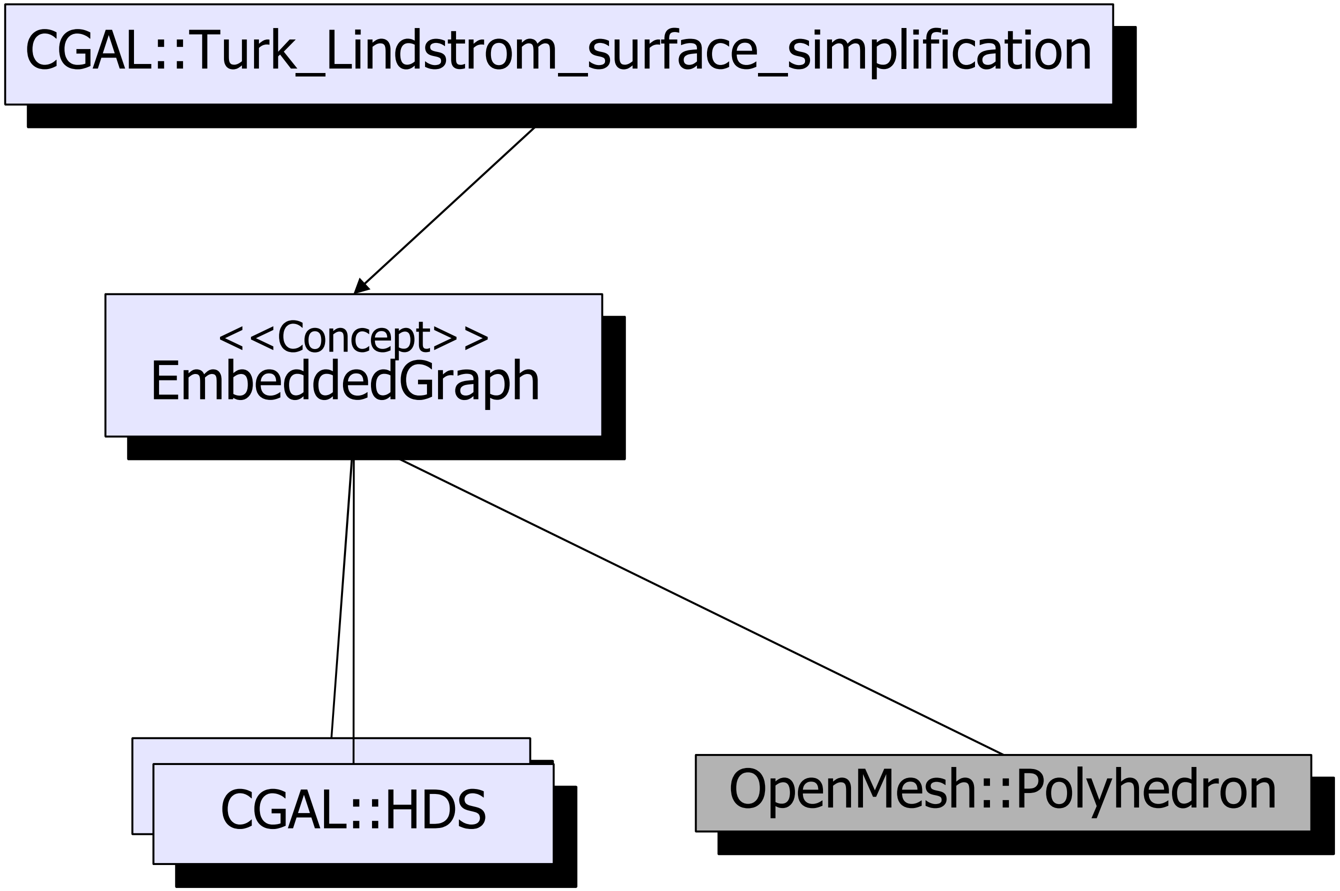
From A BGL Glue Layer for CGAL



To BGL Style CGAL Algorithms



Demo CGAL + OpenMesh

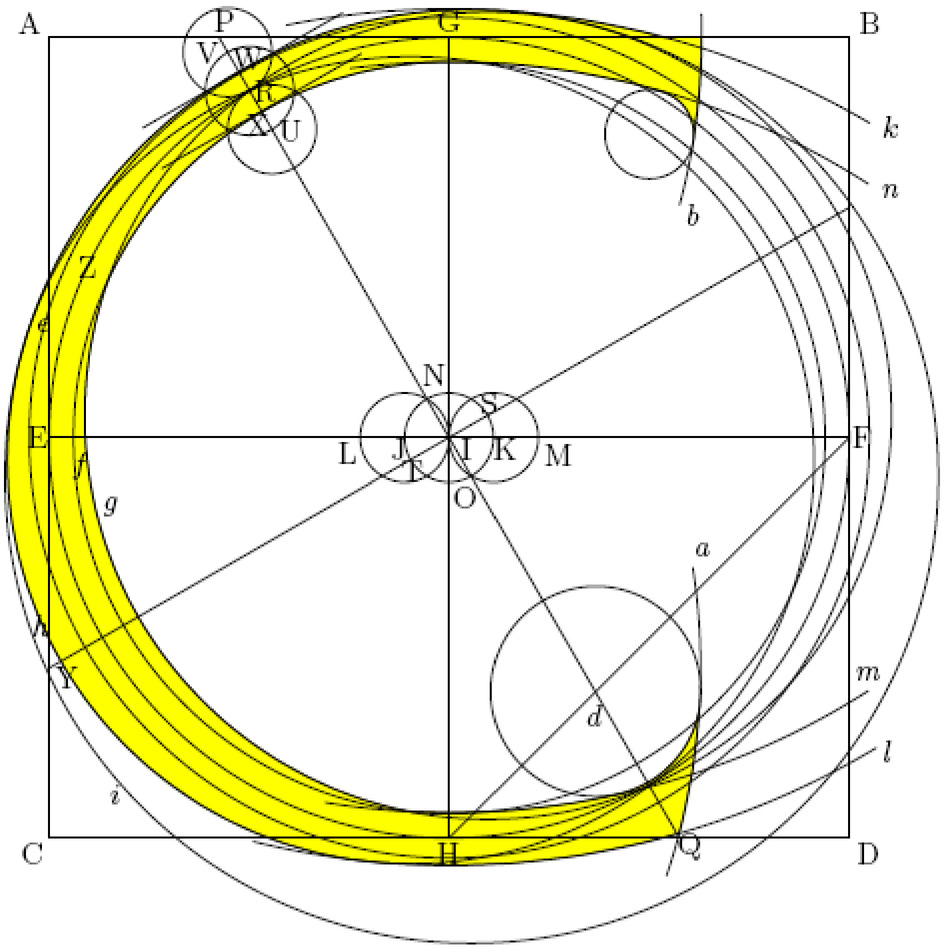


Summary: Overview

- Open Source project

- Clear focus on geometry
- Interfaces with de facto standards/leaders: STL, Boost, GMP, Qt, blas

- Robust and fast through exact geometric computing
- Easy to integrate through generic programming



CGAL for 2D Vector Graphics

Andreas Fabri

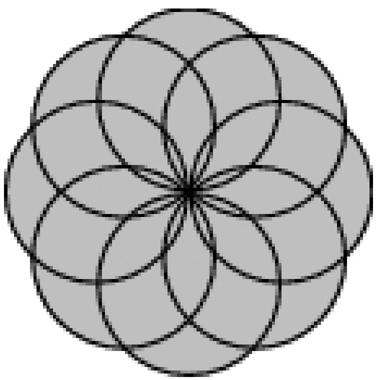
GeometryFactory

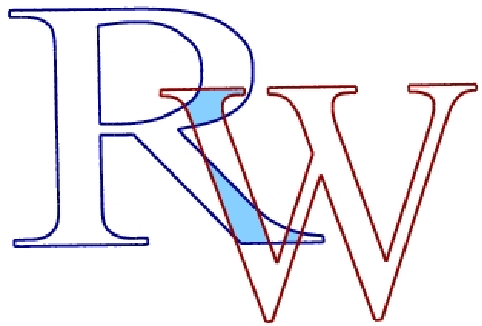
Boolean Operations

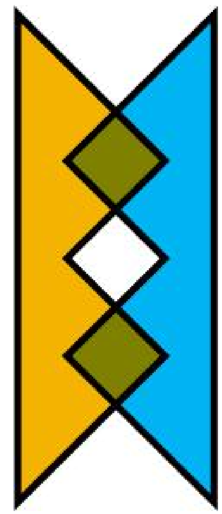
Union

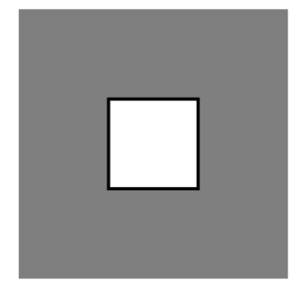
Intersection

Complement







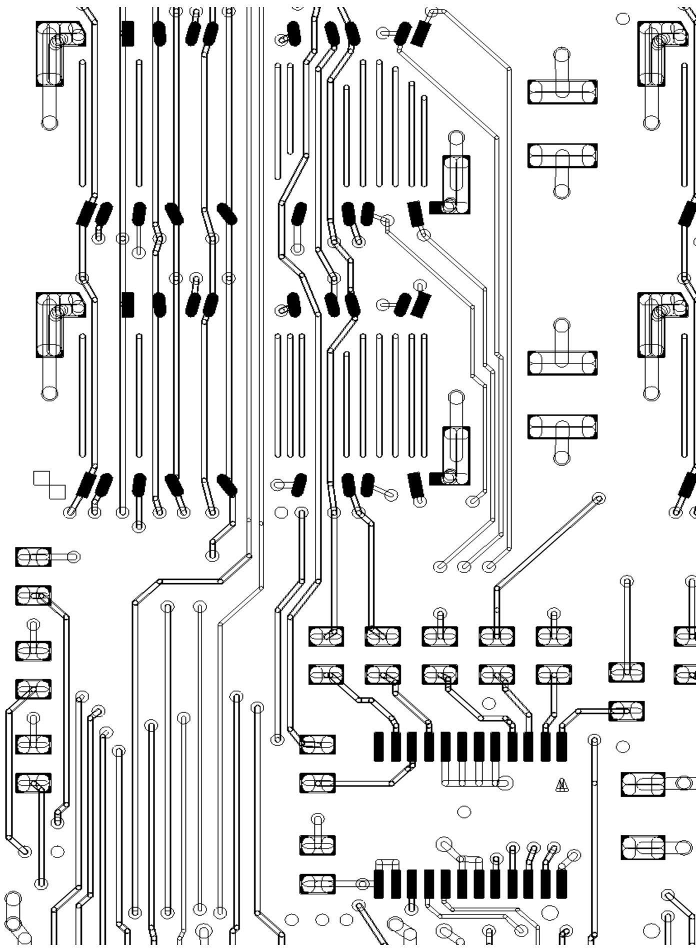


CGAL Boolean Operations can deal explicitly with

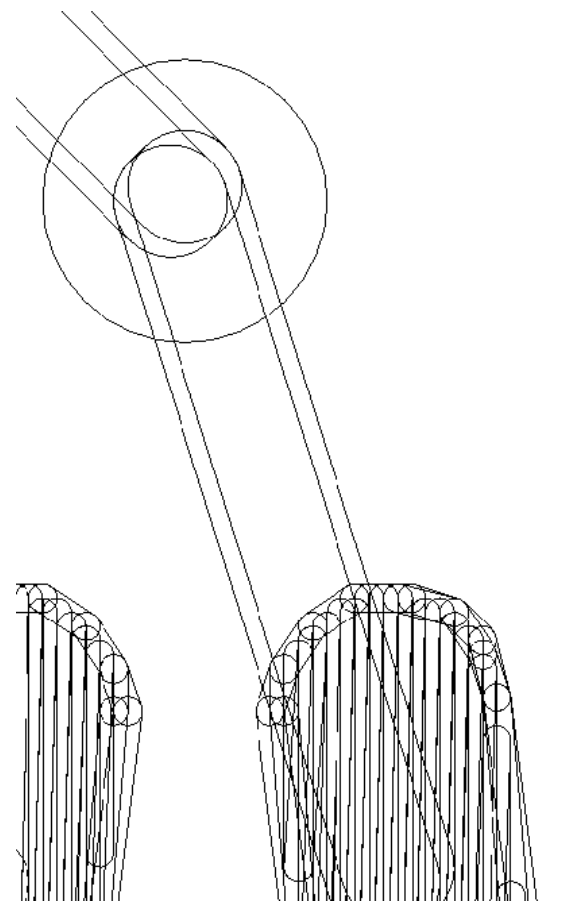
Circular arcs   Bézier Curves   Line Segments

Boolean Ops on Circular Arcs Demo

dxf file of a printed circuit board with circular arcs

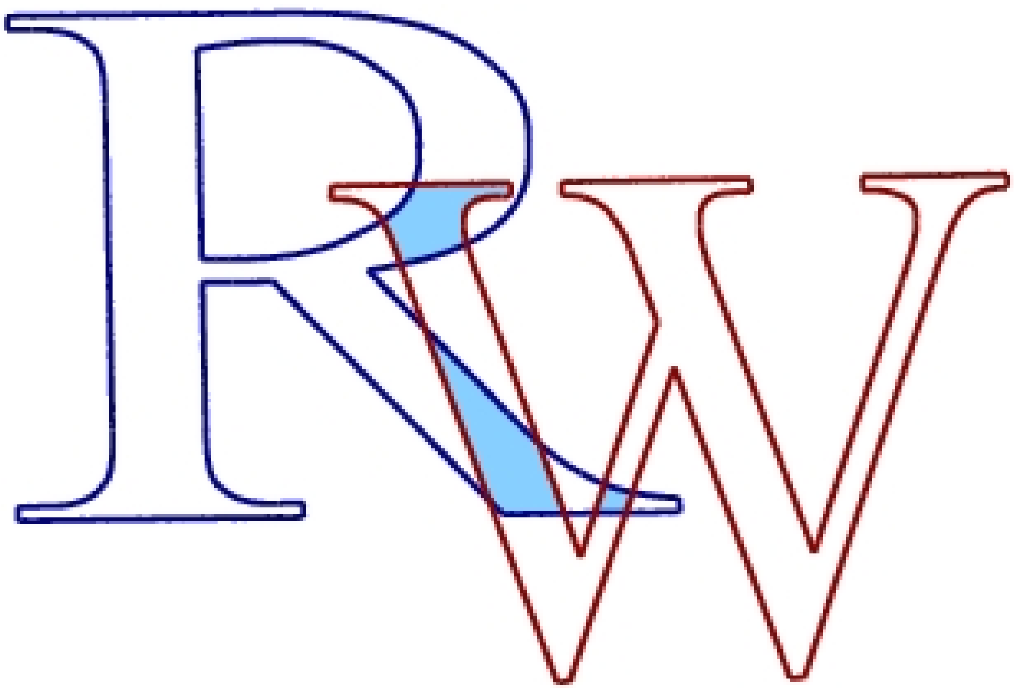


Zoom in



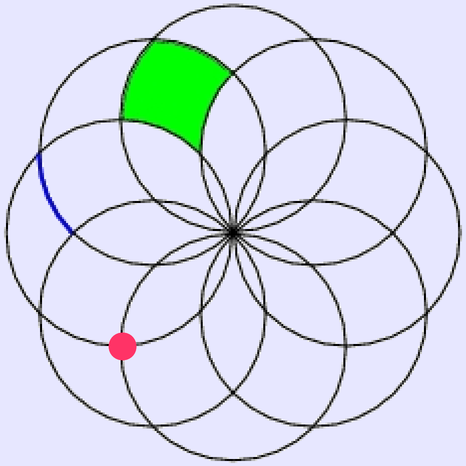
Boolean Ops on Bézier Curves

True Type fonts are Bézier curves

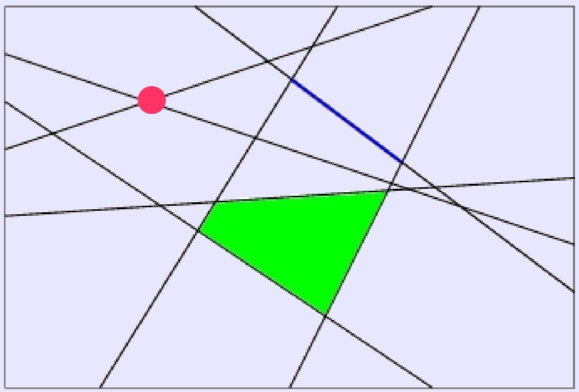


Background: 2D Arrangement

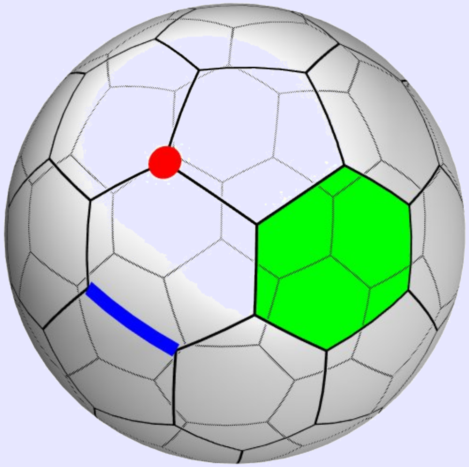
Given a collection of curves on a surface, the **arrangement** is the partition of the surface into vertices, edges and faces induced by the curves



An arrangement of circles in the plane



An arrangement of lines in the plane



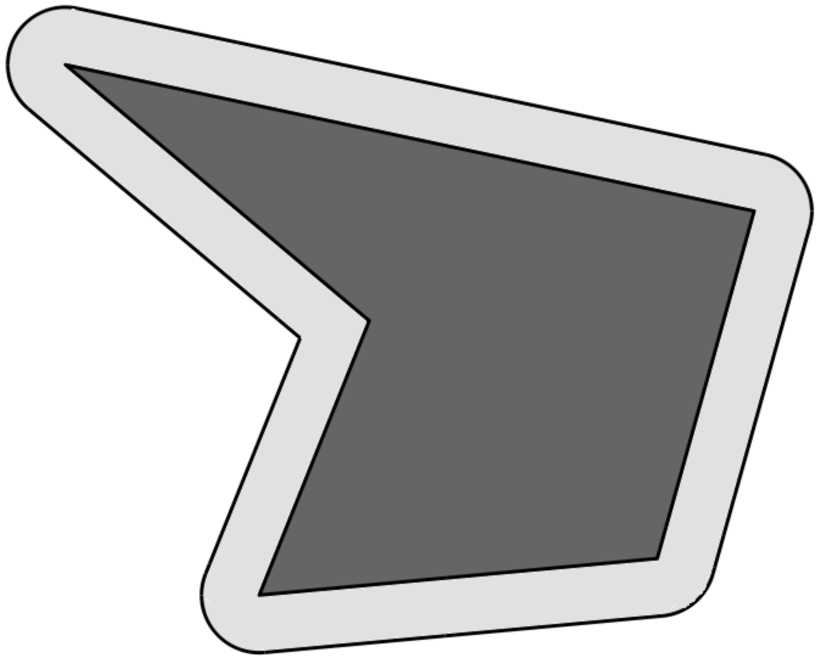
An arrangement of geodesic arcs on the sphere

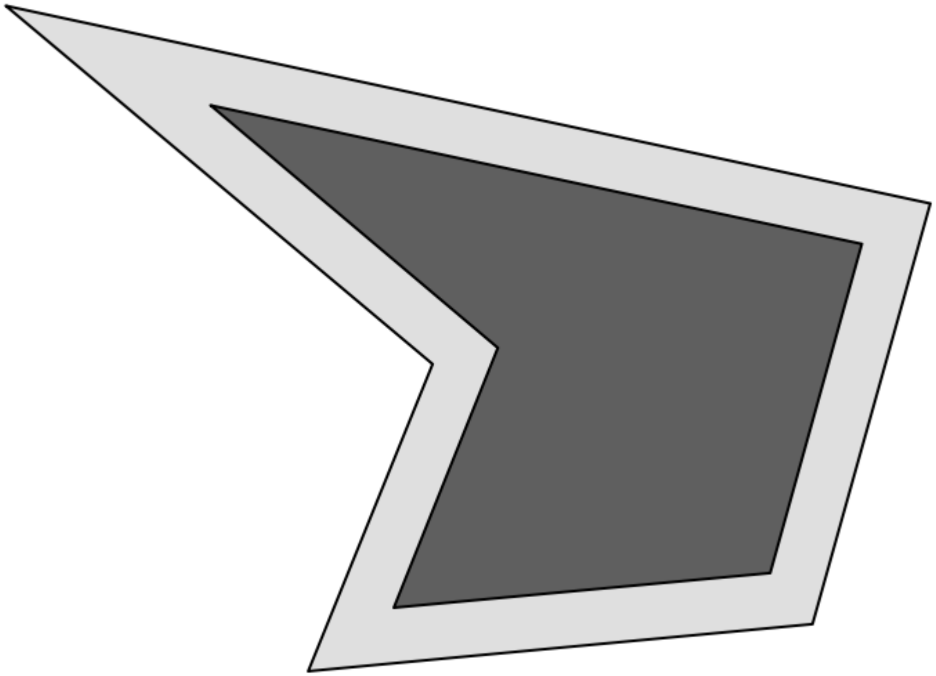
Arrangement_2<Geometry>

- Constructs, maintains, modifies, traverses, queries, and presents subdivisions of the plane
- Robust and exact
  - All inputs are handled correctly (including degenerate)
  - Exact number types are used to achieve exact results
- Efficient
- Generic
  - Easy to interface, extend, and adapt
  - Notification mechanism for change propagation
- Modular
  - *Geometric* and *topological* aspects are separated

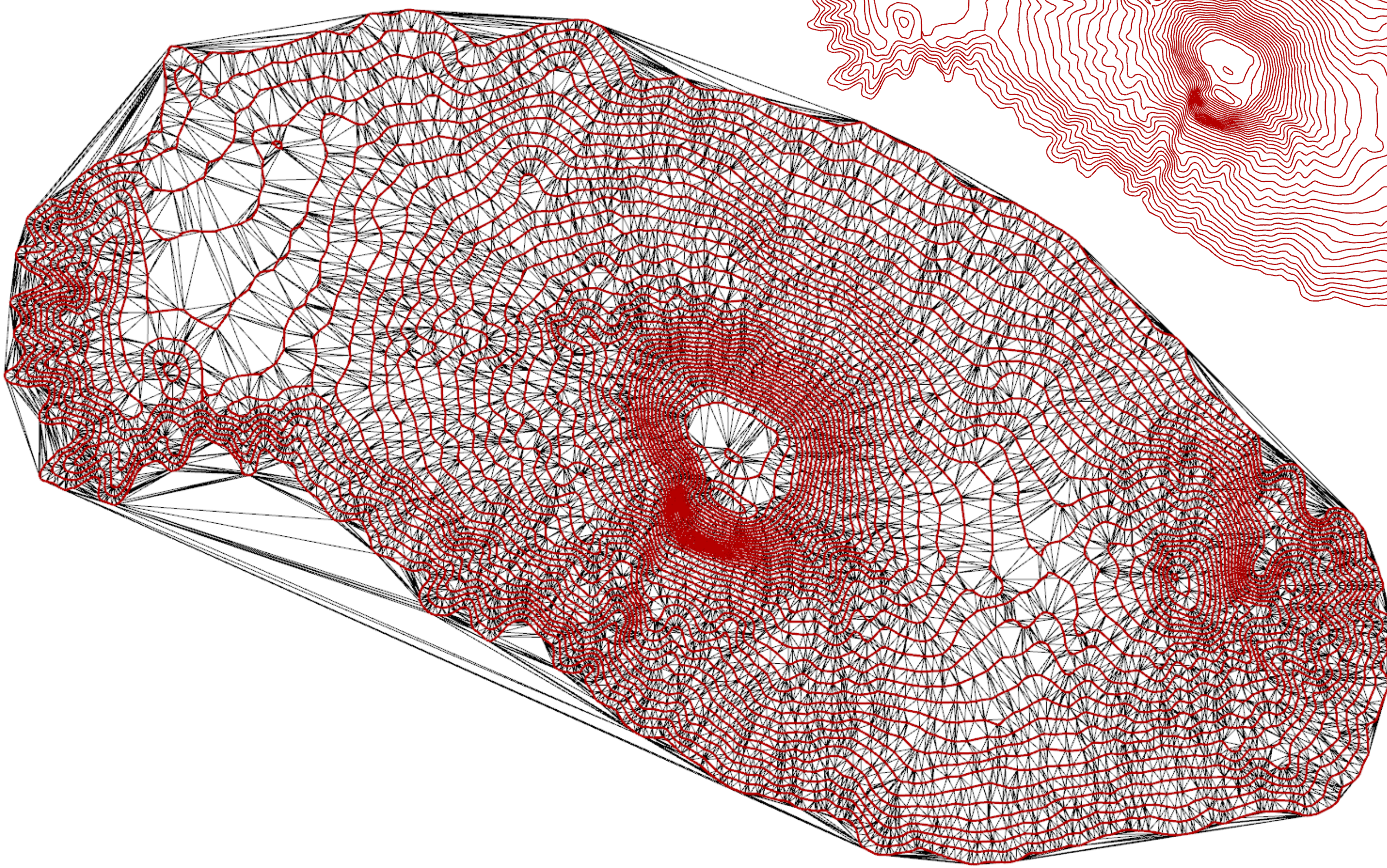
Polygon Offsets

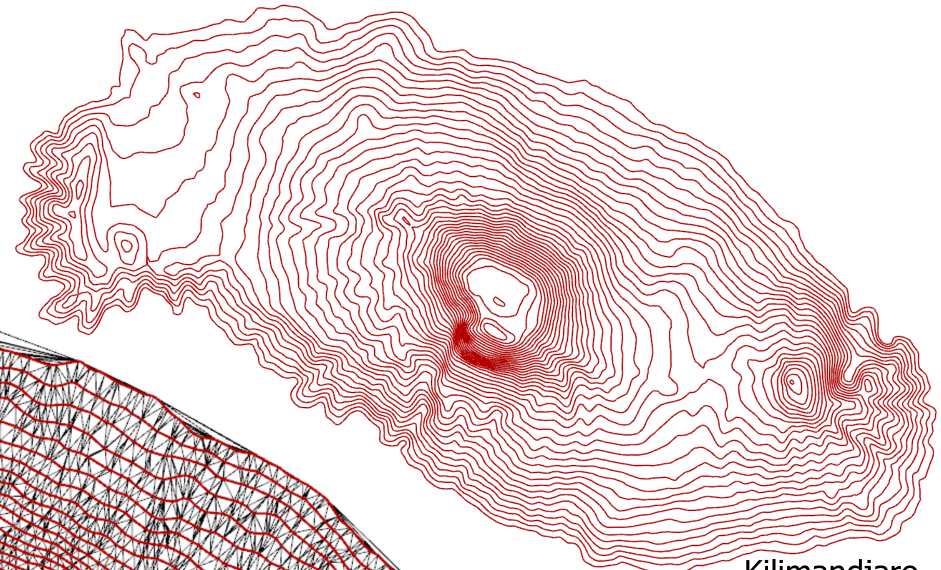
- Based on Minkowski sums, with segments and circular arcs.
- Based on straight skeleton, with segments only.





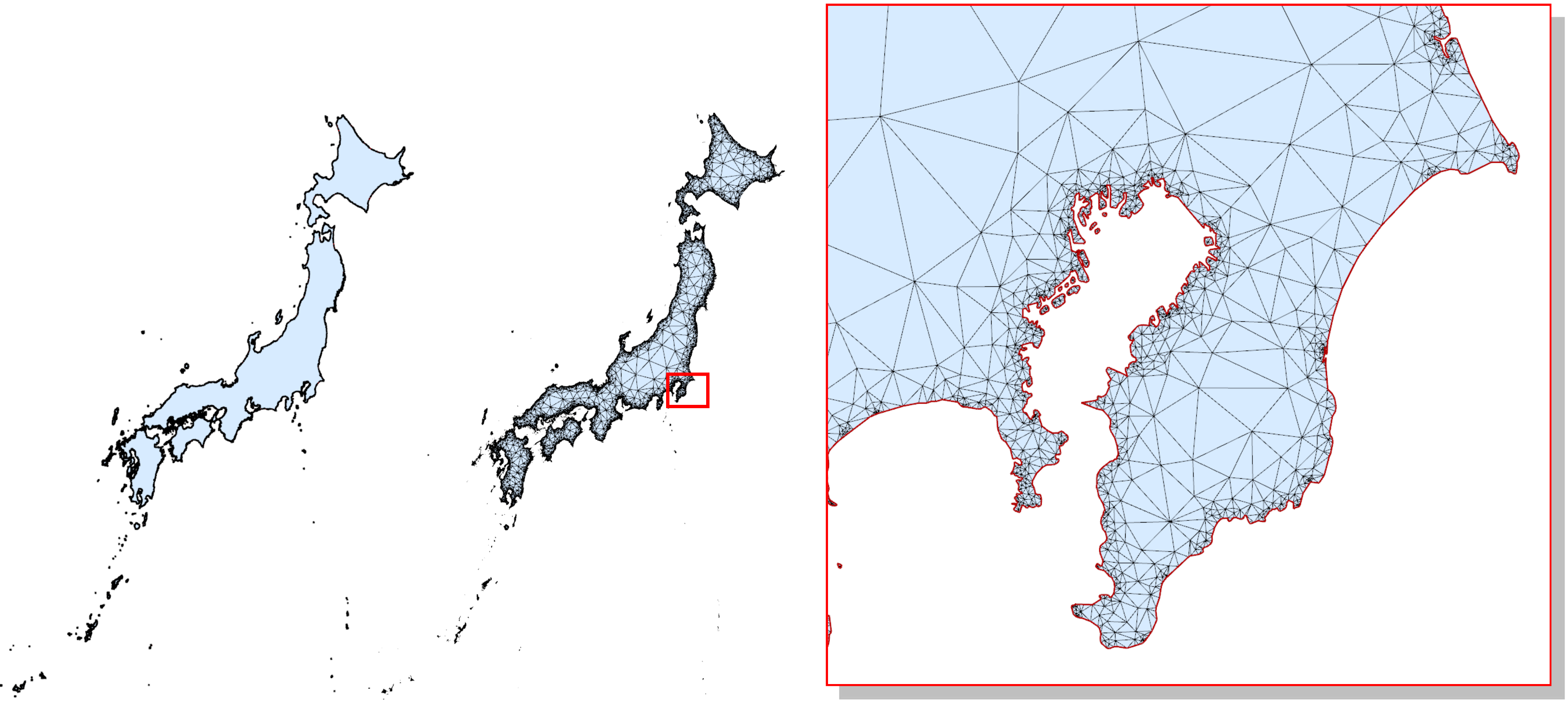
Polygon Triangulation Demo





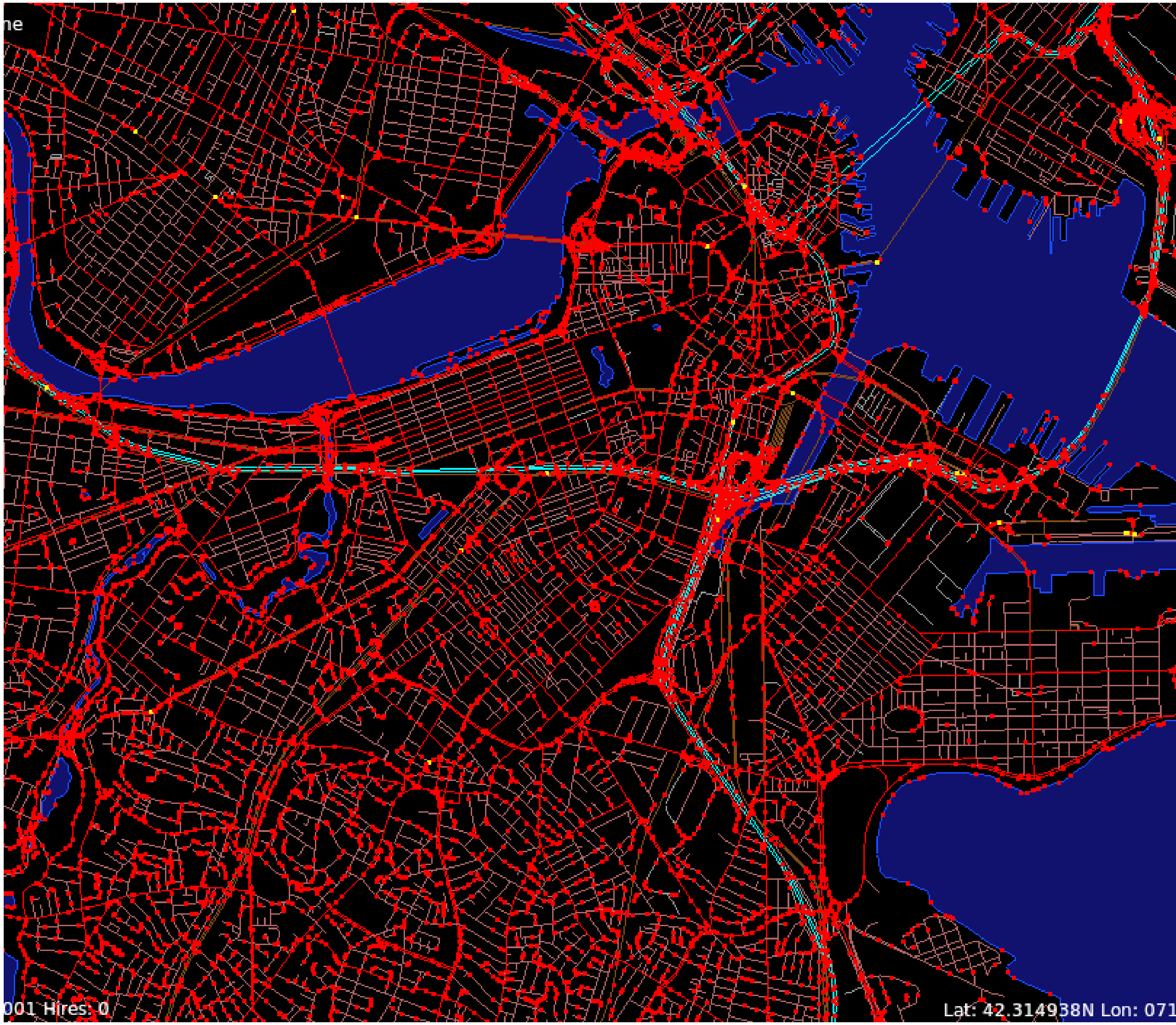
Kilimandjaro elevation contour lines (38K segments)

Polygon Mesh Generation Demo



Simultaneous Polyline Simplification

Red points were removed

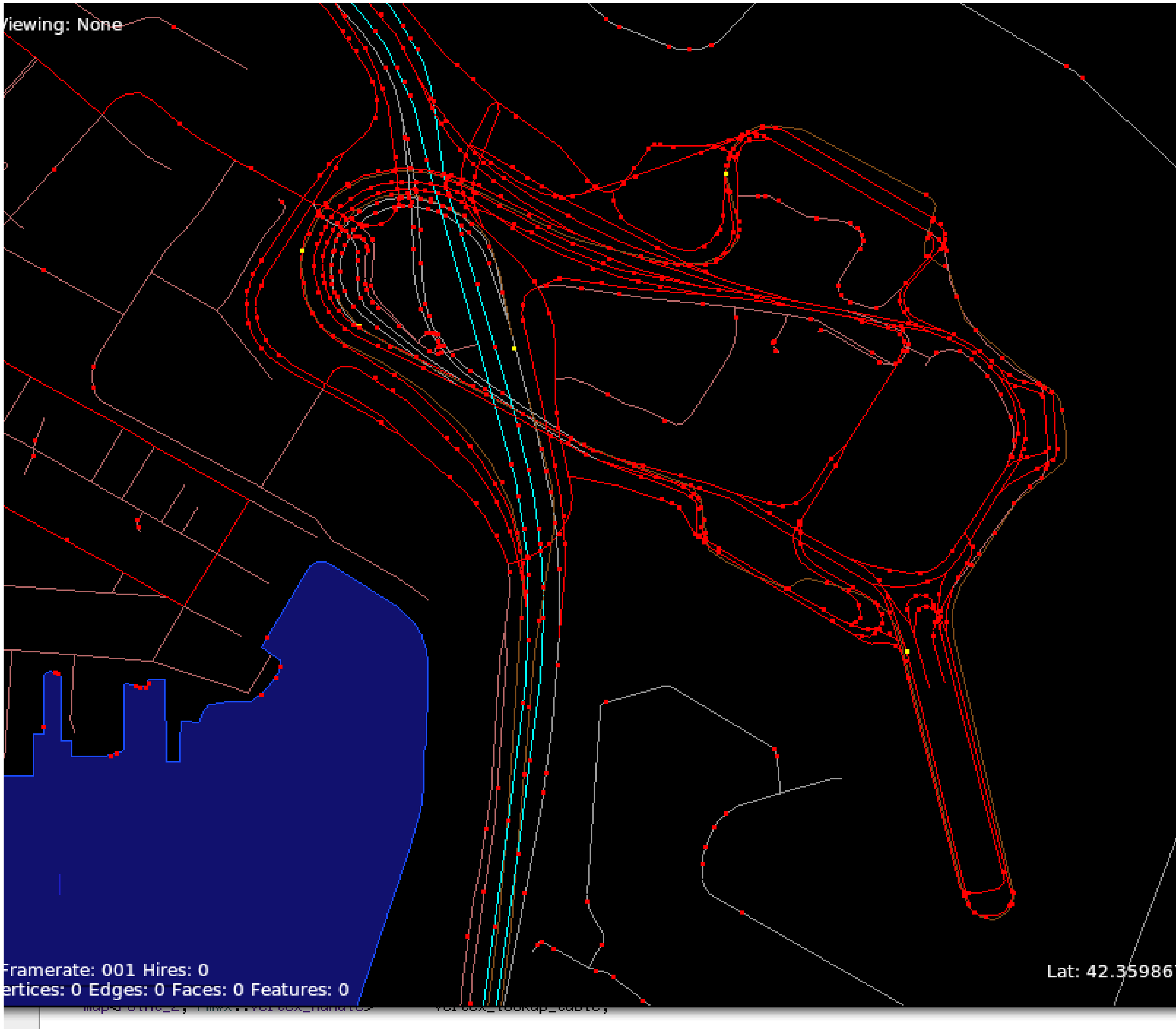


Courtesy: Laminar Research

Simultaneous Polyline Simplification

Input is the transportation and water layers of OpenStreetMap

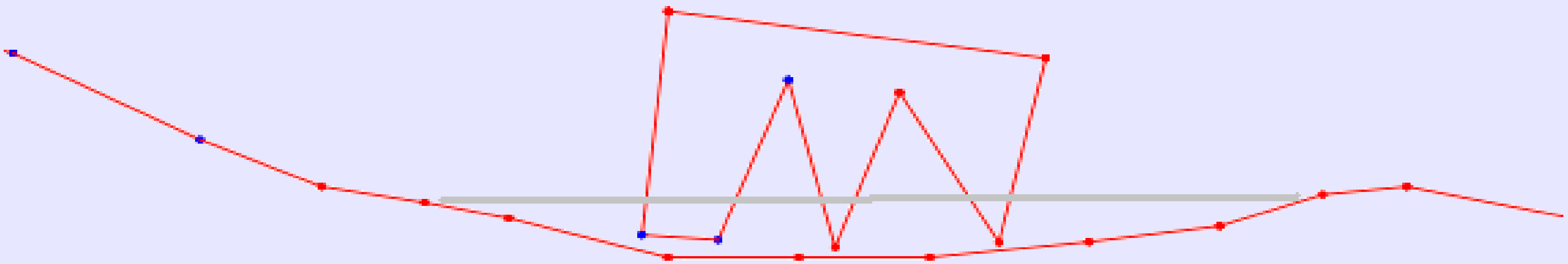
Red points were removed



Courtesy: Laminar Research

Simultaneous Polyline Simplification

- Implementation of [Dyken et al]
- Based on CGAL::Constrained_delaunay_2
- Guarantees that after simplification
  - islands stay islands
  - isolines do not intersect



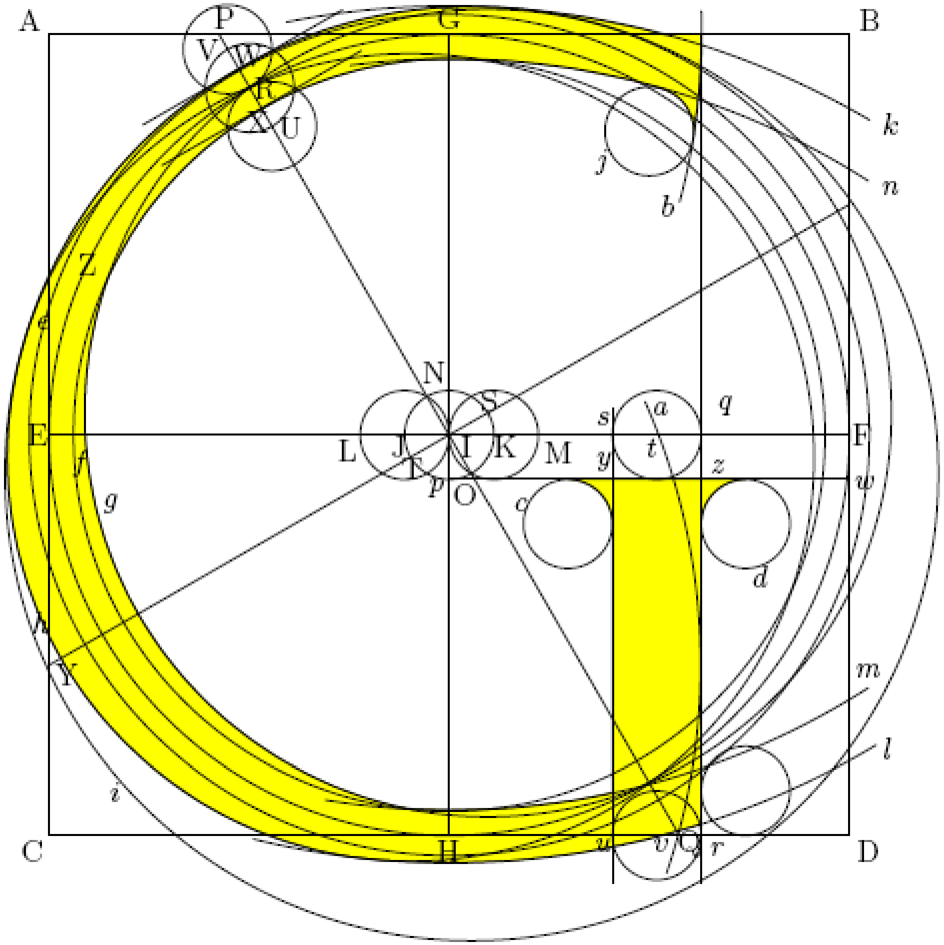
Vector Graphics on the Sphere

- Arrangement_2<Geometry, Embedding>

- Boolean operations
- Map overlay
- Voronoi diagram
- Point location
- Convex decomposition

Summary: CGAL for Vector Graphics

- Rich collection of 2D geometric algorithms
- Modular and generic design
- Linear and curved primitives
- Useful in many application domains

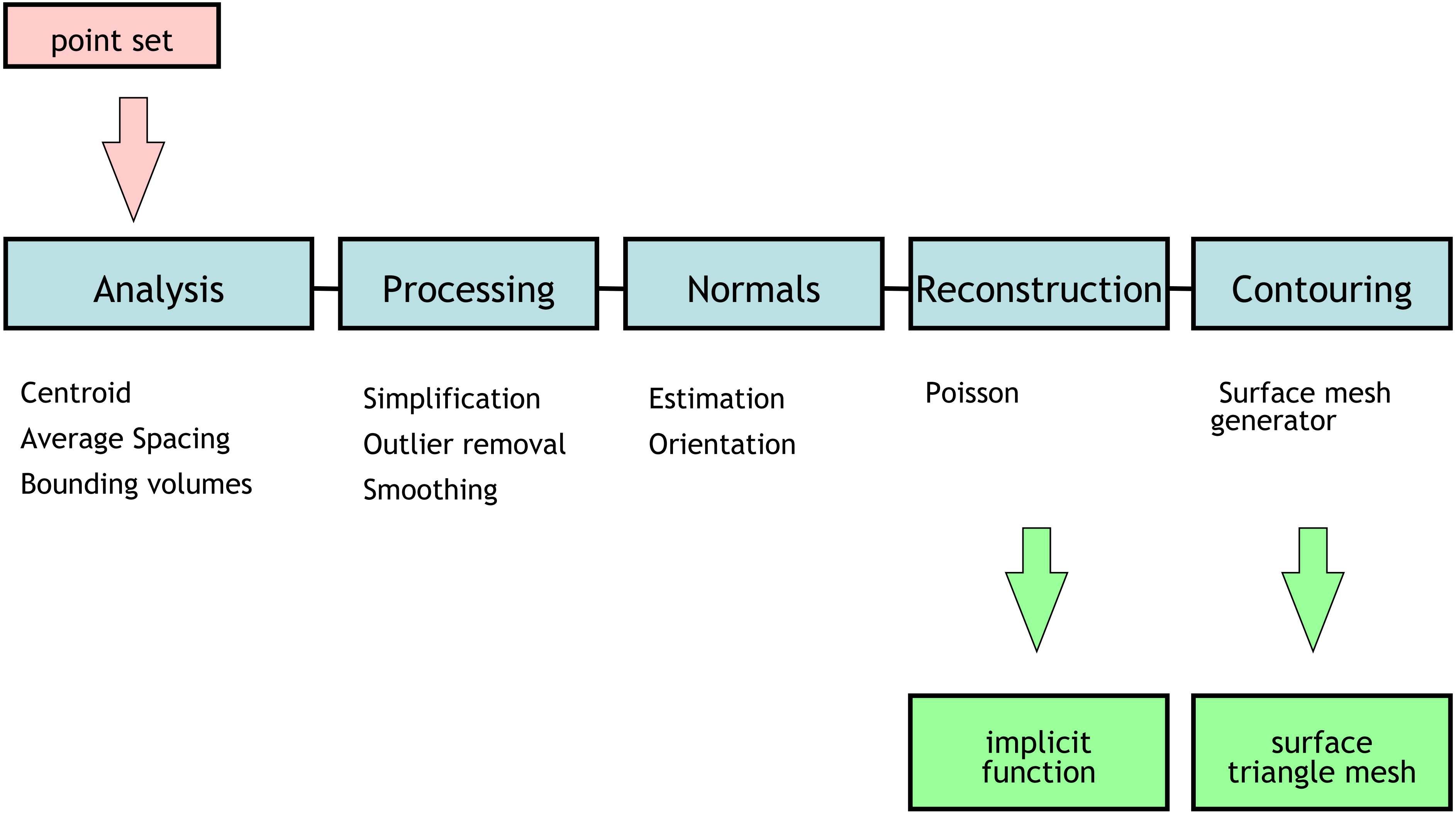


CGAL for Point Sets

Pierre Alliez

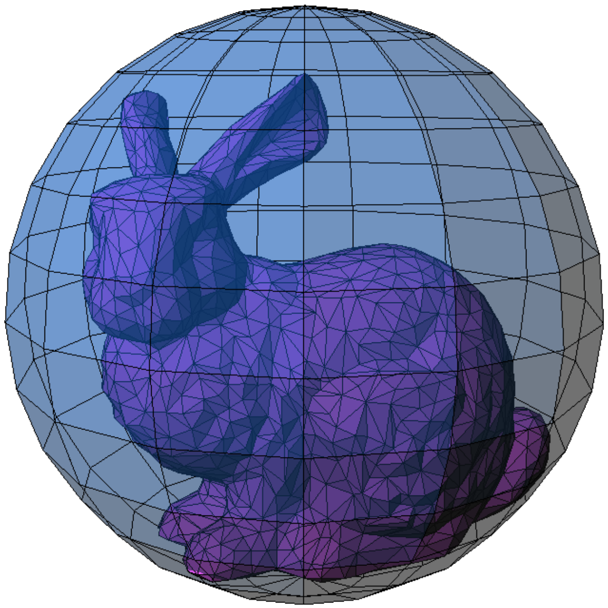
INRIA

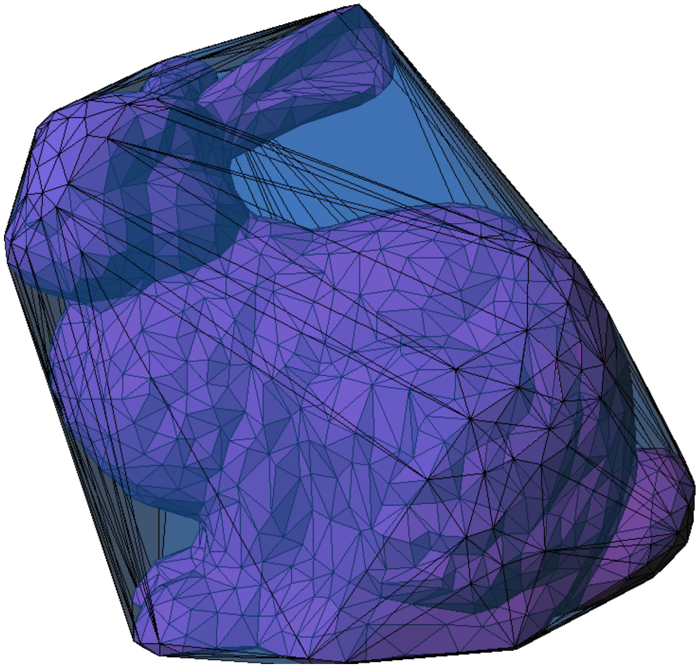
Surface Reconstruction Pipeline



Bounding Volumes

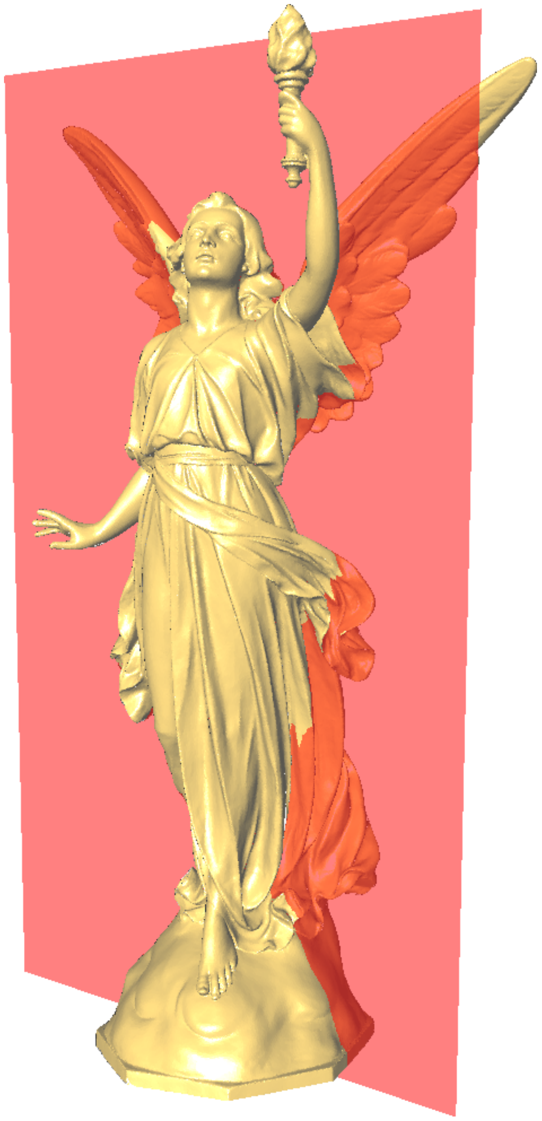
- Convex hull
- Bounding sphere





Principal Component Analysis

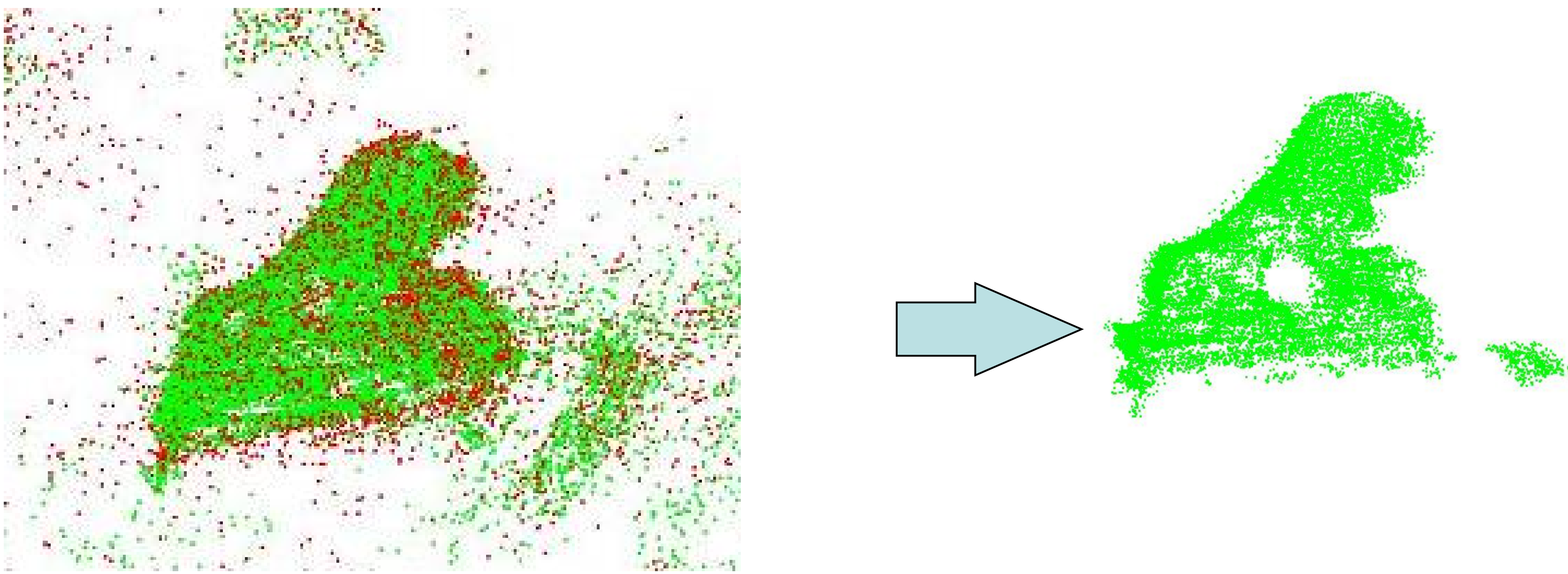
Linear least squares fitting on sets of 3D points



CGAL manual

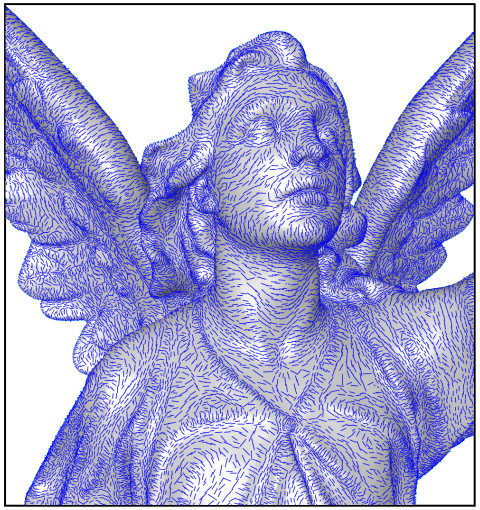
Outlier Removal

- Sort w.r.t. sum of squared distances to k-nearest neighbors (CGAL::K_nearest_neighbor_search) and cut at specified percentile.



Estimation of Curvatures

- Estimates general differential properties (Monge form) on point sets.
- Through polynomial (d-jet) fitting



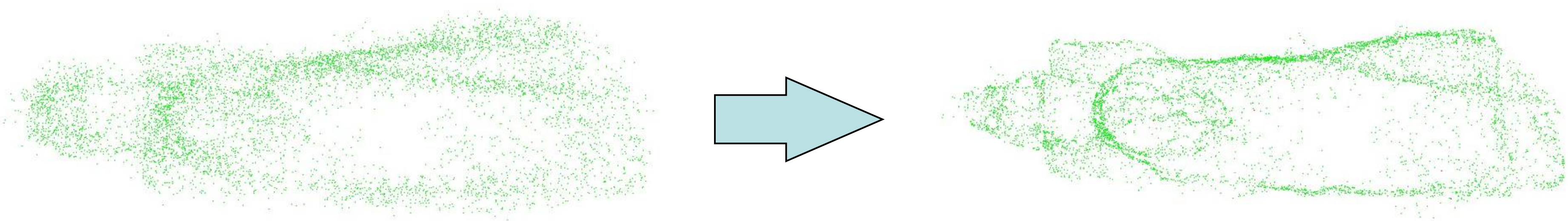
min curvature directions

max curvature directions

CGAL manual

Point Cloud Smoothing

- For each point
  - Find k-nearest neighbors
  - fit jet (smooth parametric surface)
  - project onto jet

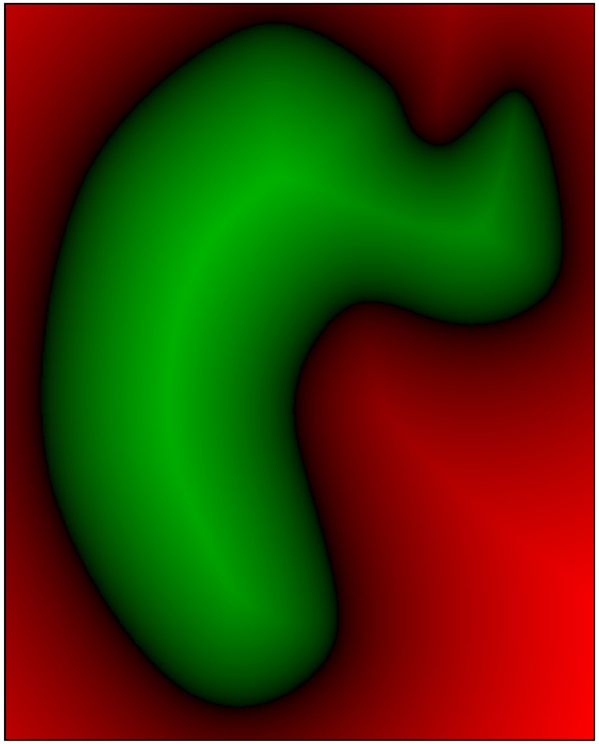


(noisy point set) (smoothed point set)

Surface Reconstruction

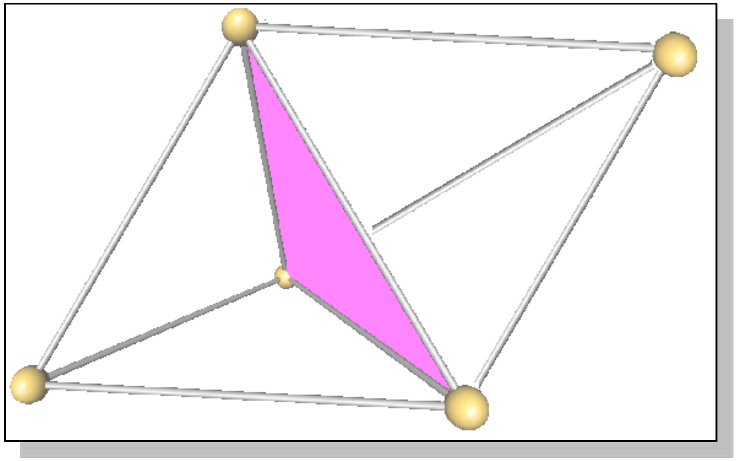
Poisson Surface Reconstruction [Kazhdan-Bolitho-Hoppe, SGP 2006]

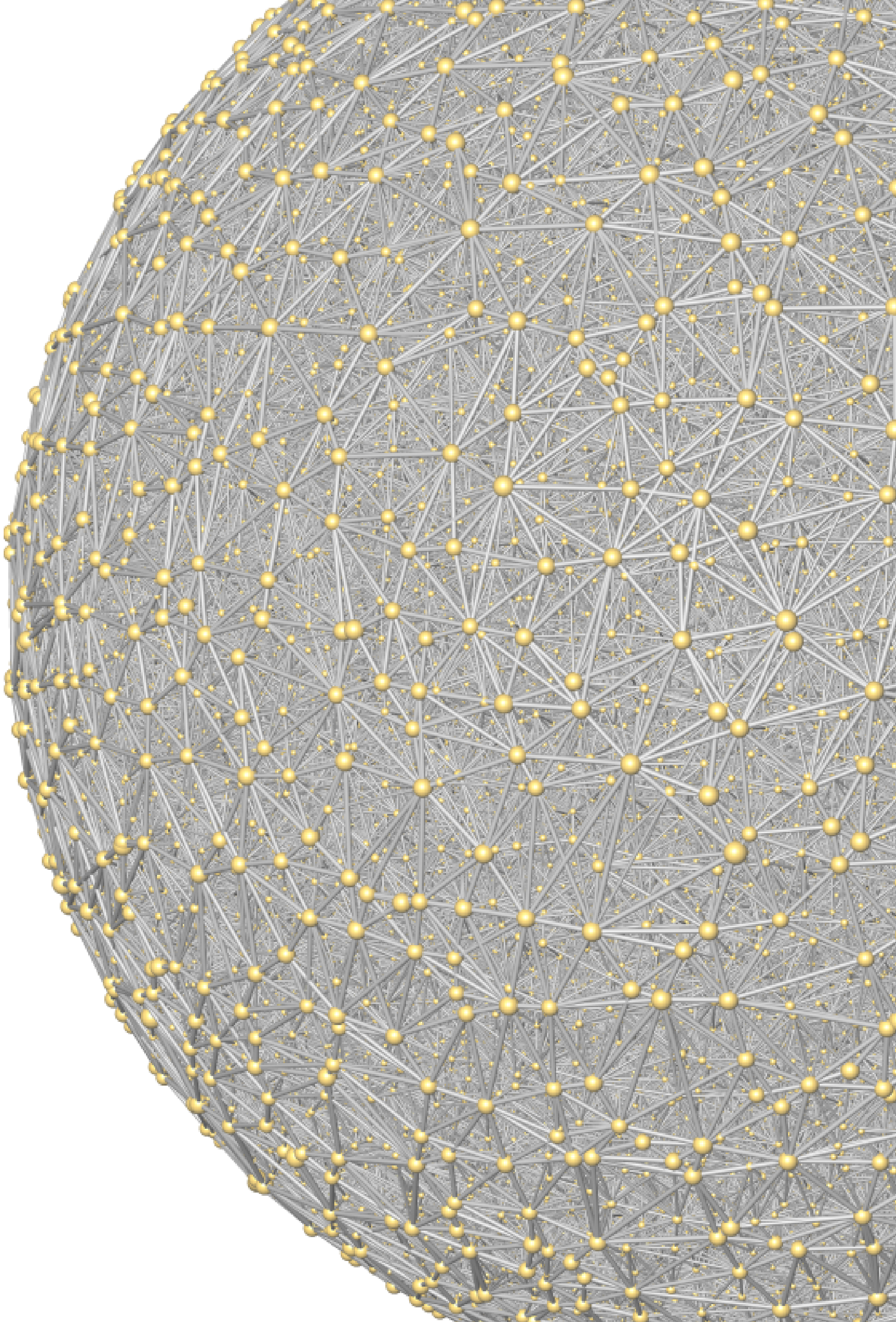
- Solves for an implicit function (~indicator function)
- Isosurface extracted by CGAL::Surface_mesher



3D Triangulations

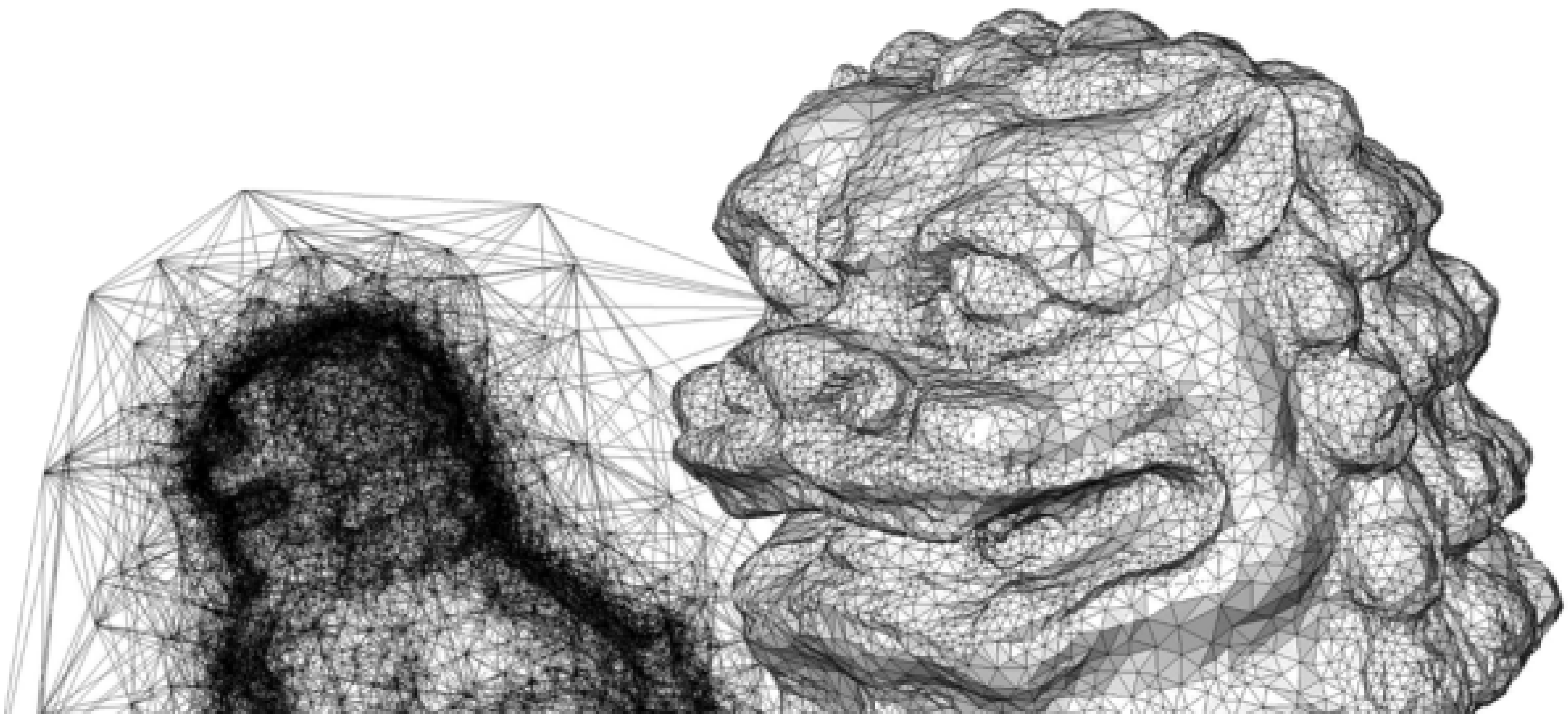
- Delaunay
- Fully dynamic
- 1M 3D points in 16 sec





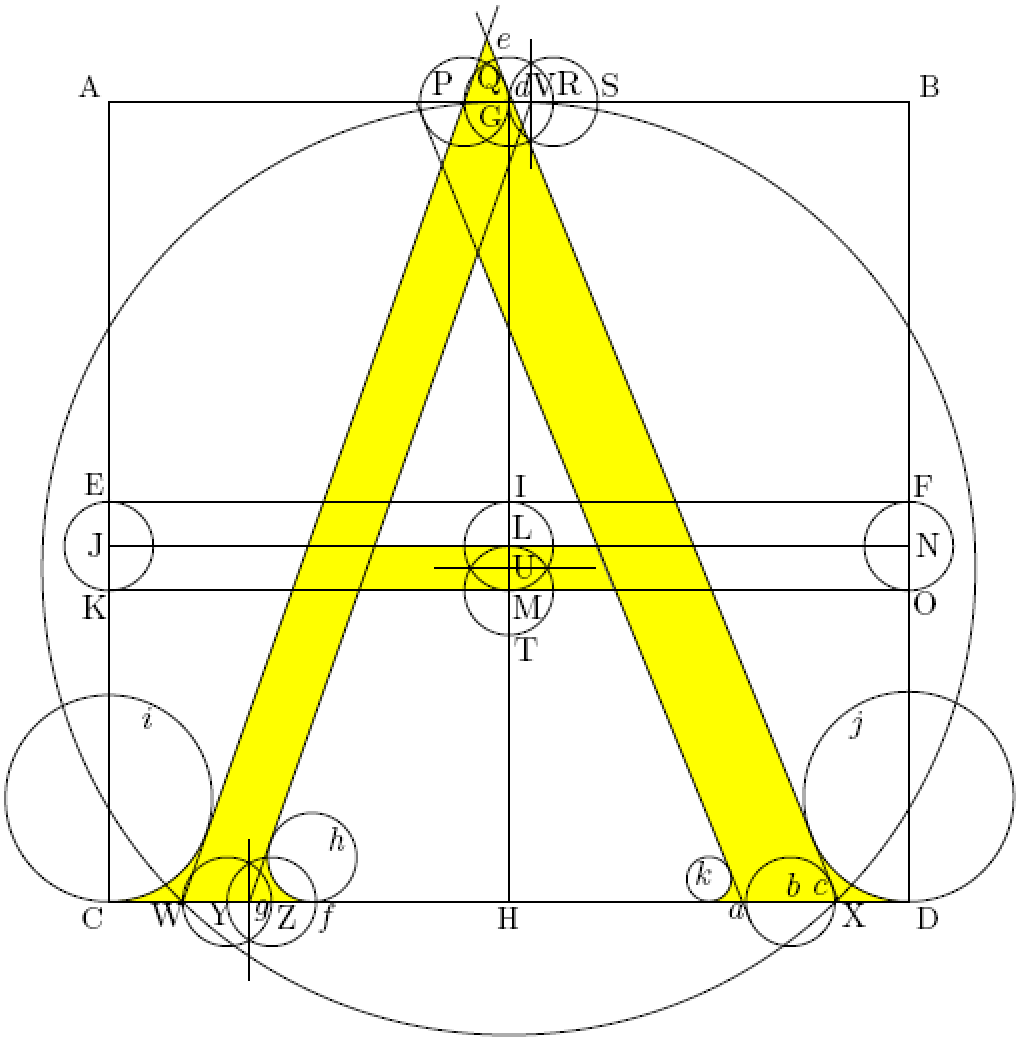
Surface Reconstruction Demo

- Solves for the Poisson equation onto the vertices of a (refined) 3D Delaunay triangulation.



Summary: CGAL for Point Sets

- Algorithms are modular components
  - in this course: positioned along the surface reconstruction pipeline.
  - can be used individually
- Poisson reconstruction is the first algorithm of the surface reconstruction package.



CGAL for Modeling and Processing of Polyhedral Surfaces

Andreas Fabri

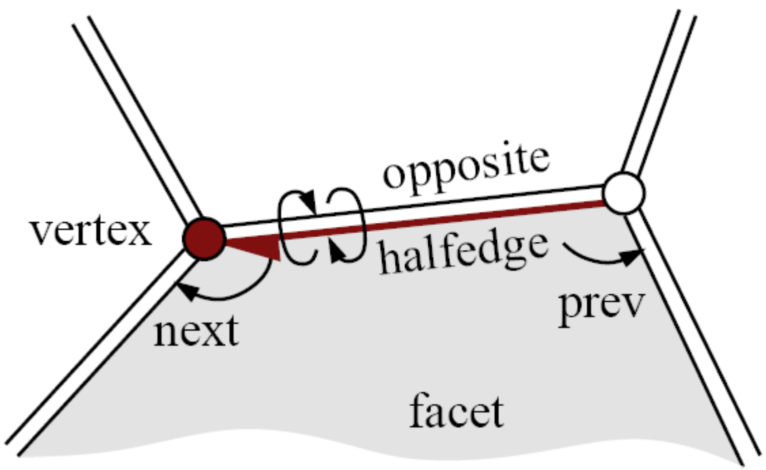
GeometryFactory

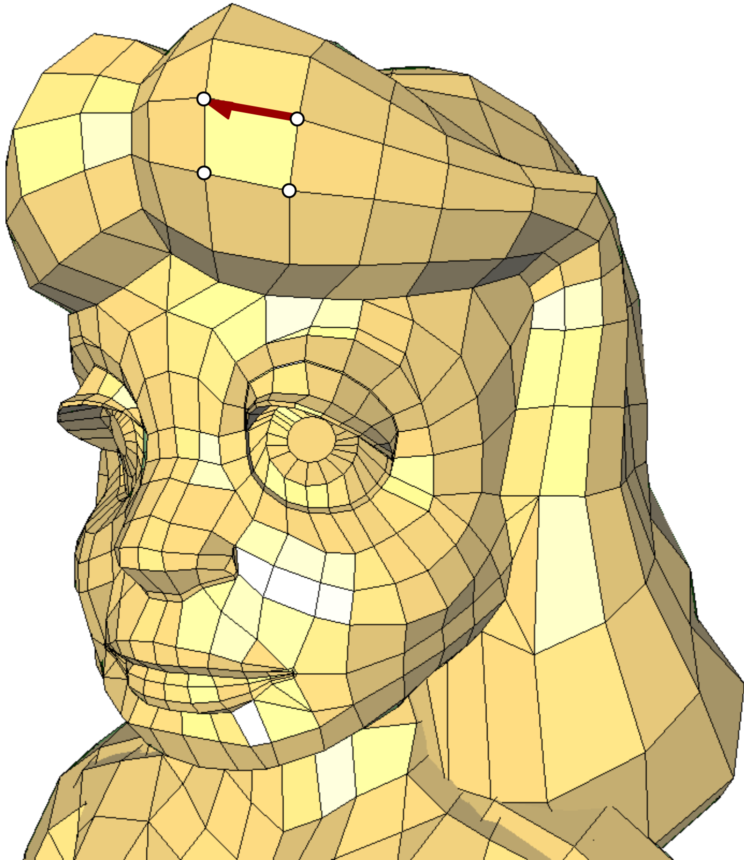
Outline

- Polyhedral Surface
  - Halfedge data structure
  - Euler Operators
  - Customization
- Algorithms for Geometric Modelling and Geometry Processing

Halfedge Data Structure

Represented by vertices, edges, facets and an **incidence relation** on them, restricted to orientable 2-manifolds with boundary.

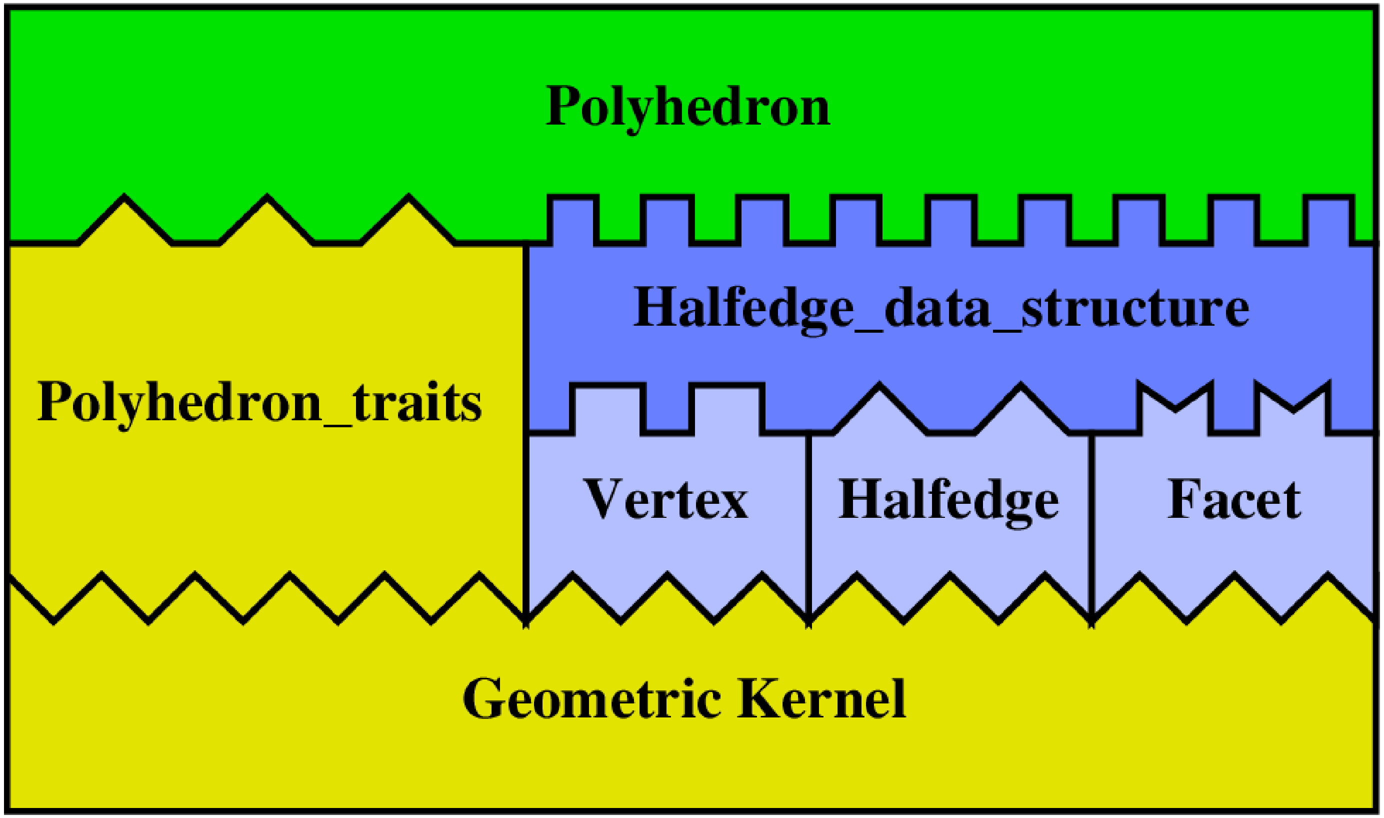




CGAL manual

Polyhedron

Building blocks assembled with C++ templates

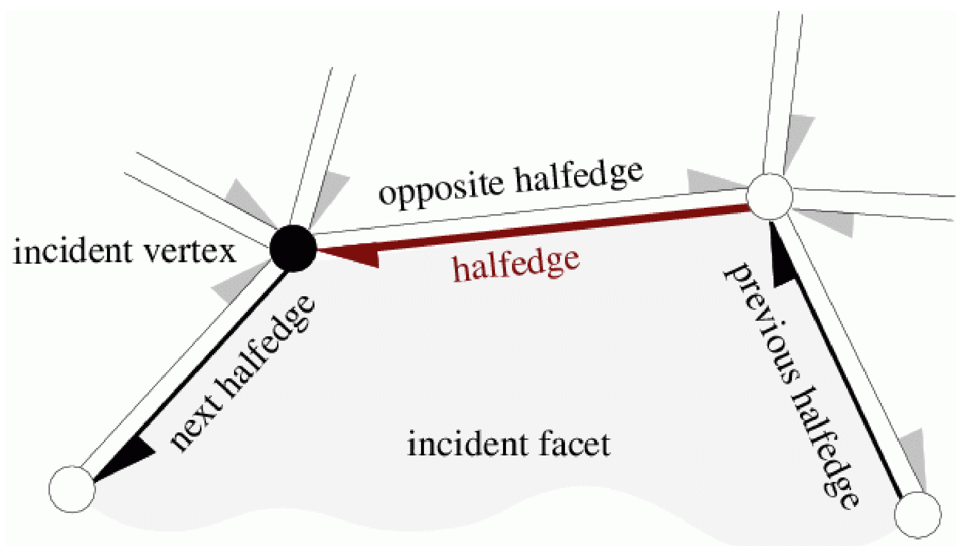


Default Polyhedron

Vertex	
Halfedge_handle	halfedge()
Point&	point()
......	...

Halfedge	
Halfedge_handle	opposite()
Halfedge_handle	next()
Halfedge_handle	prev()
Vertex_handle	vertex()
Facet_handle	facet()
......	...

Facet	
Halfedge_handle	halfedge()
Plane&	plane()
Normal&	normal()
Color&	color()
......	...



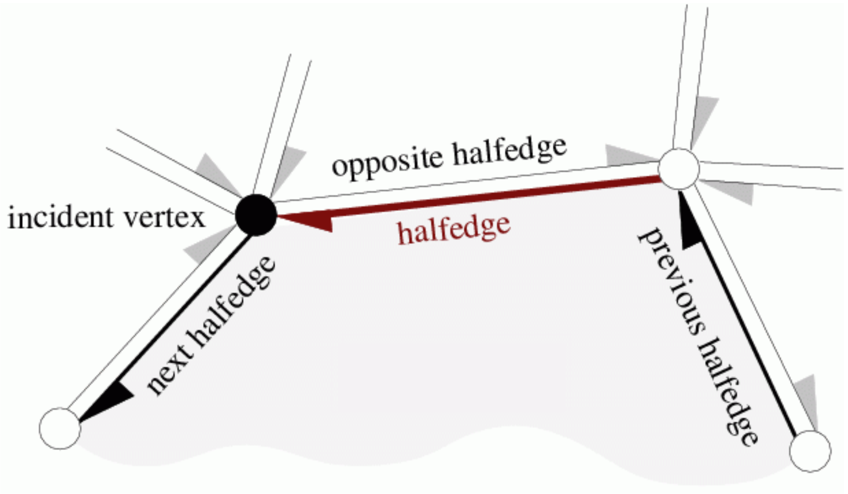
CGAL manual

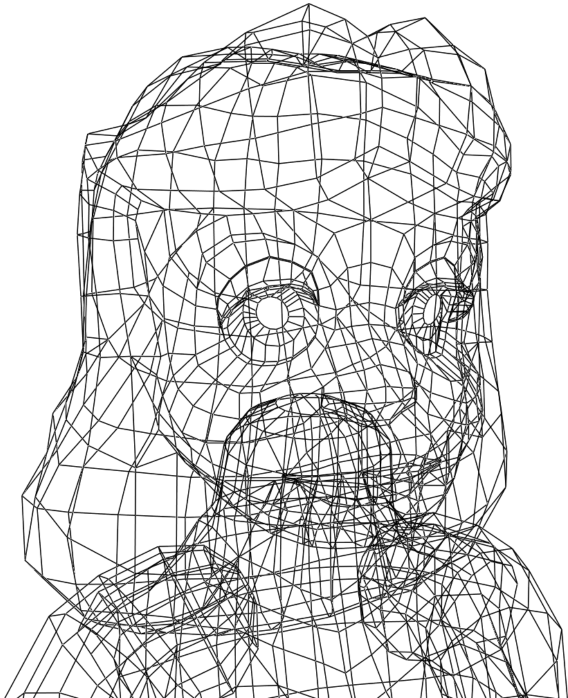
Flexible Data Structure

Vertex	
Halfedge_handle	halfedge()
Point&	point()
......	...

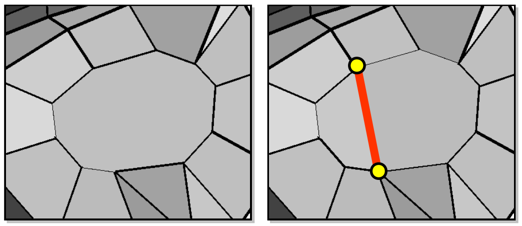
Halfedge	
Halfedge_handle	opposite()
Halfedge_handle	next()
Halfedge_handle	prev()
Vertex_handle	vertex()
Facet_handle	facet()
......	...

Facet	
Halfedge_handle	halfedge()
Plane&	plane()
Normal&	normal()
Color&	color()
......	...

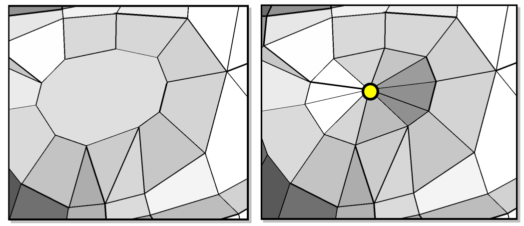




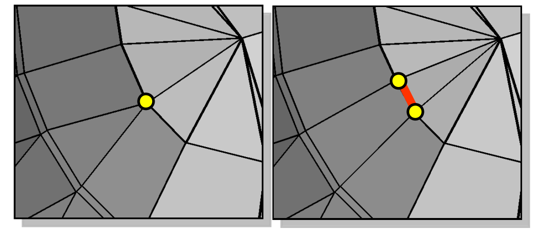
Euler Operators



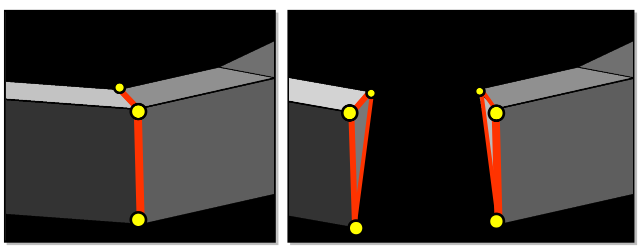
split_facet
join_facet



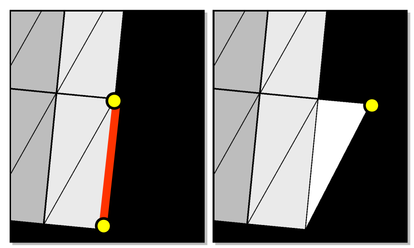
create_center_vertex
erase_center_vertex



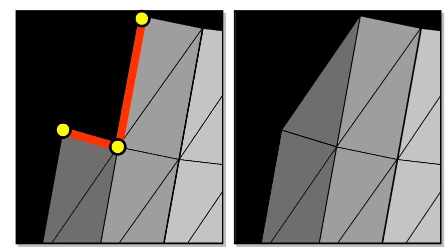
split_vertex
join_vertex
(aka edge collapse)



split_loop
join_loop



add_vertex_and_facet
_to_border
erase_facet



add_facet_to_border
erase_facet

CGAL manual

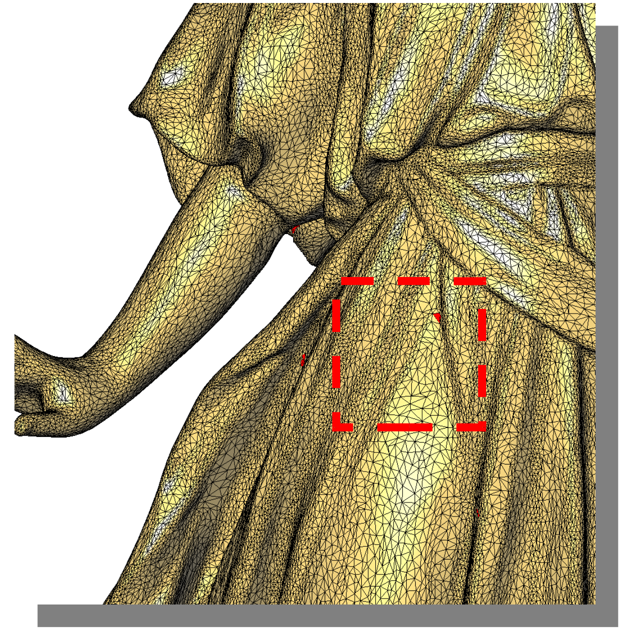
Algorithms

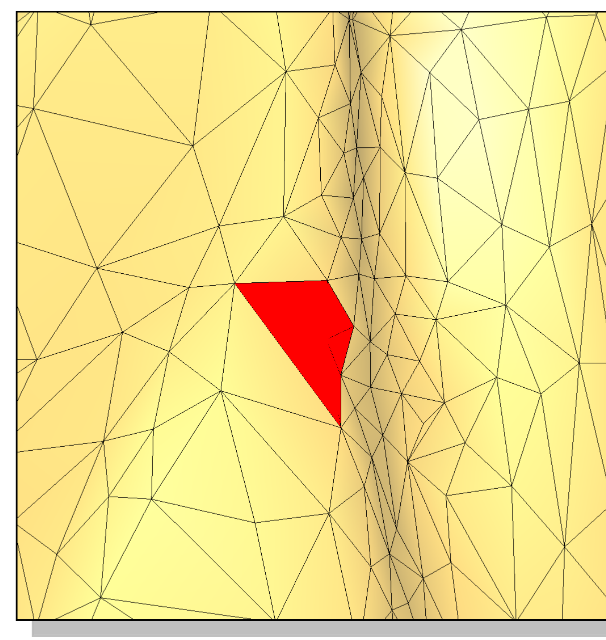
Algorithms

- Intersection detection
- AABB Tree
- Bounding volumes
- Boolean operations
- Kernel
- Parameterization
- Subdivision
- Principal component analysis
- Extraction of ridges
- Simplification

Intersection Detection

- Efficient algorithm for finding all intersecting pairs for large numbers of axis-aligned bounding boxes.
- Generic programming: Boxes can contain objects of any type

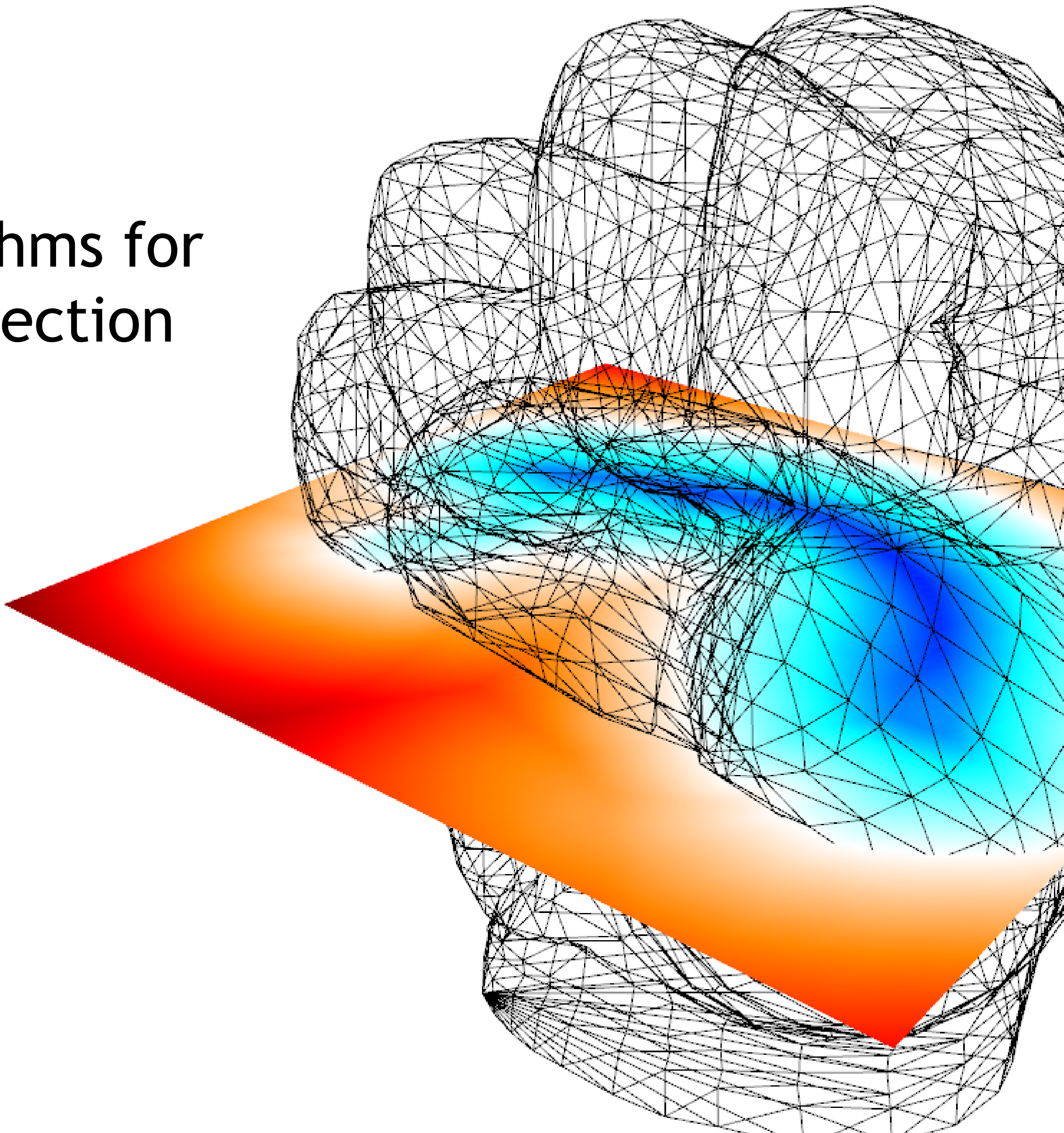




CGAL manual

AABB Tree

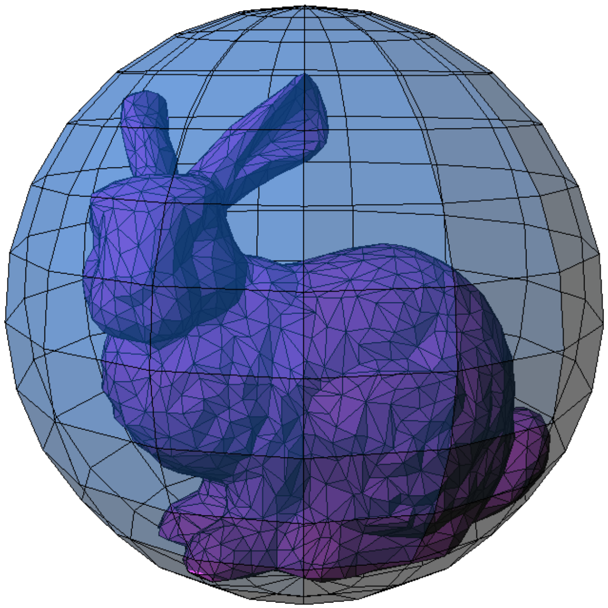
Efficient algorithms for intersection detection and distance computation

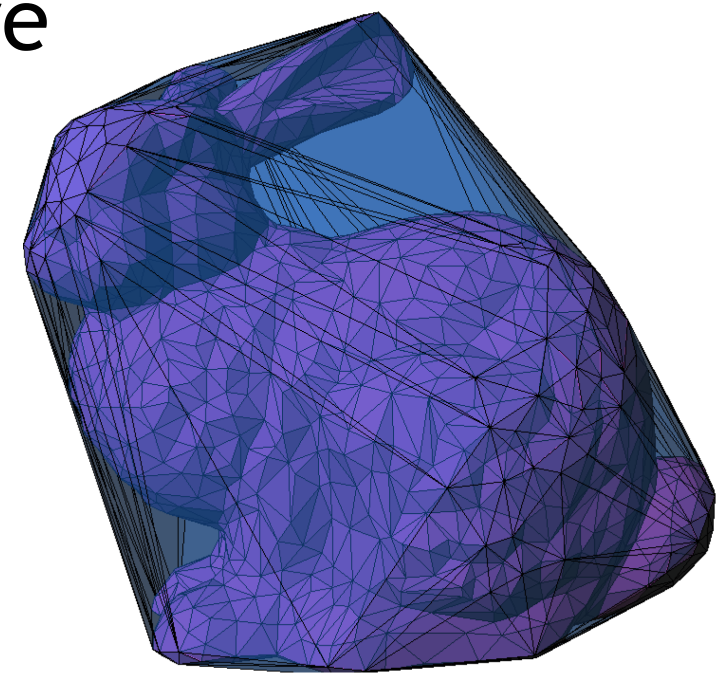


Bounding Volumes

- Convex hull
- Bounding sphere

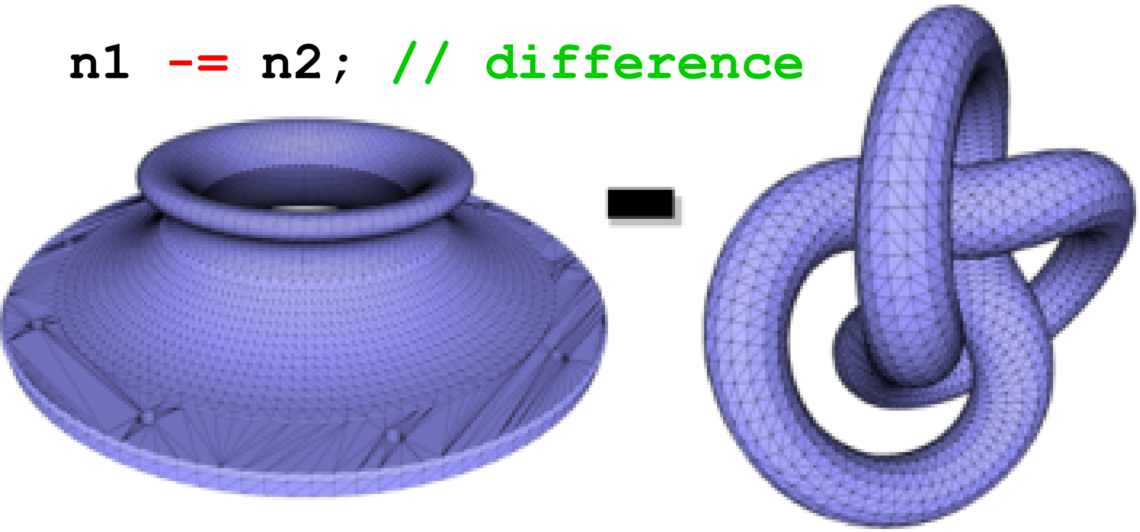
- Bounding sphere of spheres

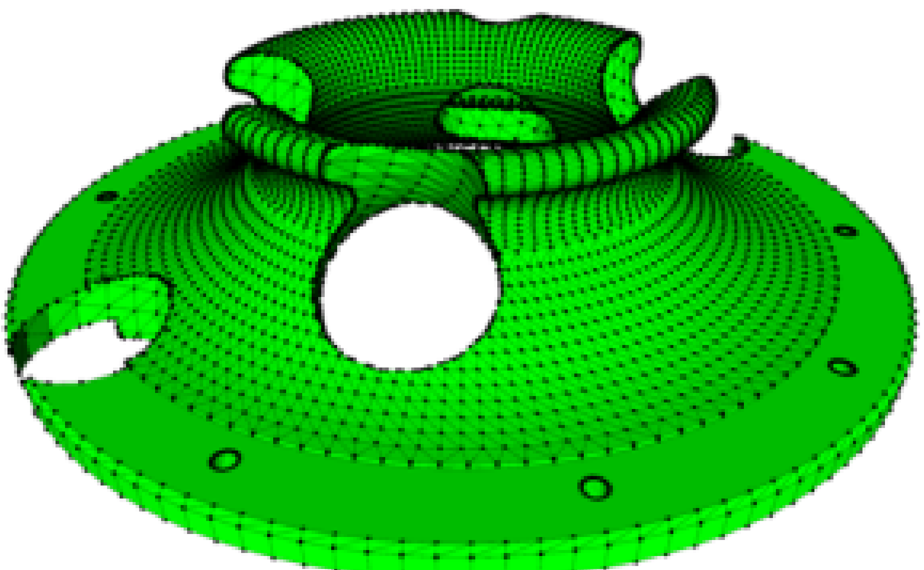


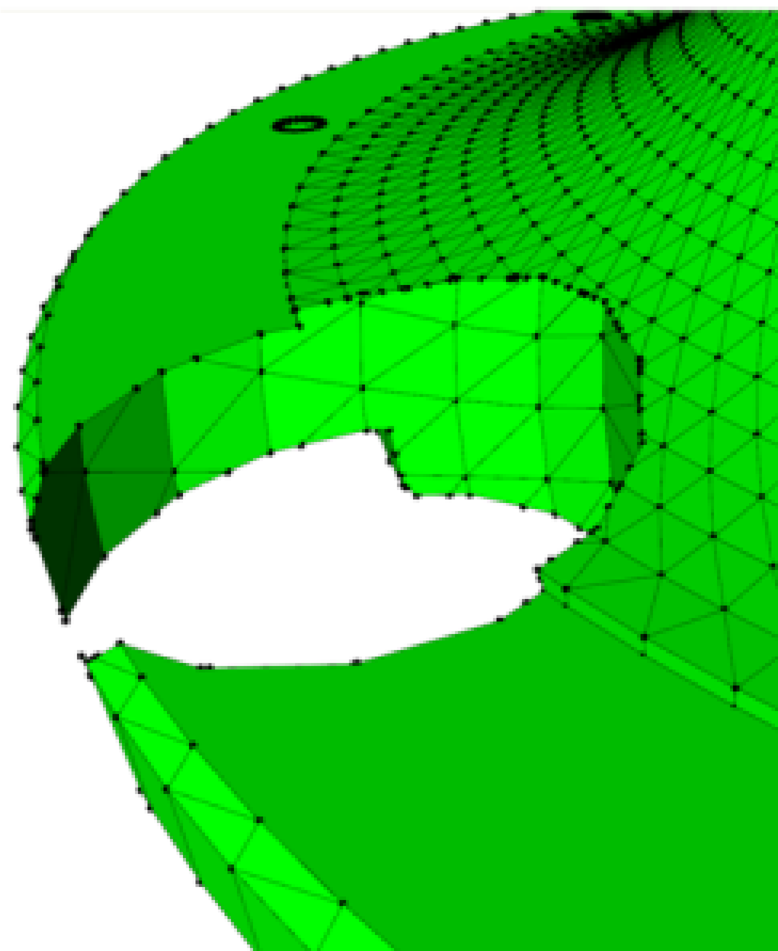


3D Boolean Operations

```
n1 -= n2; // difference
```

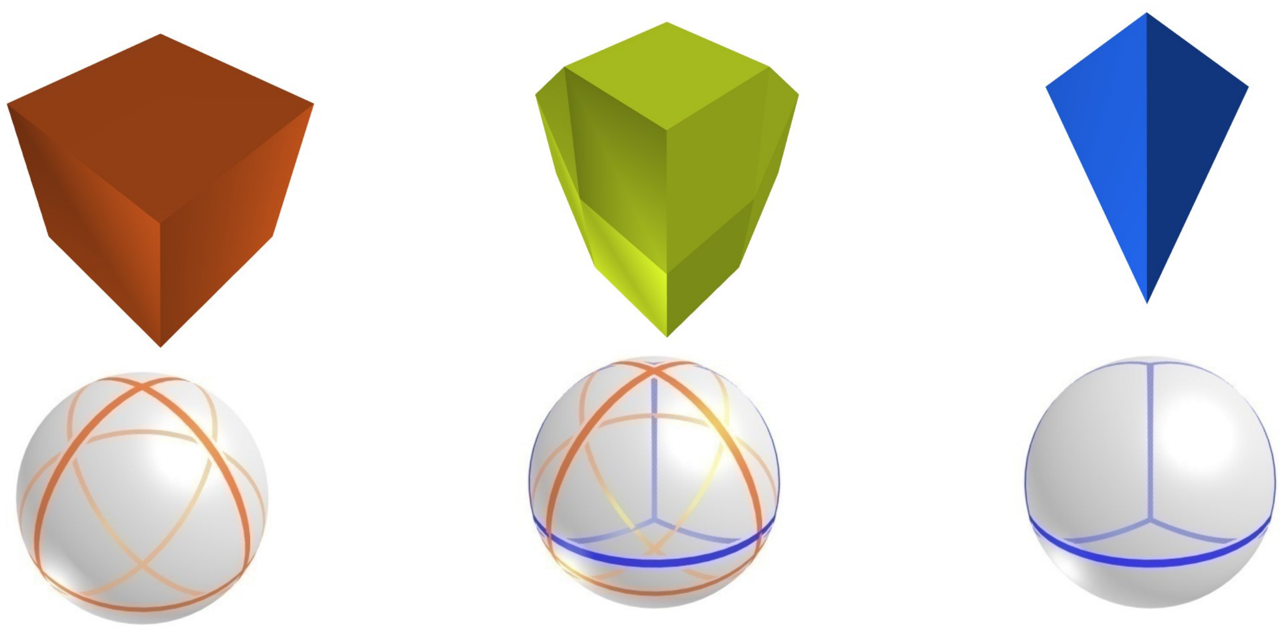






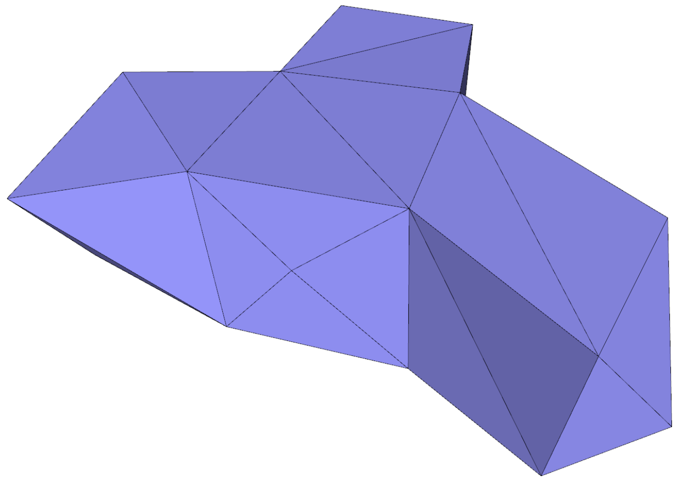
Minkowski-Sums of Polytopes

- The Gaussian map of a polytope is the decomposition of S2 into maximal connected regions so that the extremal point is the same for all directions within one region
- The overlay of the Gaussian maps of two polytopes P and Q is the Gaussian map of the Minkowski sum of P and Q

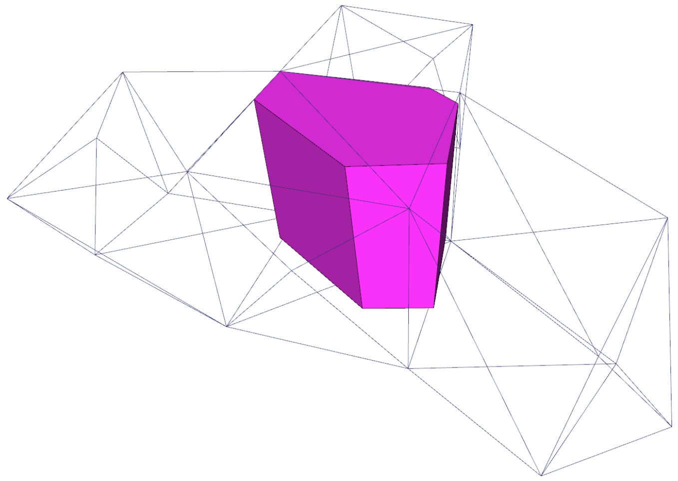


Kernel of a Polyhedron

- Intersection of all its interior half-spaces
- Uses linear programming: CGAL::QP_solver



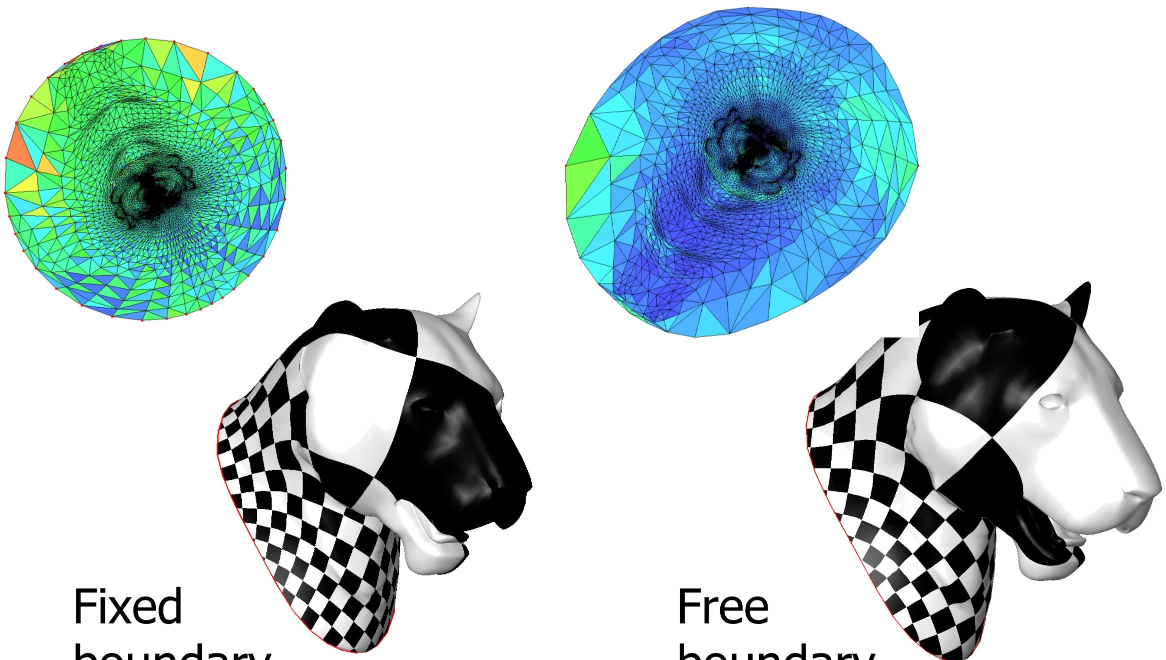
input polyhedron



kernel

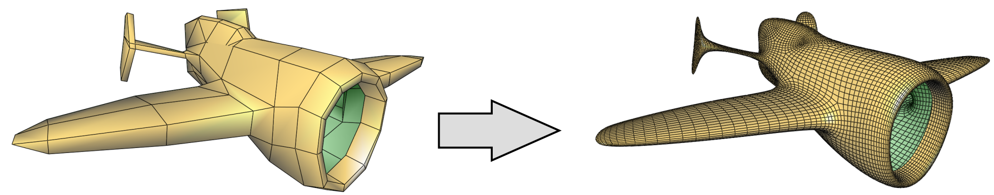
Parameterization

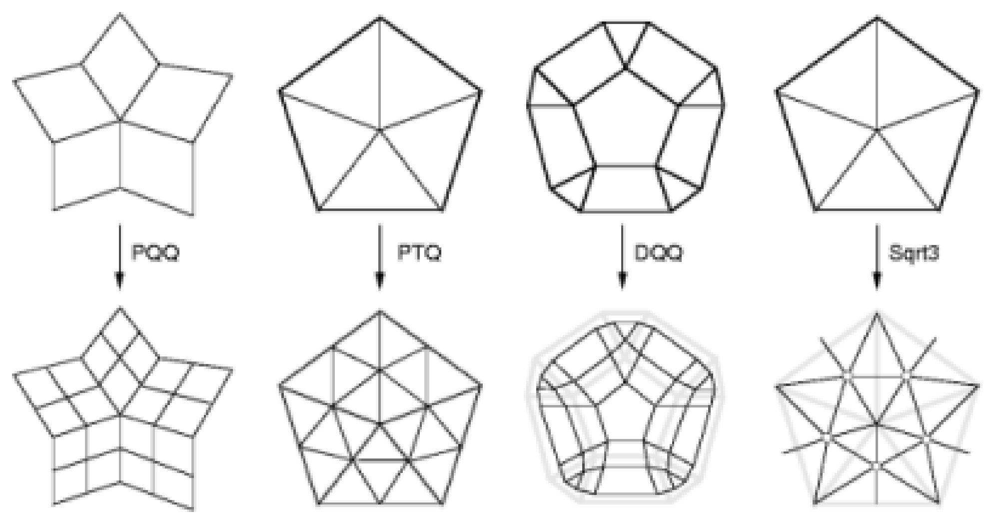
- Planar
- Conformal [Eck et al., Levy et al., Desbrun et al.]
- Mean value coordinates [Floater]
- ...



Subdivision

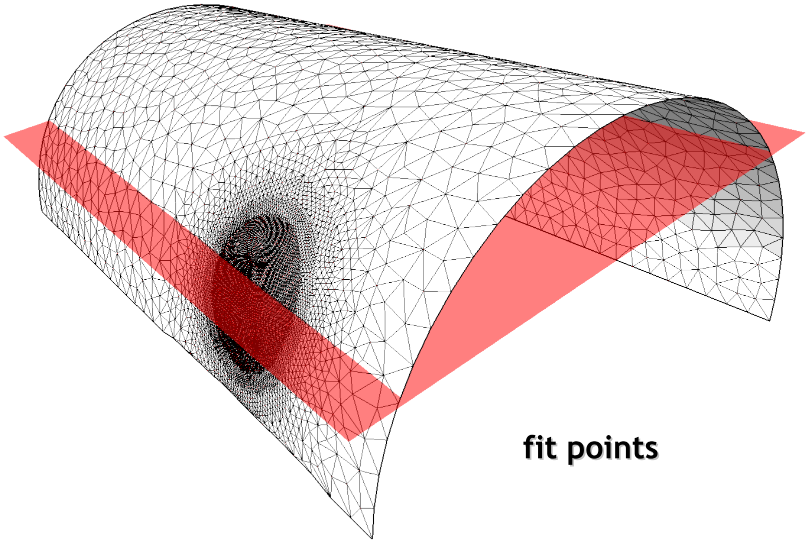
- Designed to work on CGAL polyhedron
- Catmull-Clark
- Loop
- Doo-Sabin
- Sqrt3
- …



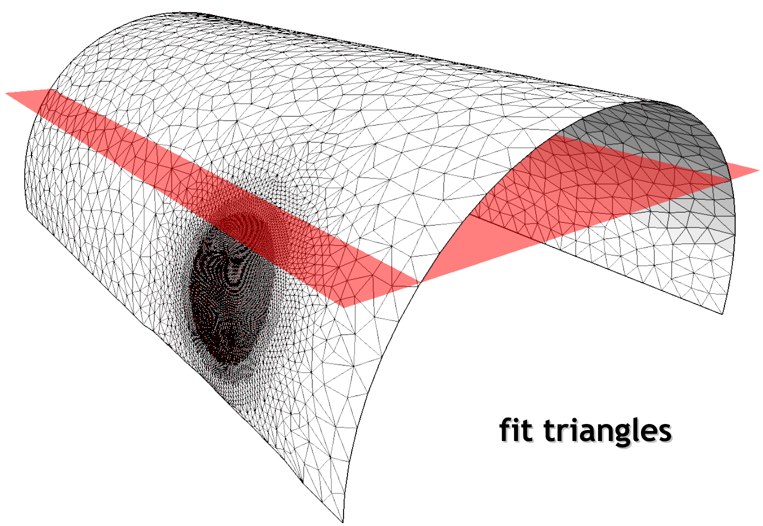


Principal Component Analysis

- Linear least squares fitting on sets of 3D points or triangles



fit points



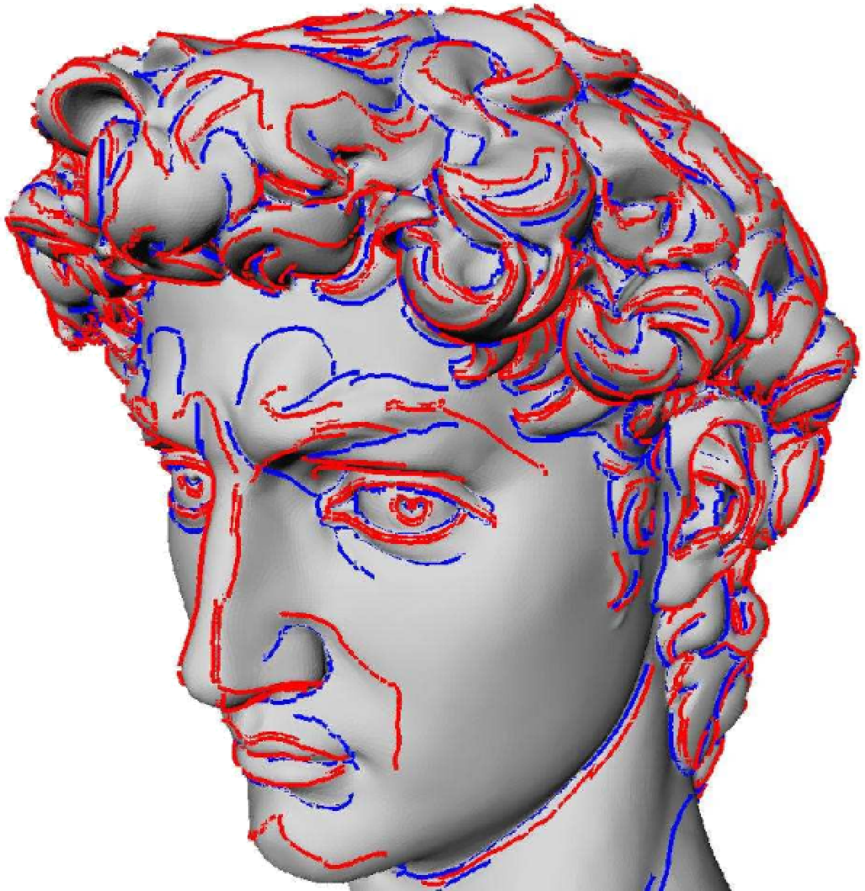
fit triangles

CGAL manual

Extraction of Ridges

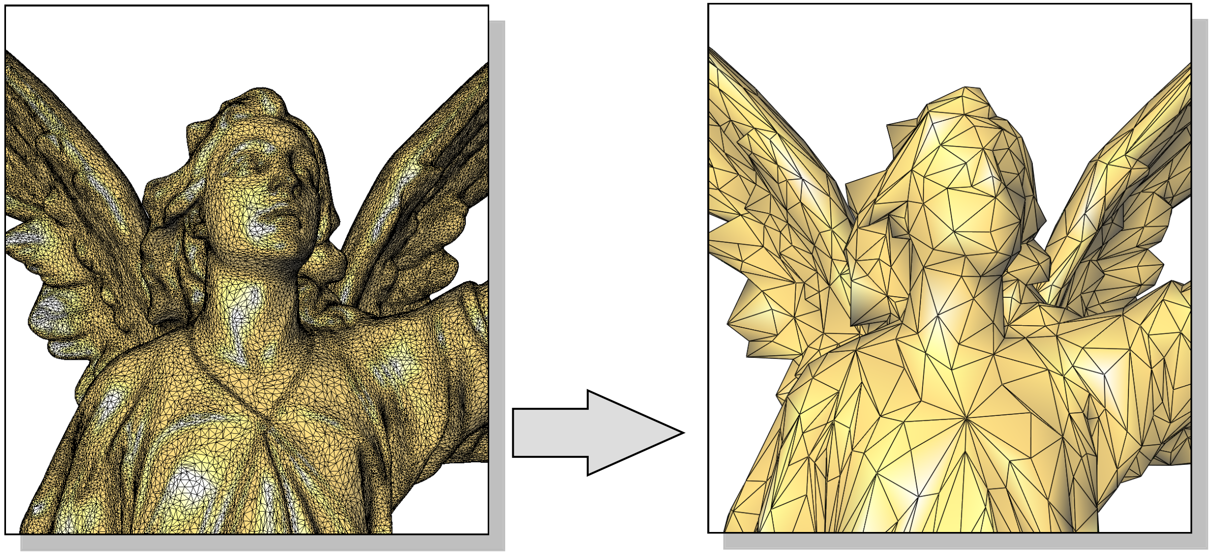
- Ridge: curve along which one of the principal curvatures has an extremum along its curvature line.

CGAL manual



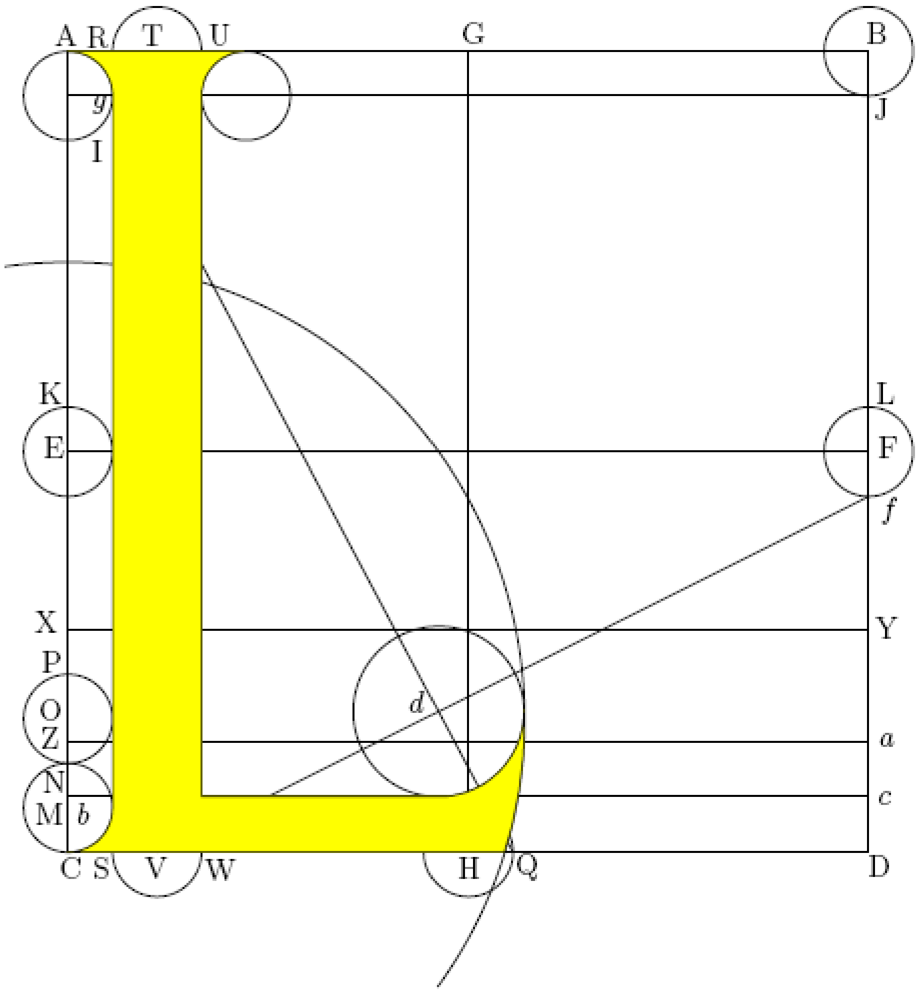
Simplification

- Implementation of [Lindstrom-Turk] volume-preserving method.



Summary: CGAL for Modeling and Polyhedral Surfaces

- The halfedge data structure and the polyhedron are highly flexible
- CGAL provides many algorithms for geometric modeling and geometry processing
- Polyhedral surface as output of surface mesh generation algorithms



CGAL for Mesh Generation

Pierre Alliez

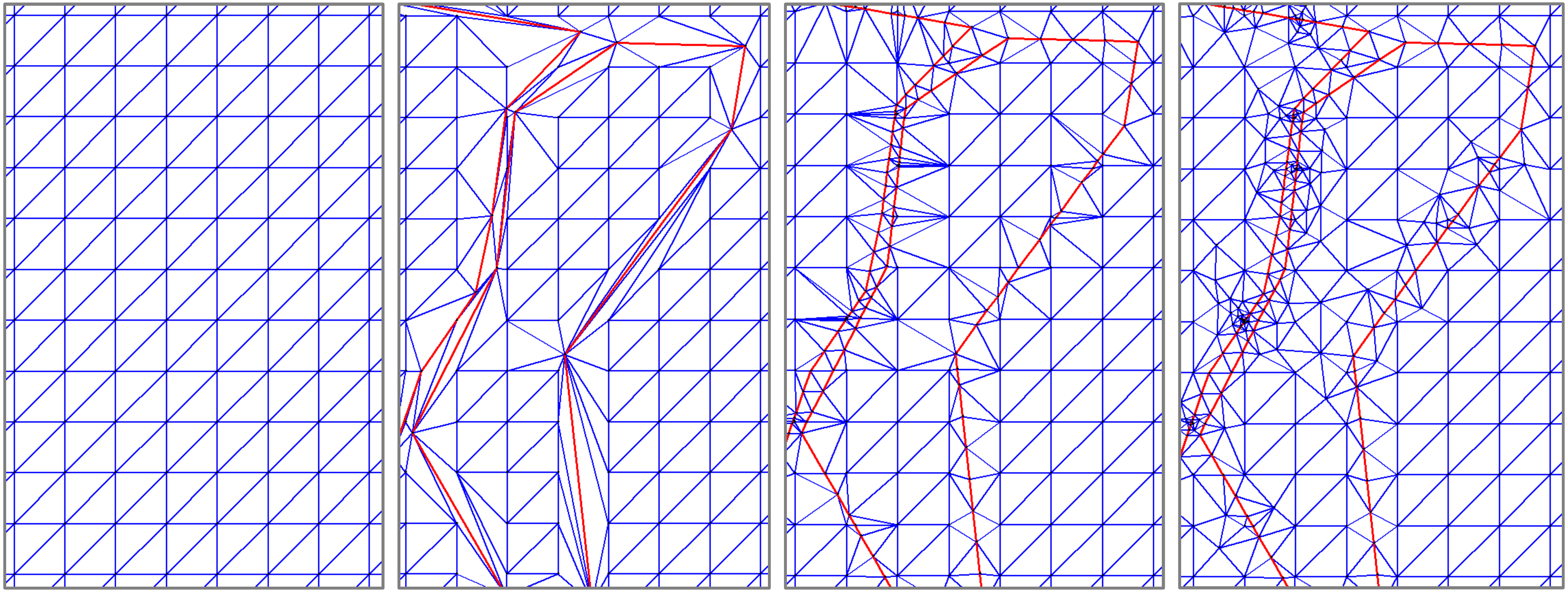
INRIA

Outline

- 2D mesh generation
- Surface mesh generation
- 3D mesh generation
- Work in progress

2D Mesh Generation

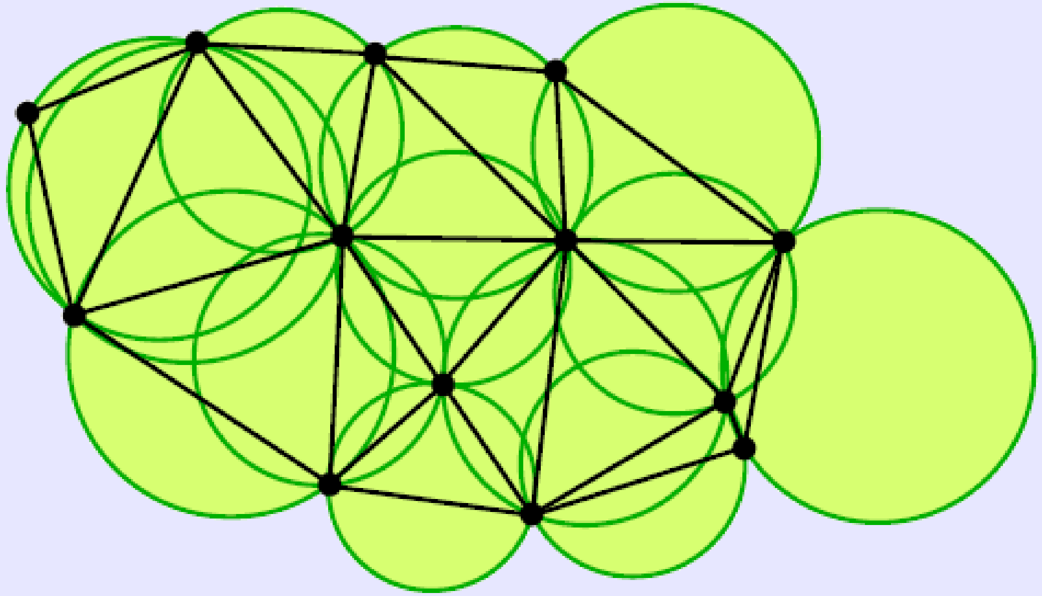
2D Mesh Generation



From Triangulations to Quality Meshes

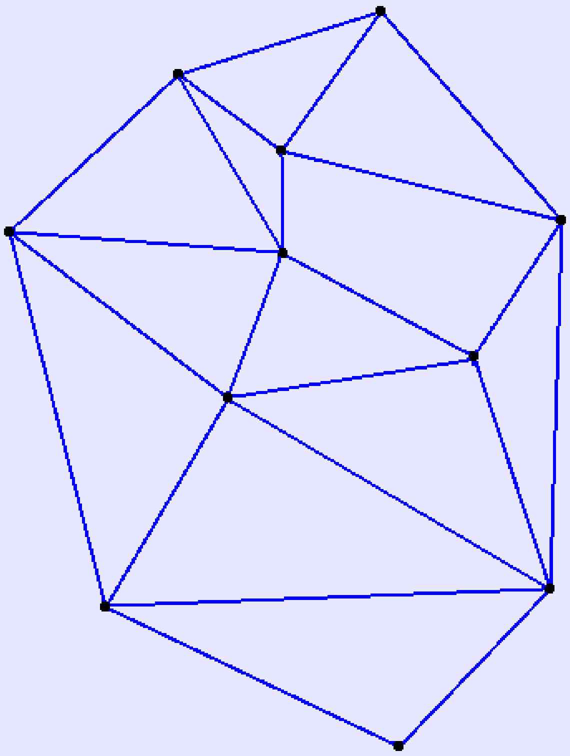
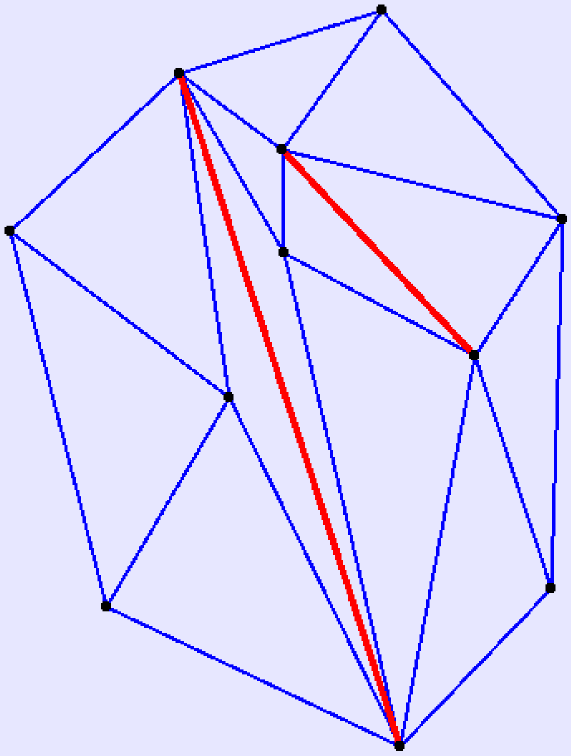
Delaunay Triangulation

- A triangulation is a Delaunay triangulation, if the circumscribing circle of any facet of the triangulation contains no vertex in its interior

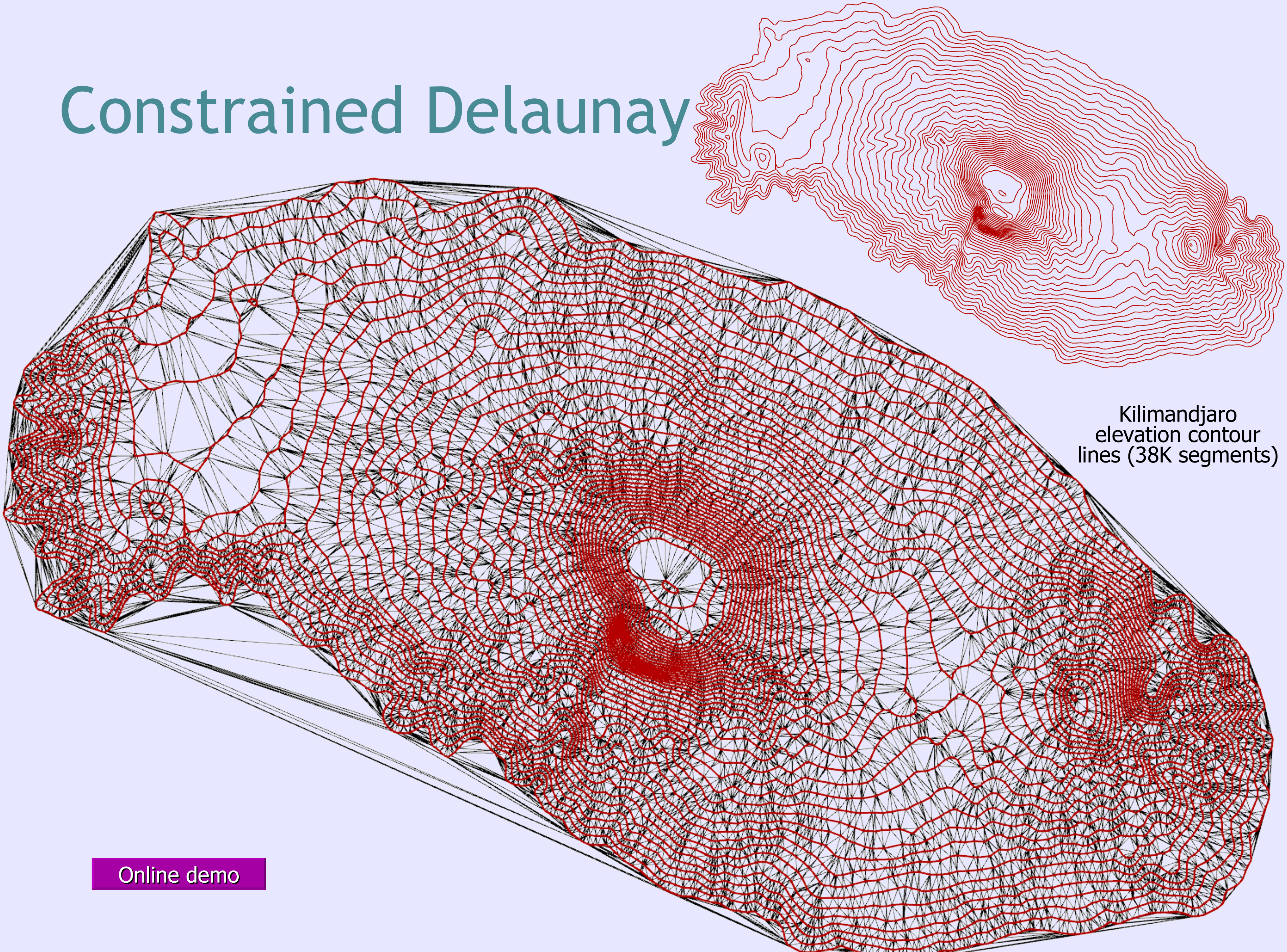


Demo

Constrained Delaunay Triangulation

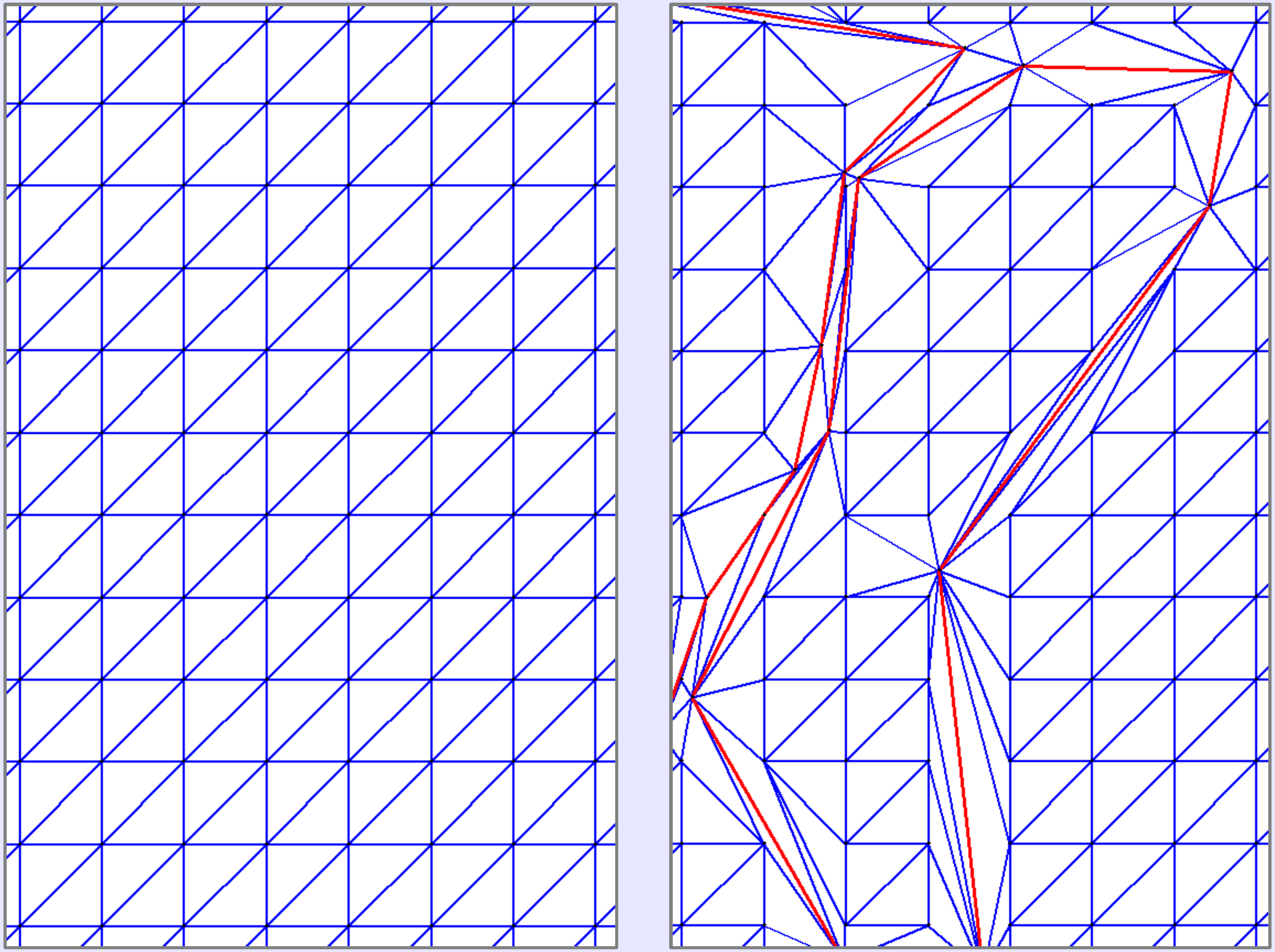
Constrained Delaunay



Kilimandjaro
elevation contour
lines (38K segments)

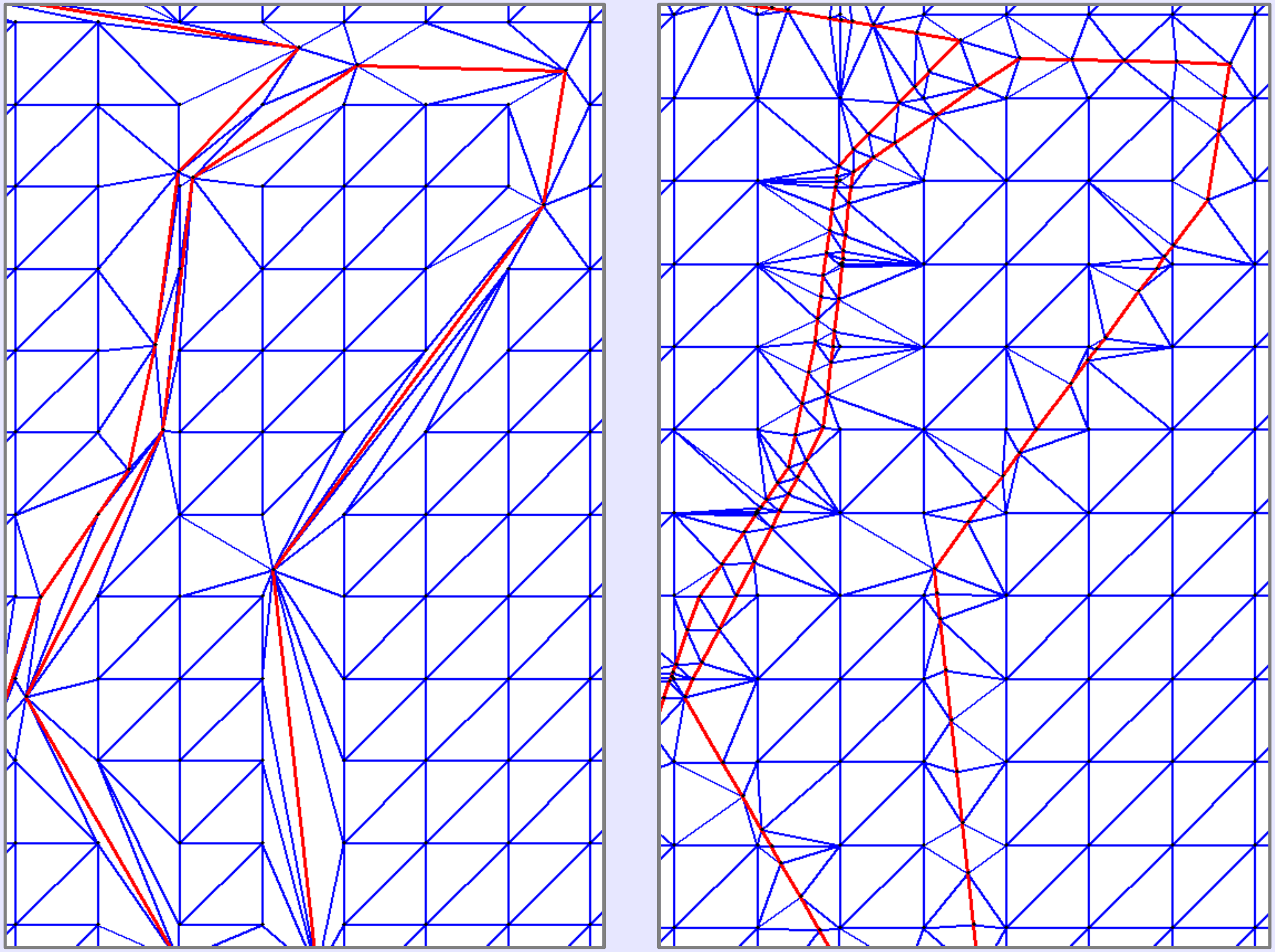
Online demo

Adding Constraints



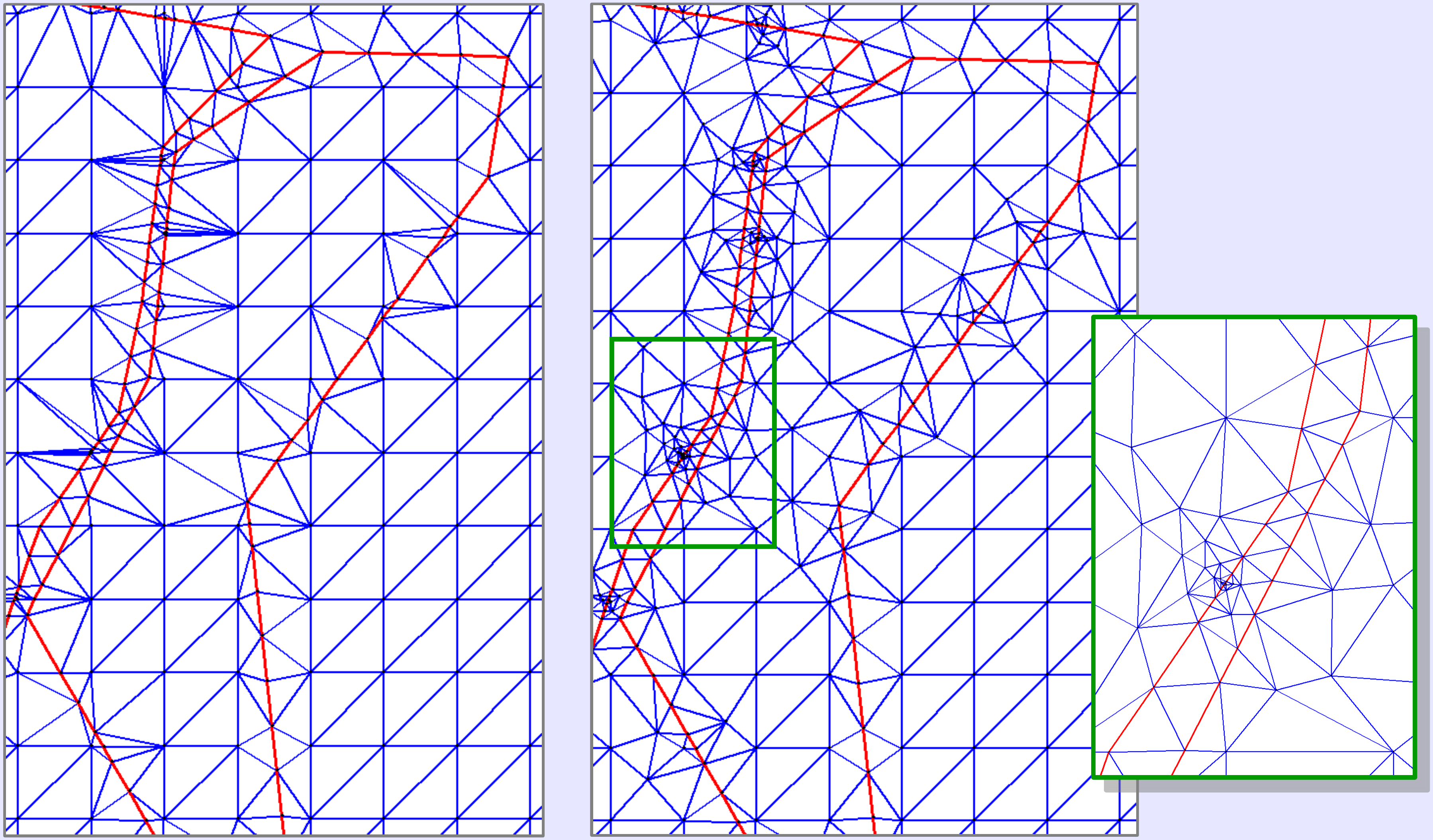
CGAL manual

Conforming a Triangulation



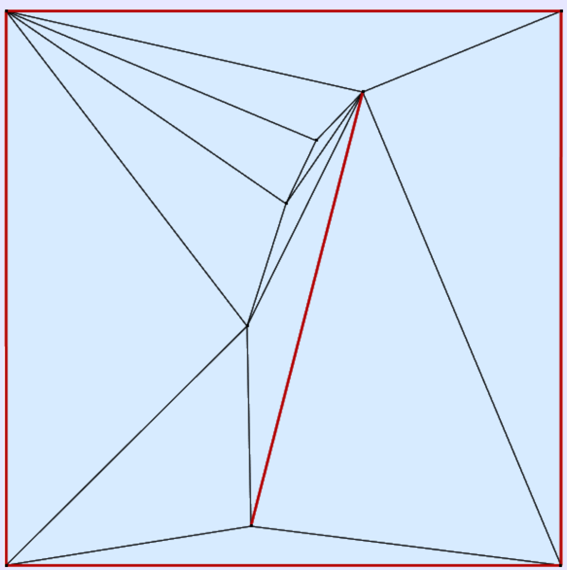
CGAL manual

Delaunay Refinement

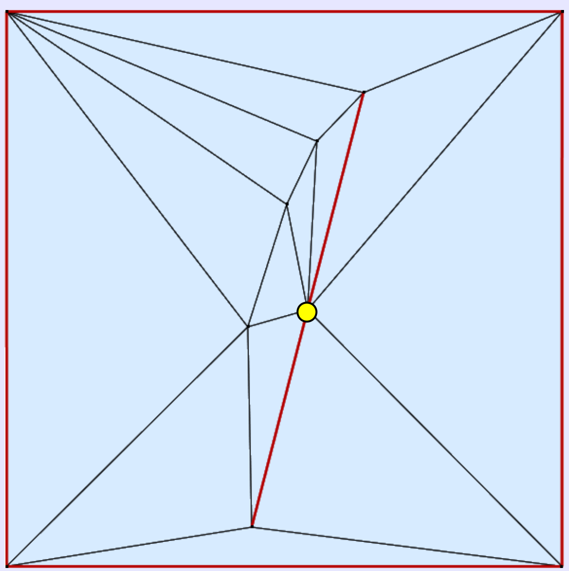


Conforming a Triangulation

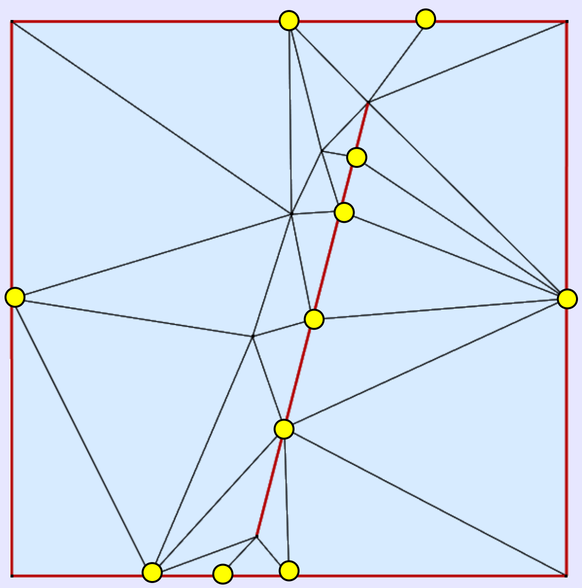
Any constrained Delaunay triangulation can be refined into a conforming Delaunay or Gabriel triangulation by adding Steiner vertices.



non conforming



Delaunay conforming

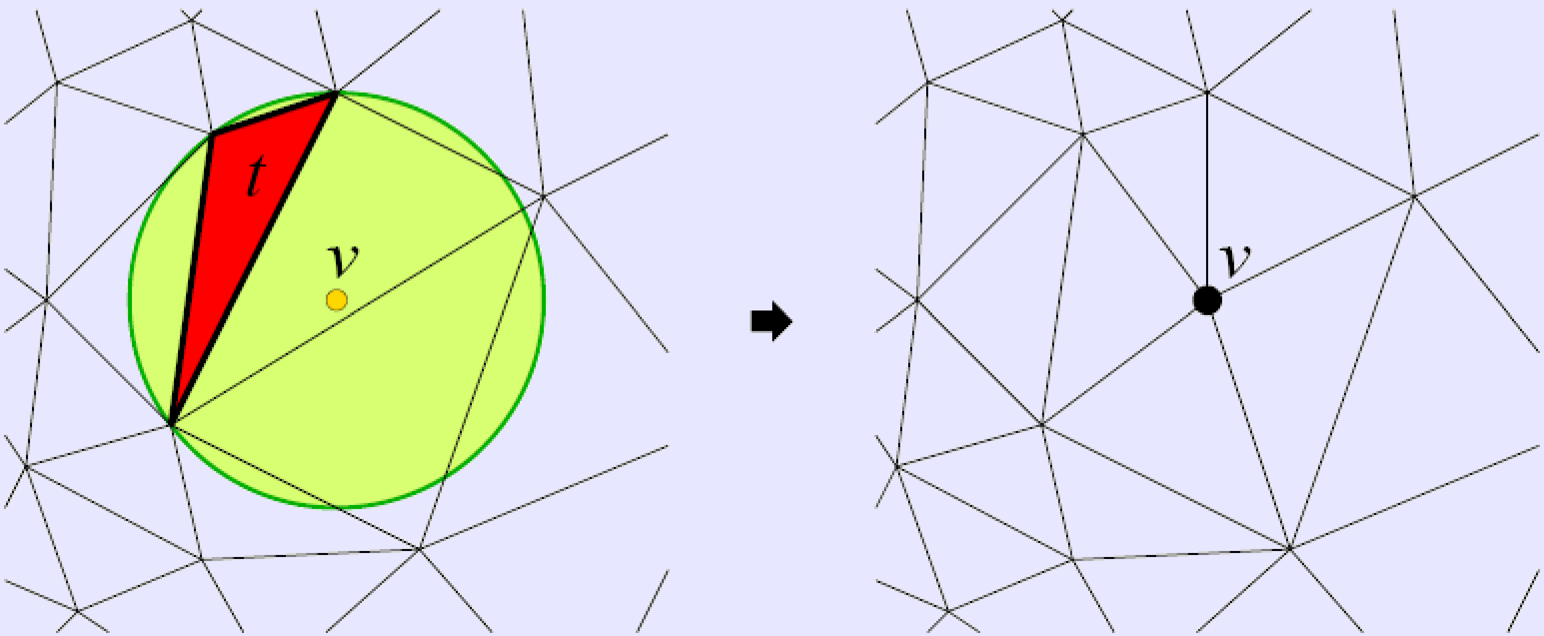


Gabriel conforming

Delaunay Refinement Rules

Rule #1: break bad elements by inserting circumcenters (Voronoi vertices)

- "bad" in terms of size or shape (too big or skinny)

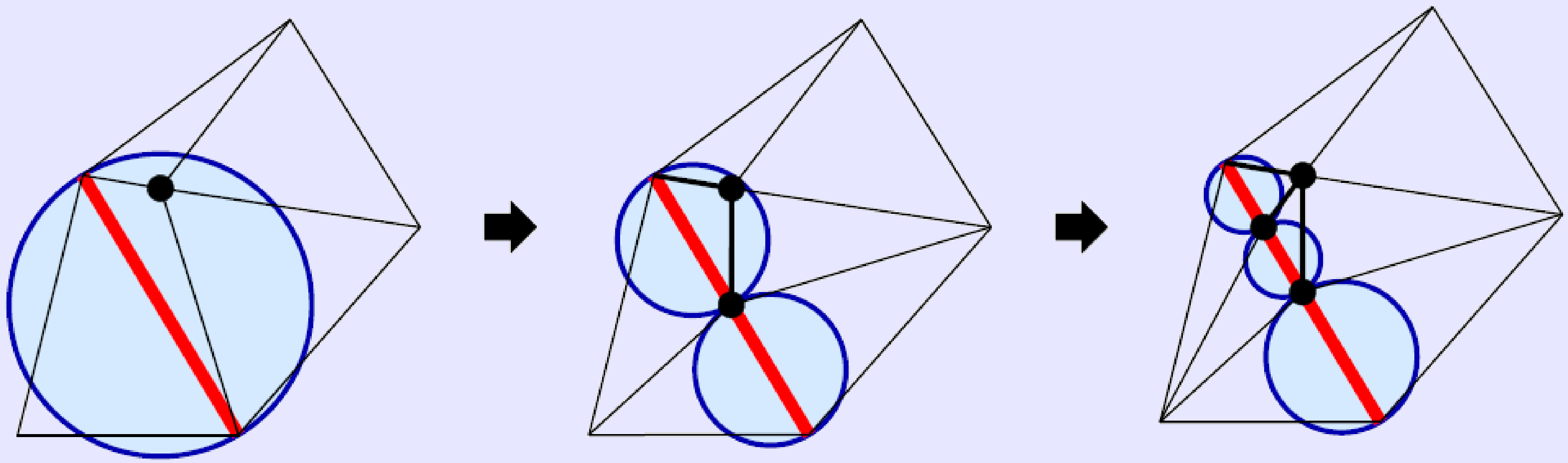


Picture taken from [Shewchuk]

Delaunay Refinement Rules

Rule #2: Midpoint vertex insertion

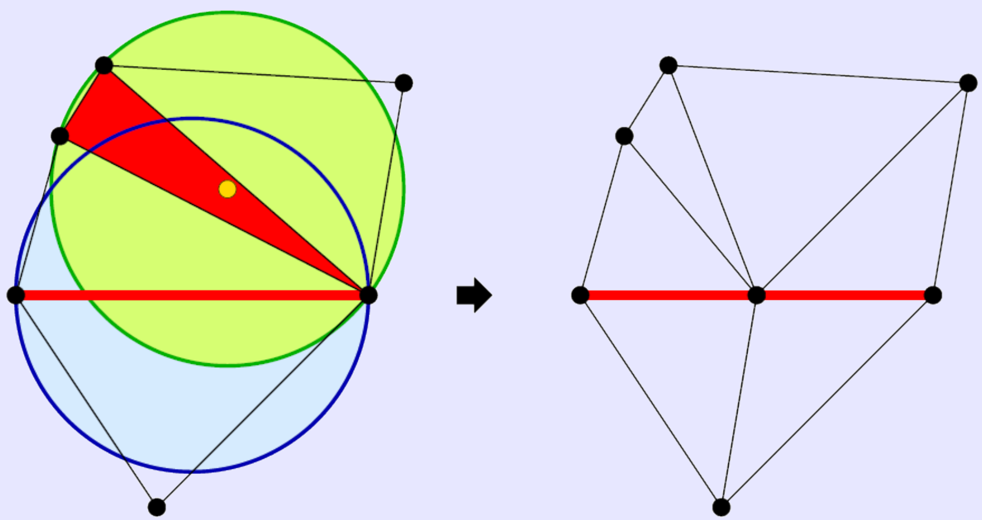
A constrained segment is said to be encroached, if there is a vertex inside its diametral circle



Picture taken from [Shewchuk]

Delaunay Refinement Rules

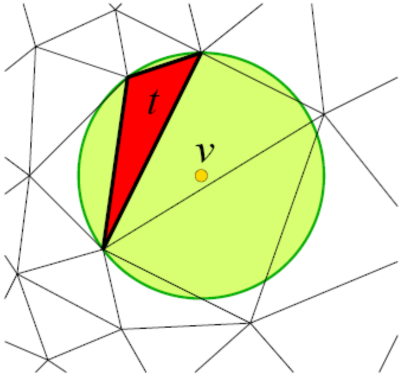
Encroached subsegments have priority over skinny triangles

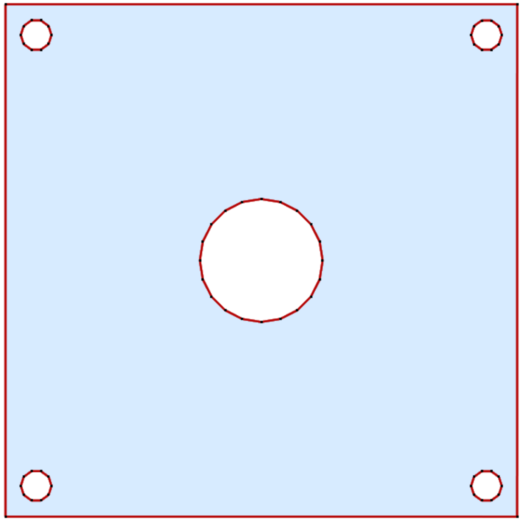


Picture taken from [Shewchuk]

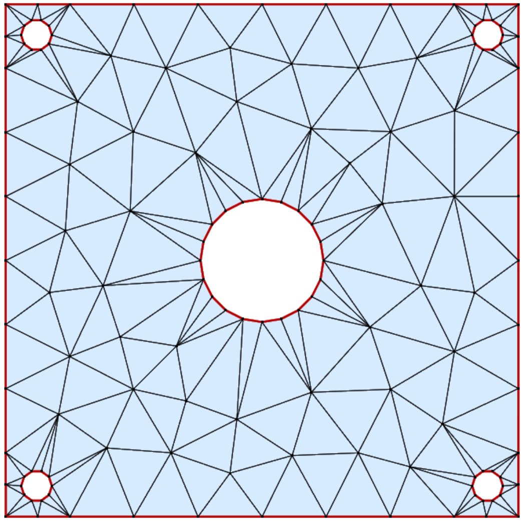
Parameters for Delaunay Refinement

- **Shape**
  - Lower bound on triangle angles

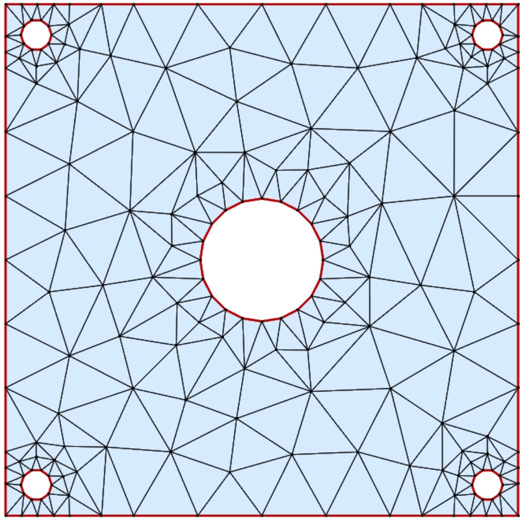




Input PLSG



5 deg

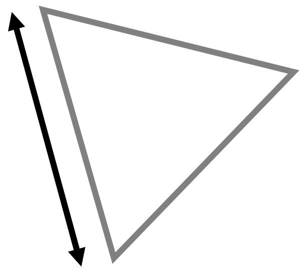


20.7 deg

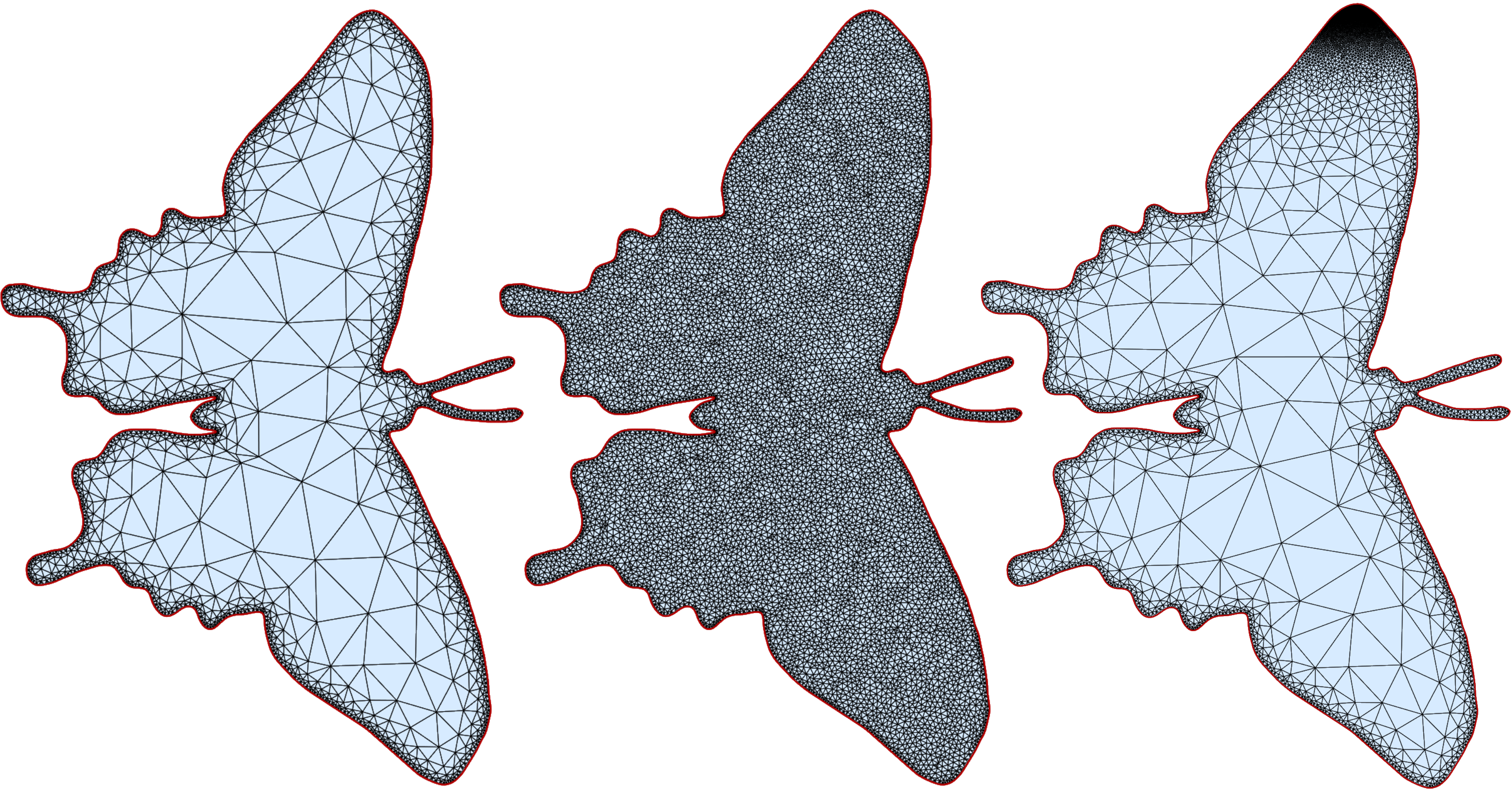
Online manual

Parameters for Delaunay Refinement

- Shape
  - Lower bound on triangle angles
- **Size**
  - No constraint
  - Uniform sizing
  - Sizing function



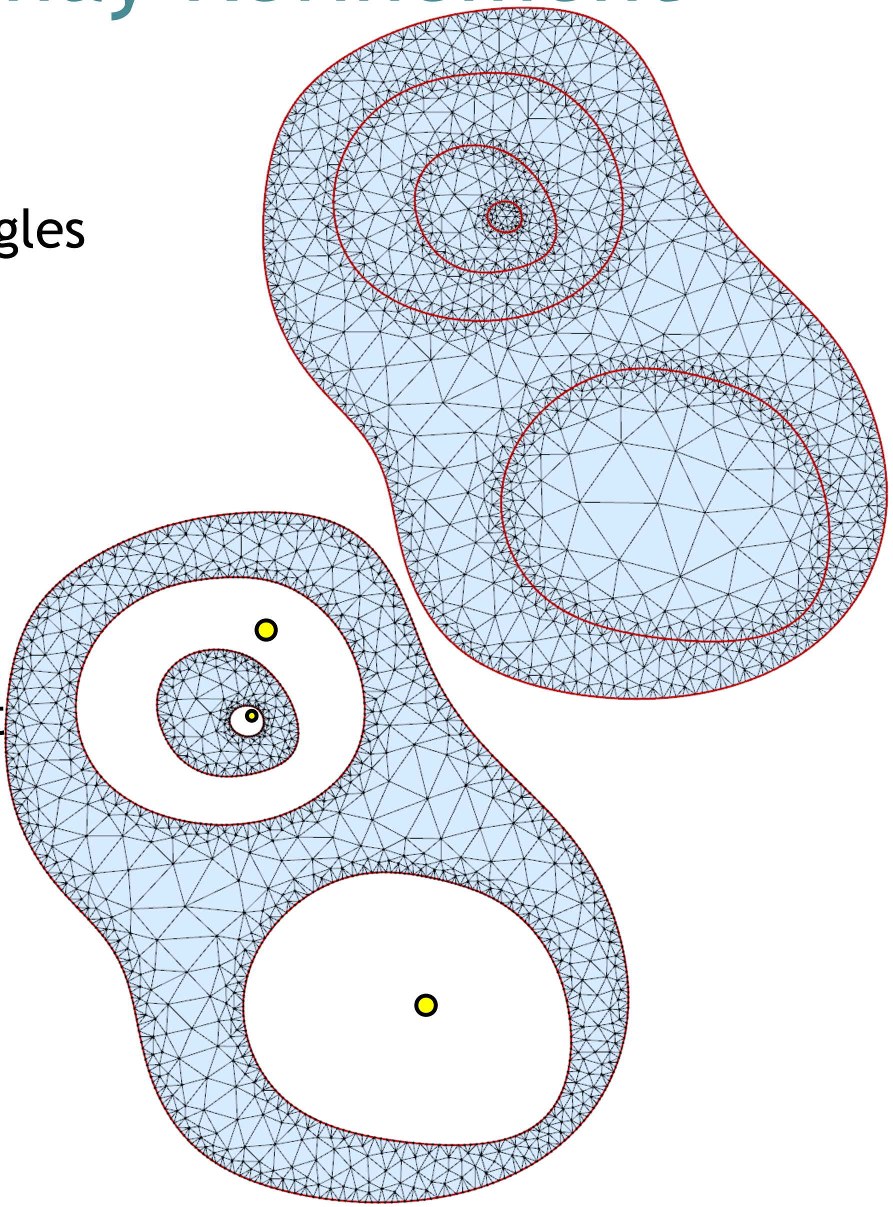
Sizing Parameter



No constraint   Uniform   Sizing function

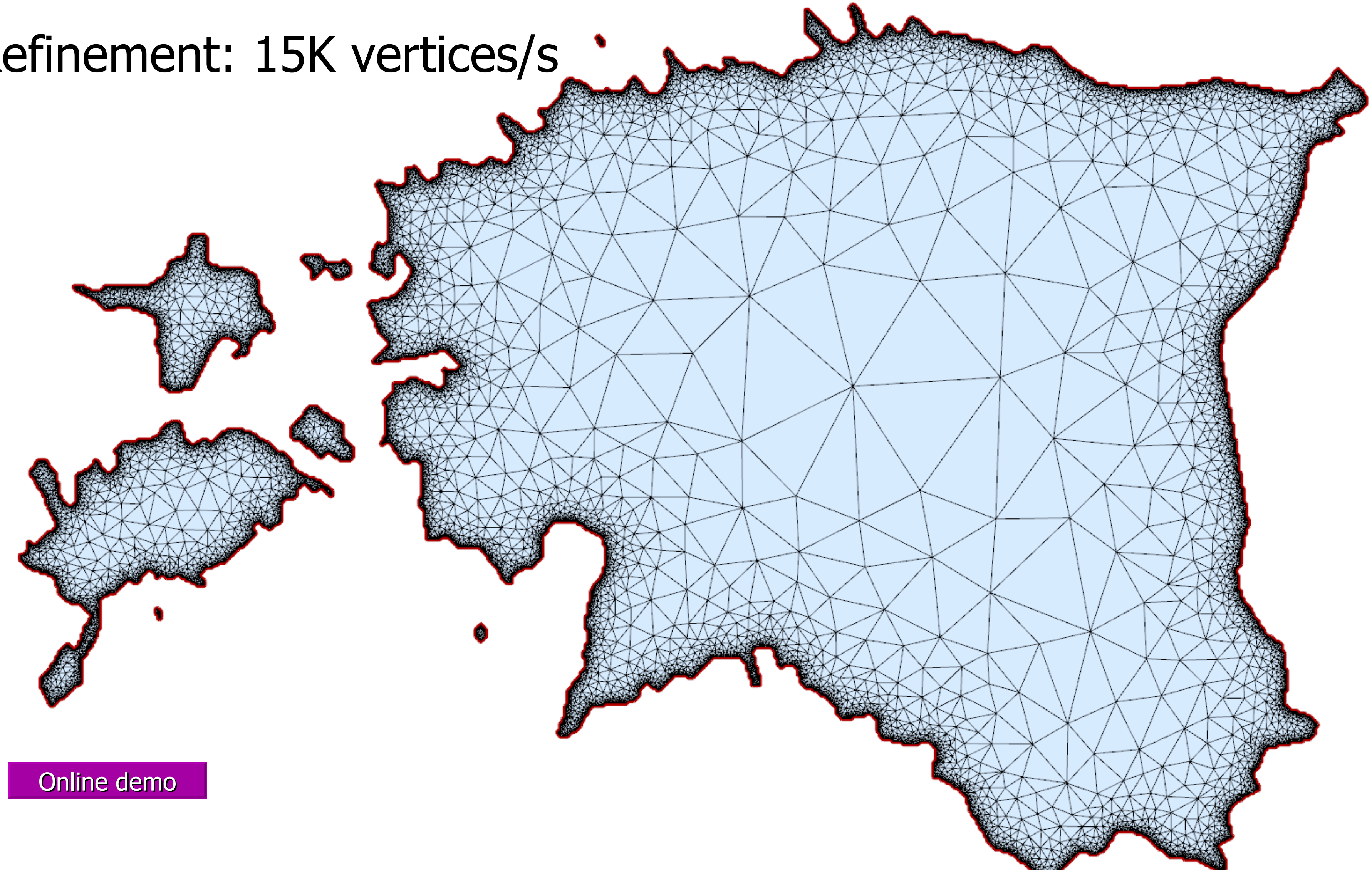
Parameters for Delaunay Refinement

- Shape
  - Lower bound on triangle angles
- Size
  - No constraint
  - Uniform sizing
  - Sizing function
- **Seeds**
  - Exclude/include component



Performances

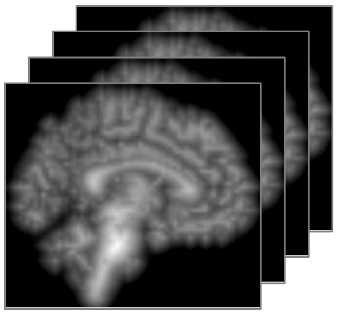
Refinement: 15K vertices/s



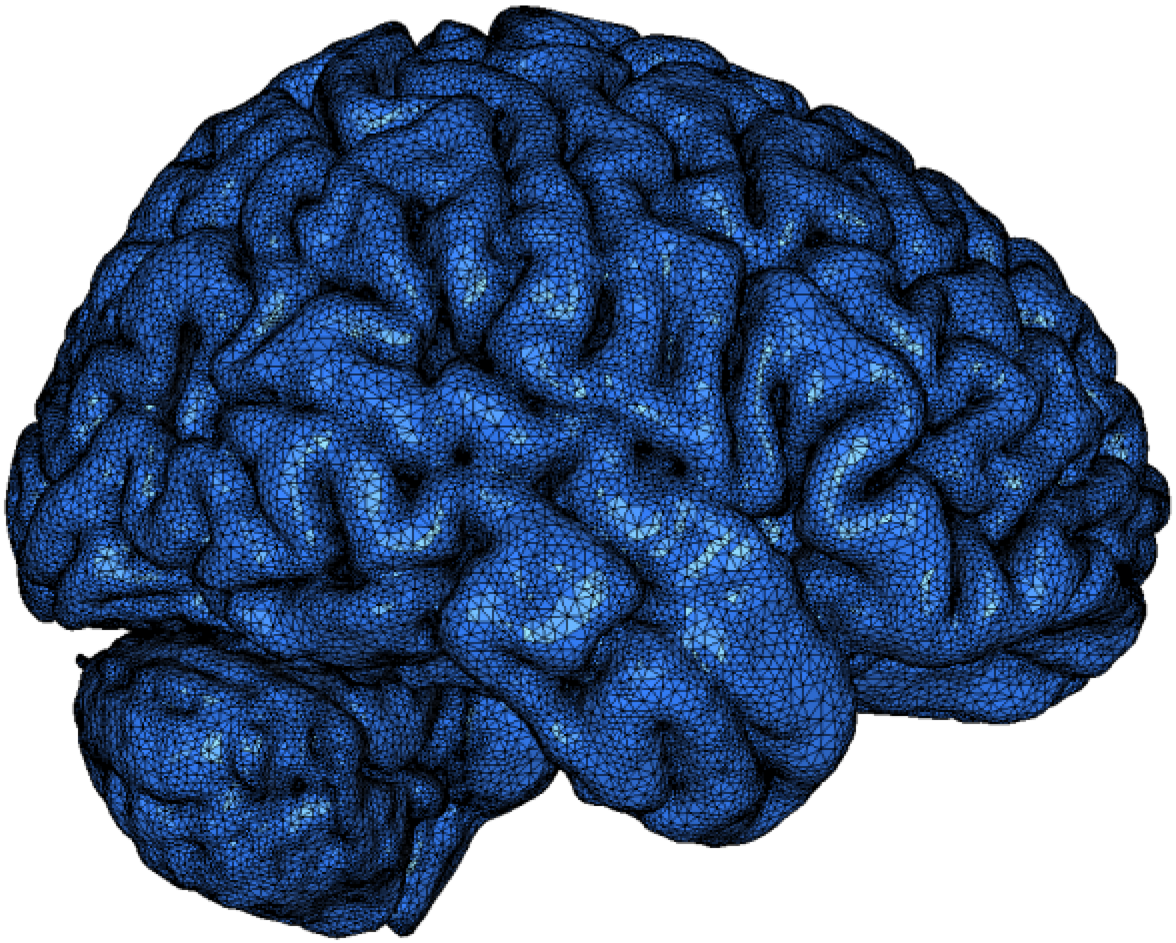
Online demo

Surface Mesh Generation

Surface Mesh Generation

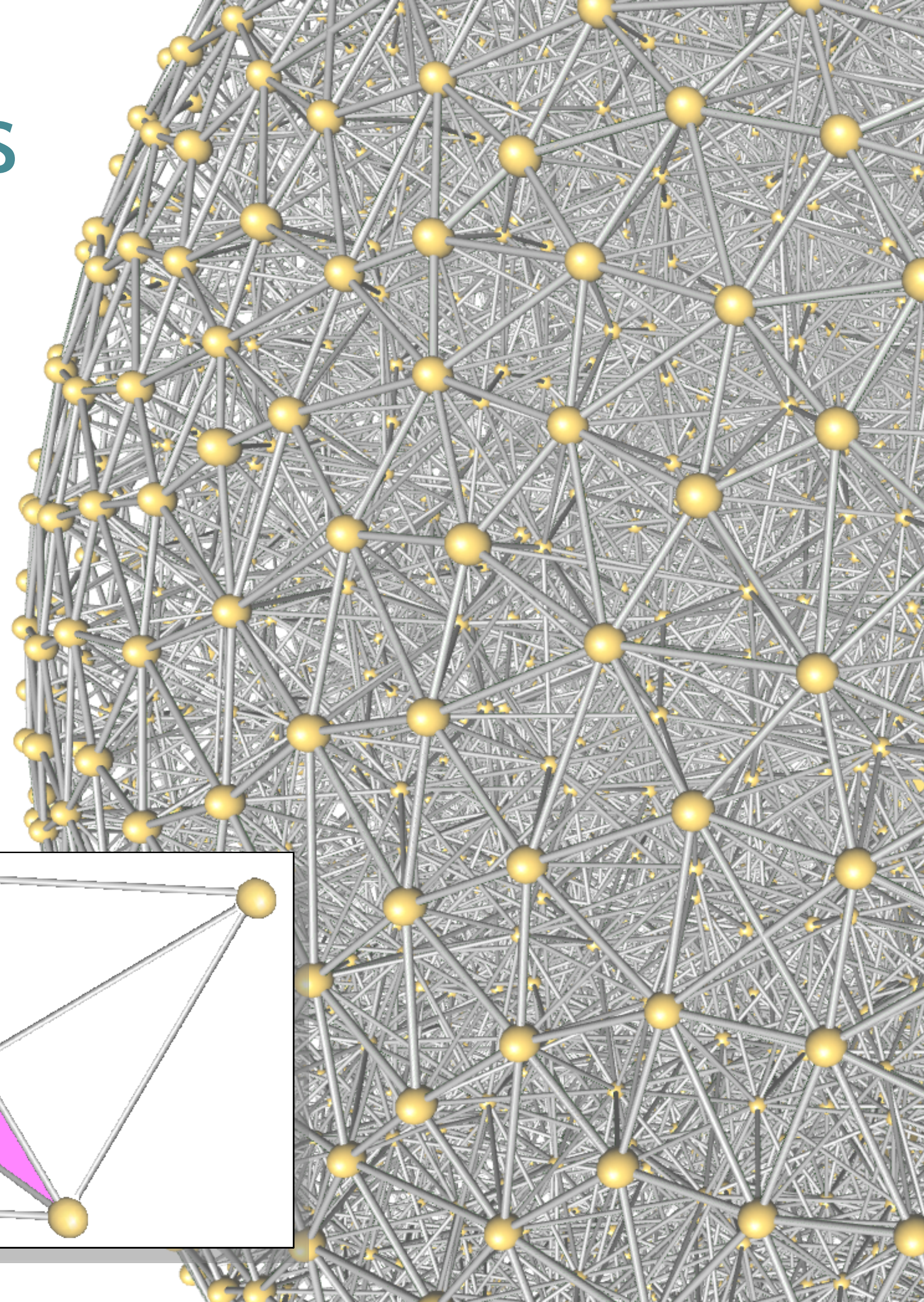


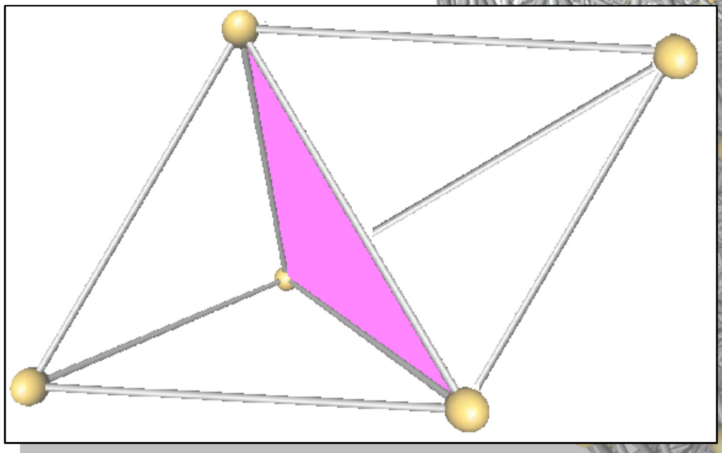
input



3D Triangulations

- Delaunay
- Regular
- Rich API
- Fully dynamic
- 1M points in 16s



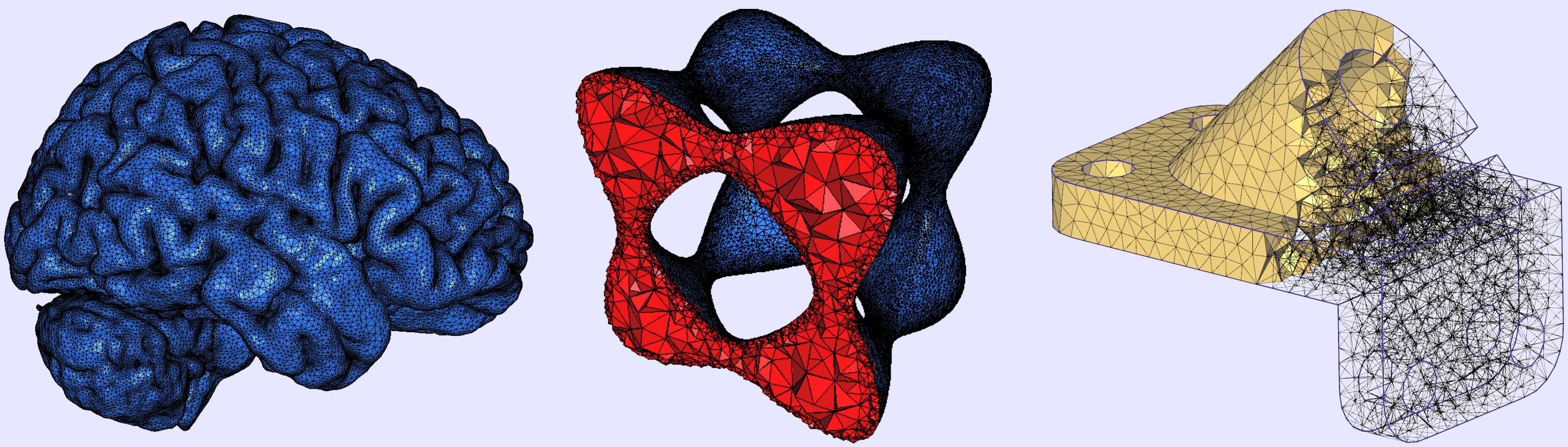


Online manual

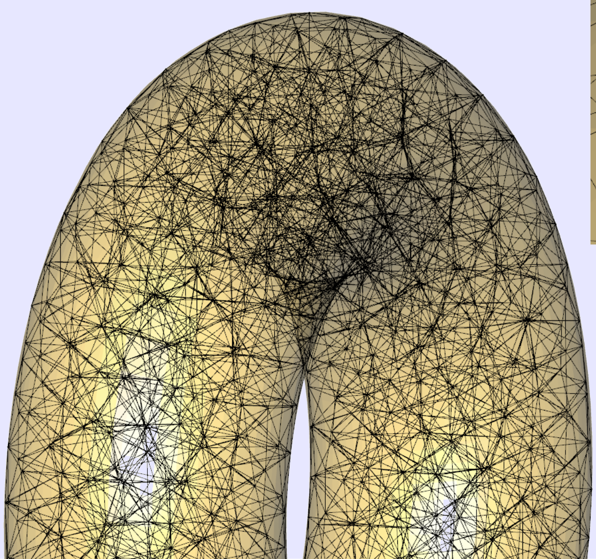
Mesh Generation

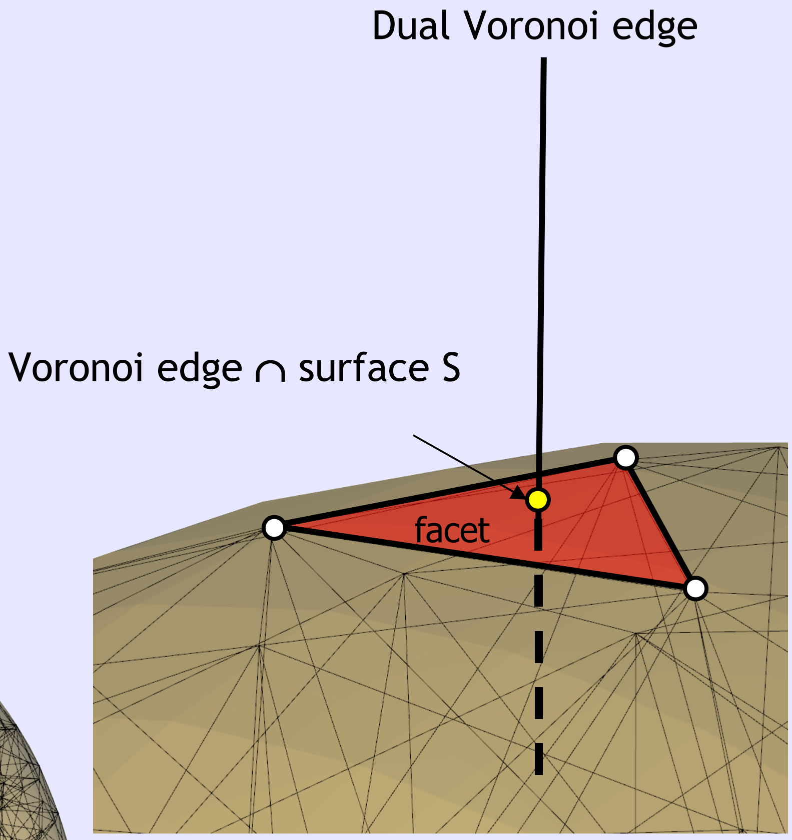
Key concepts:

- Delaunay filtering
- Delaunay refinement

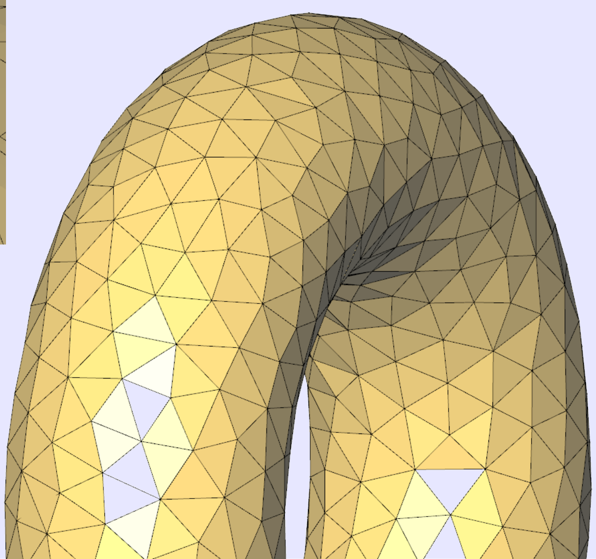


Delaunay Filtering





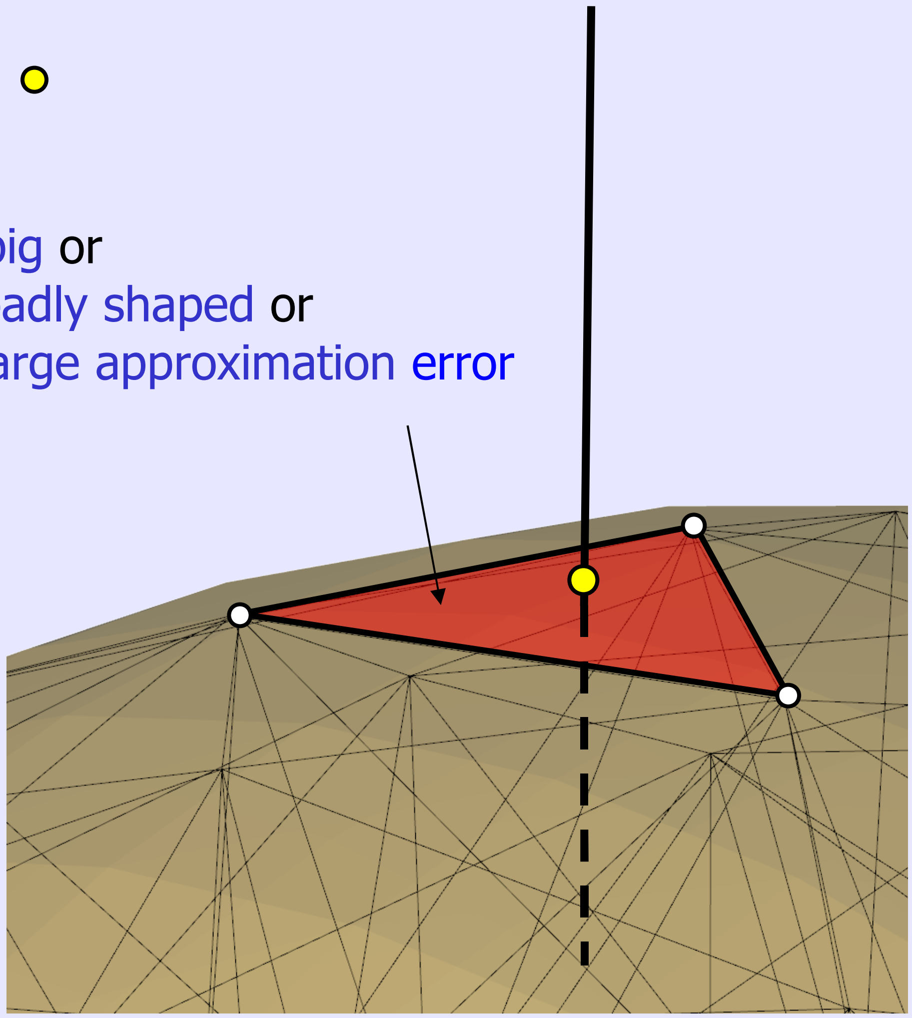
Delaunay triangulation restricted to surface S



Delaunay Refinement

Steiner point

Bad facet = big or
badly shaped or
large approximation error



Delaunay Refinement

repeat

{

pick bad facet **f**

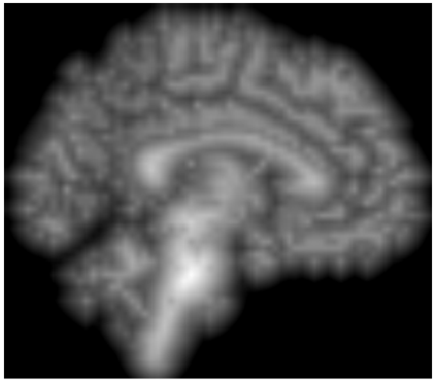
insert furthest (dual(**f**) $\cap$ **S**) in Delaunay triangulation

update Delaunay triangulation restricted to **S**

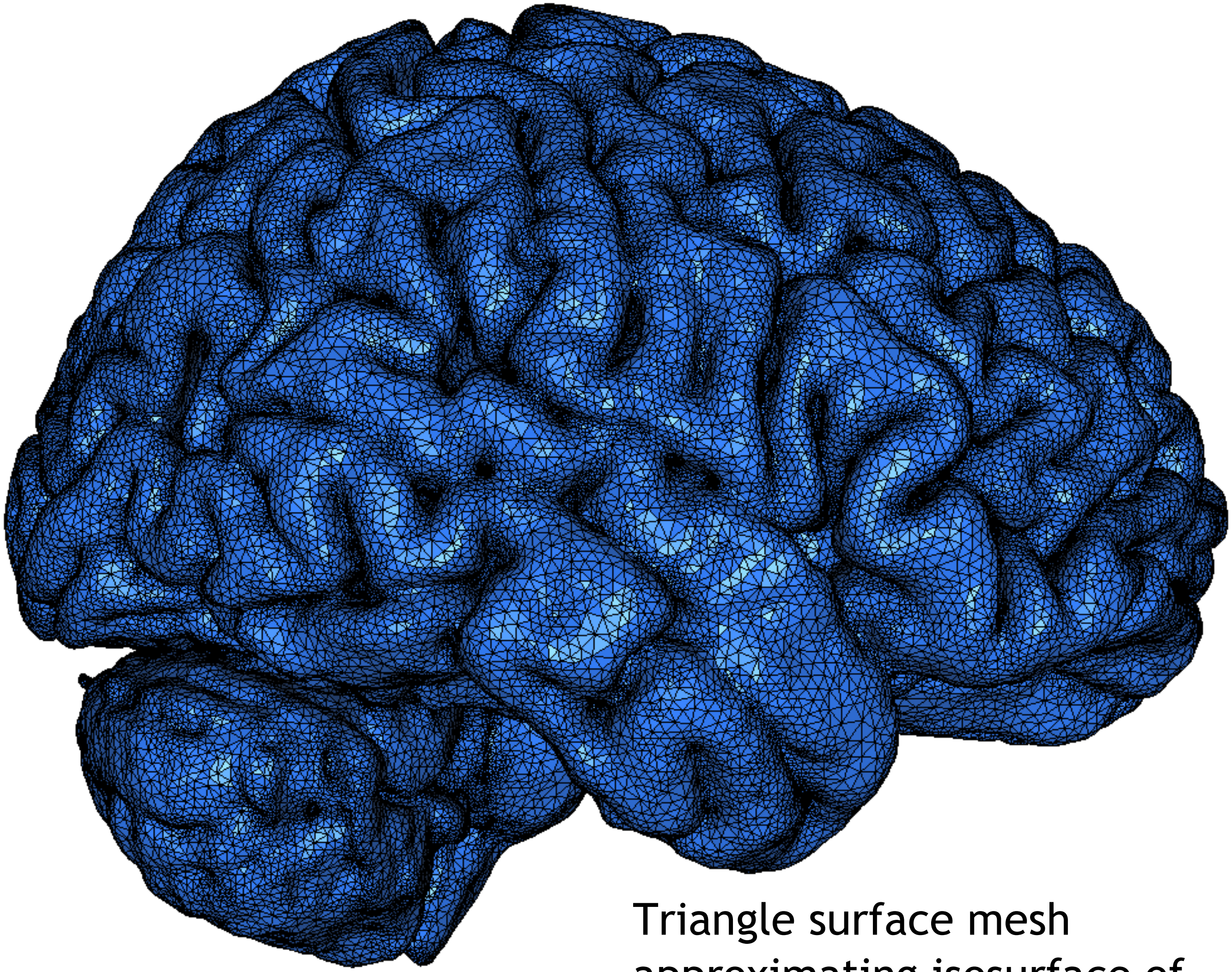
}

until all facets are good

Output Mesh



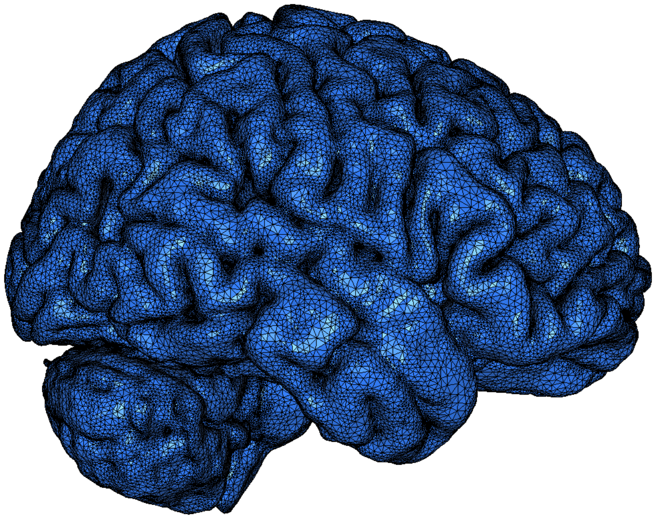
input



Triangle surface mesh approximating isosurface of input 3D image

Output Mesh

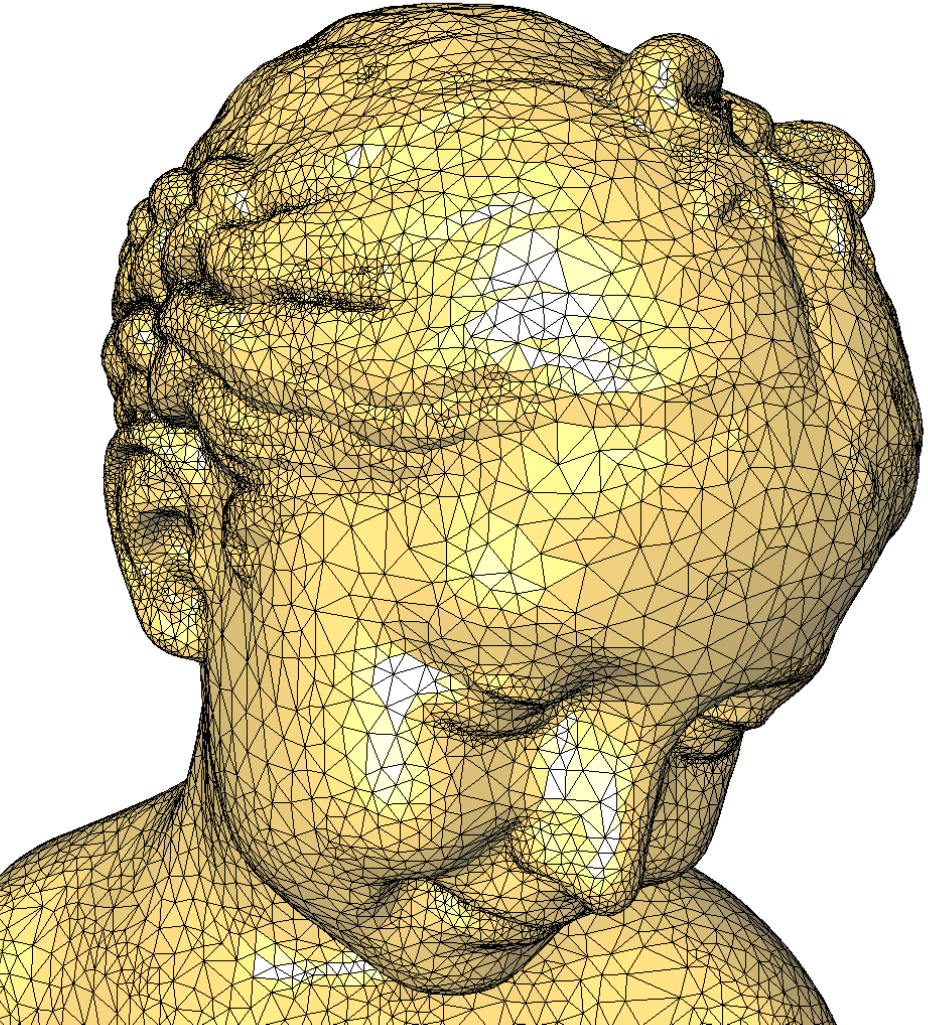
- Well shaped triangles
  - Lower bound on triangle angles
- Homeomorphic to input surface
- Manifold
  - not only combinatorially, i.e., no self-intersection
- Faithful Approximation of input surface
  - Hausdorff distance
  - Normals



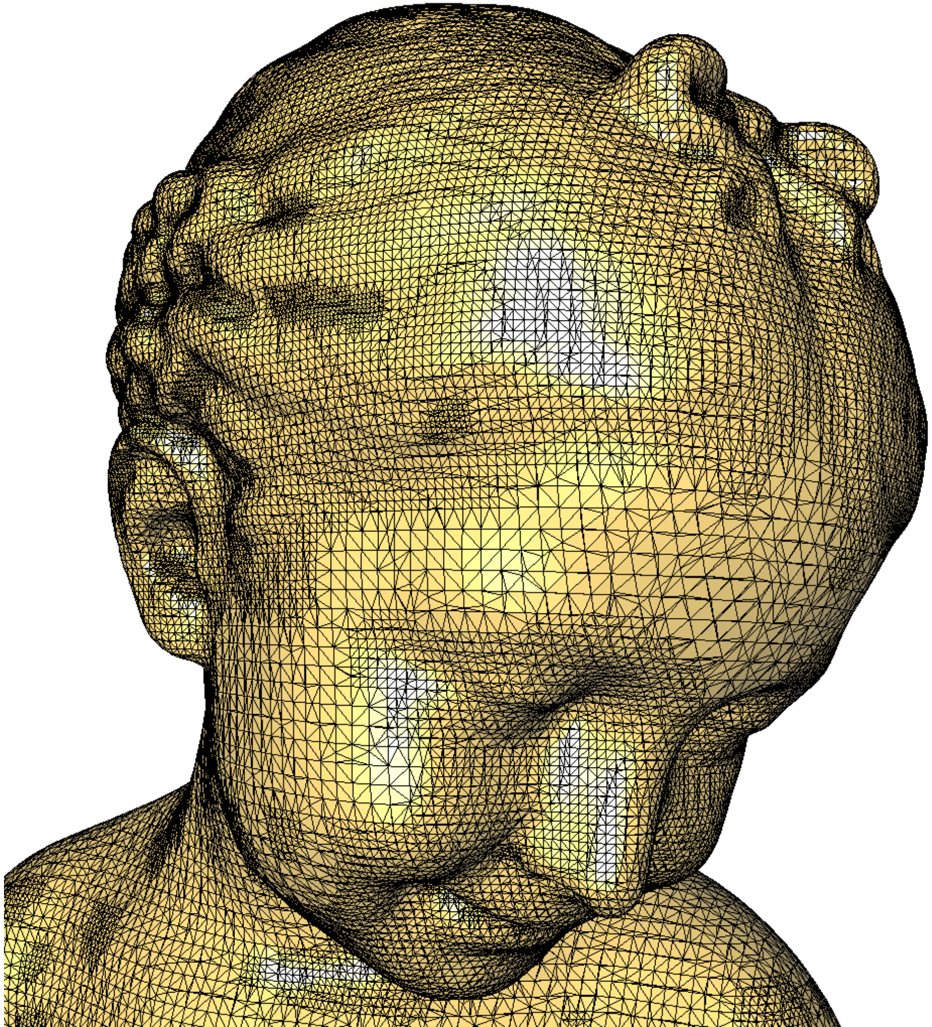
Online manual

vs Marching Cubes

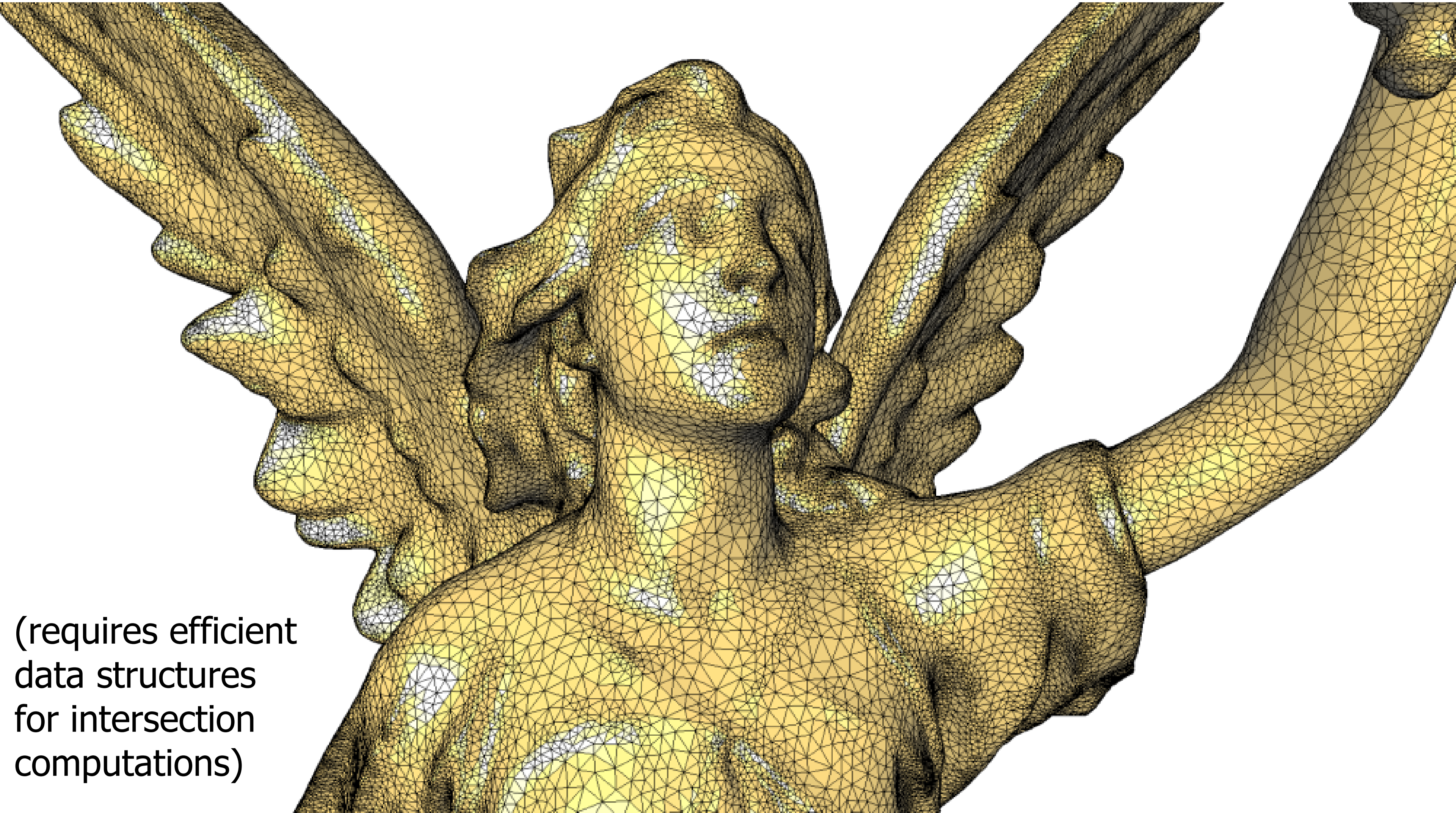
Delaunay refinement



Marching cubes in octree



Surface Remeshing (input is a polyhedral surface)



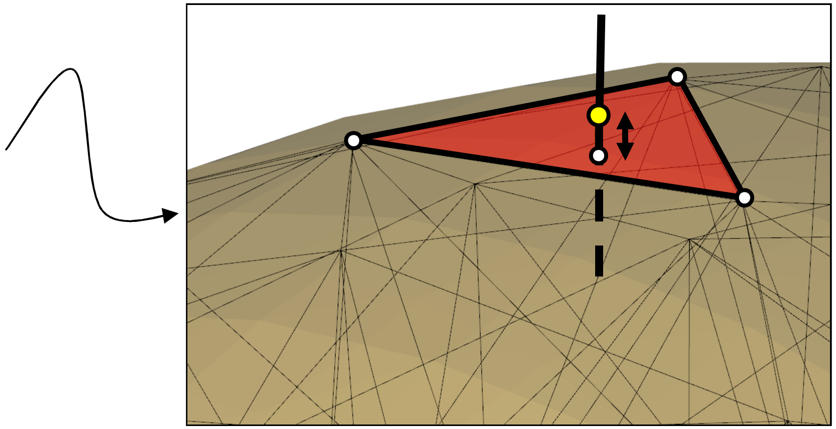
(requires efficient data structures for intersection computations)

Parameters

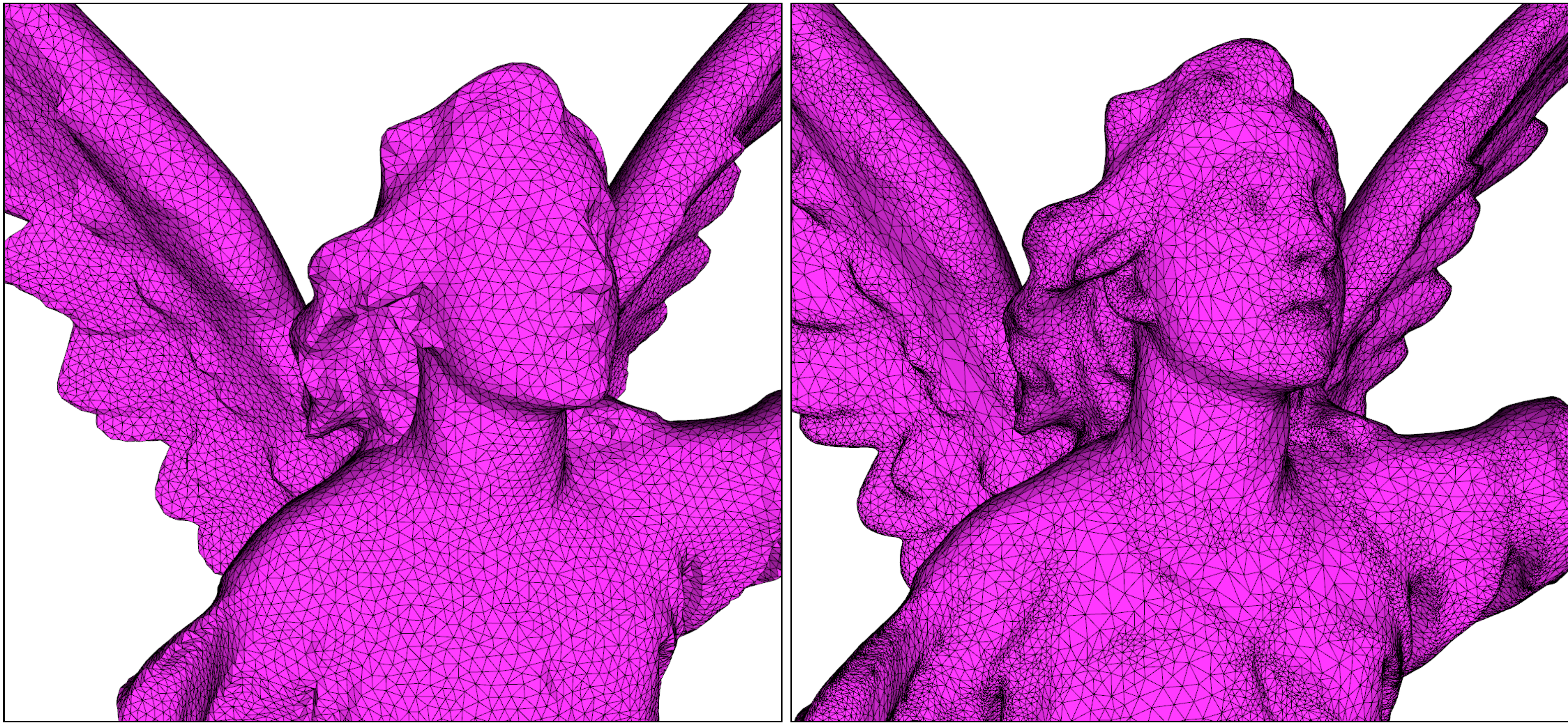
- Shape of triangles
  - lower bound on triangle angles
- Size
  - No constraint
  - Uniform sizing
  - Sizing function

Parameters

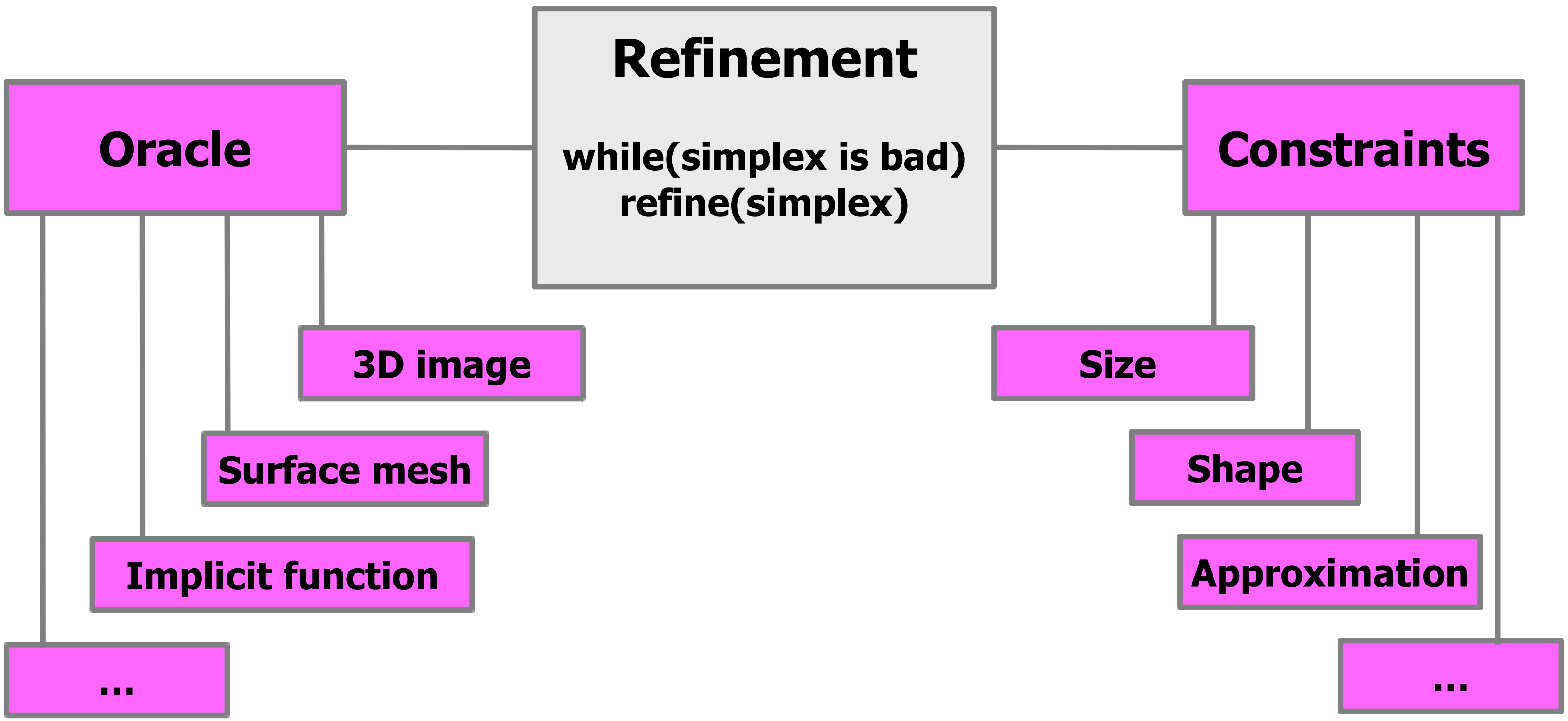
- Shape of triangles
  - lower bound on triangle angles
- Size
  - No constraint
  - Uniform sizing
  - Sizing function
- Approximation error



Uniform vs Adapted



Mesh Generation Framework

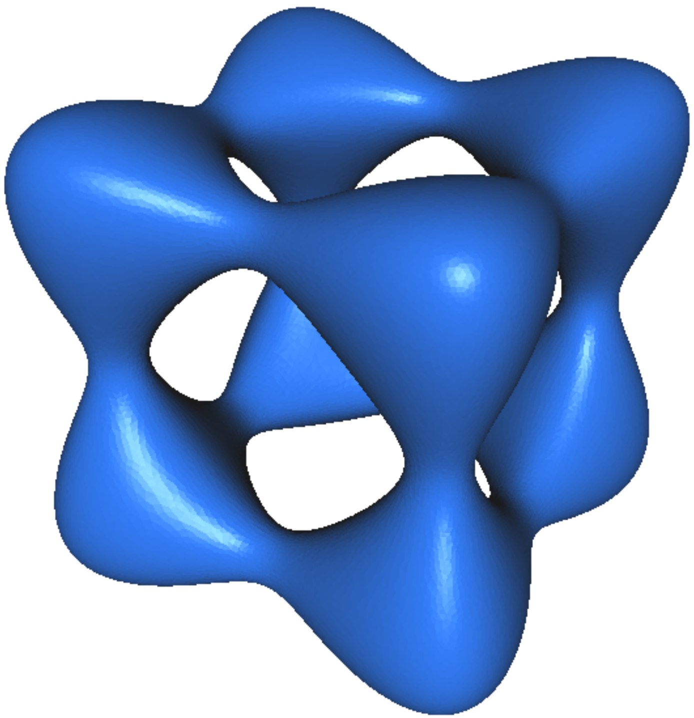


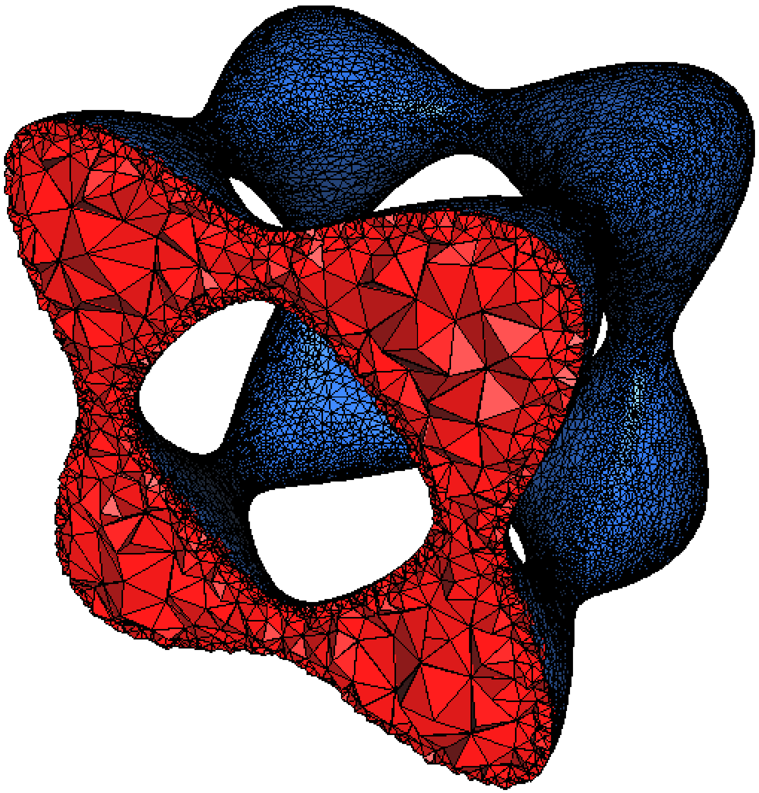
A Versatile Framework

- 3D grey level images
- 3D multi-domain images
- Implicit function: $f(x, y, z) = \text{constant}$
- Surface mesh (remeshing)
- Point set (surface reconstruction)
- Anything which provides intersections

3D Mesh Generation

3D (Volume) Mesh Generation

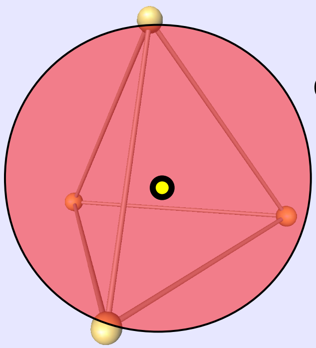




More Delaunay Filtering

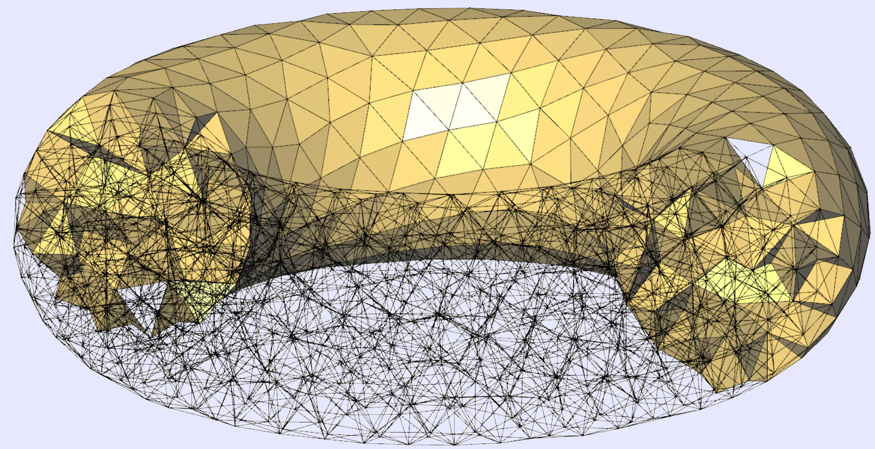
Delaunay triangulation restricted to domain Ω

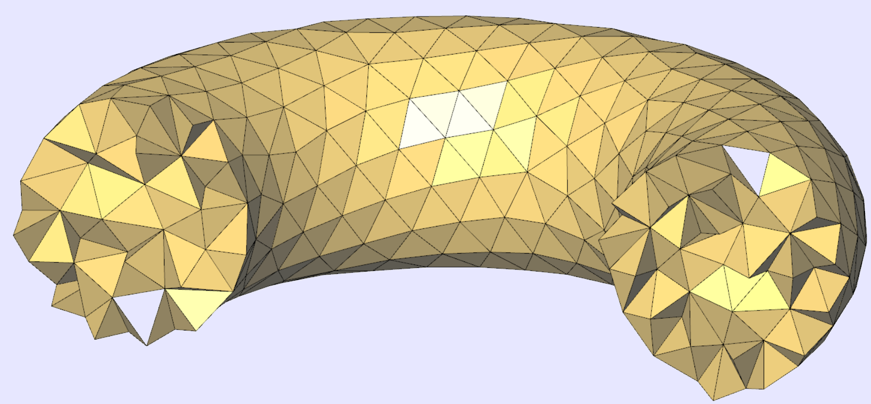
tetrahedron



circumsphere

Dual Voronoi vertex inside domain Ω (“oracle”)

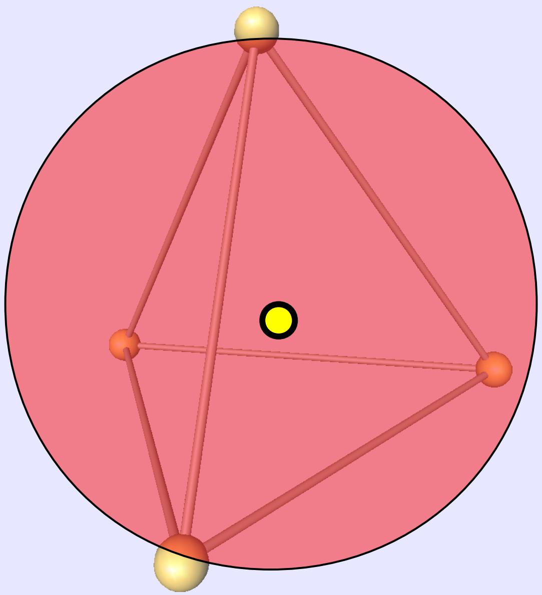




Delaunay Refinement

Steiner point 

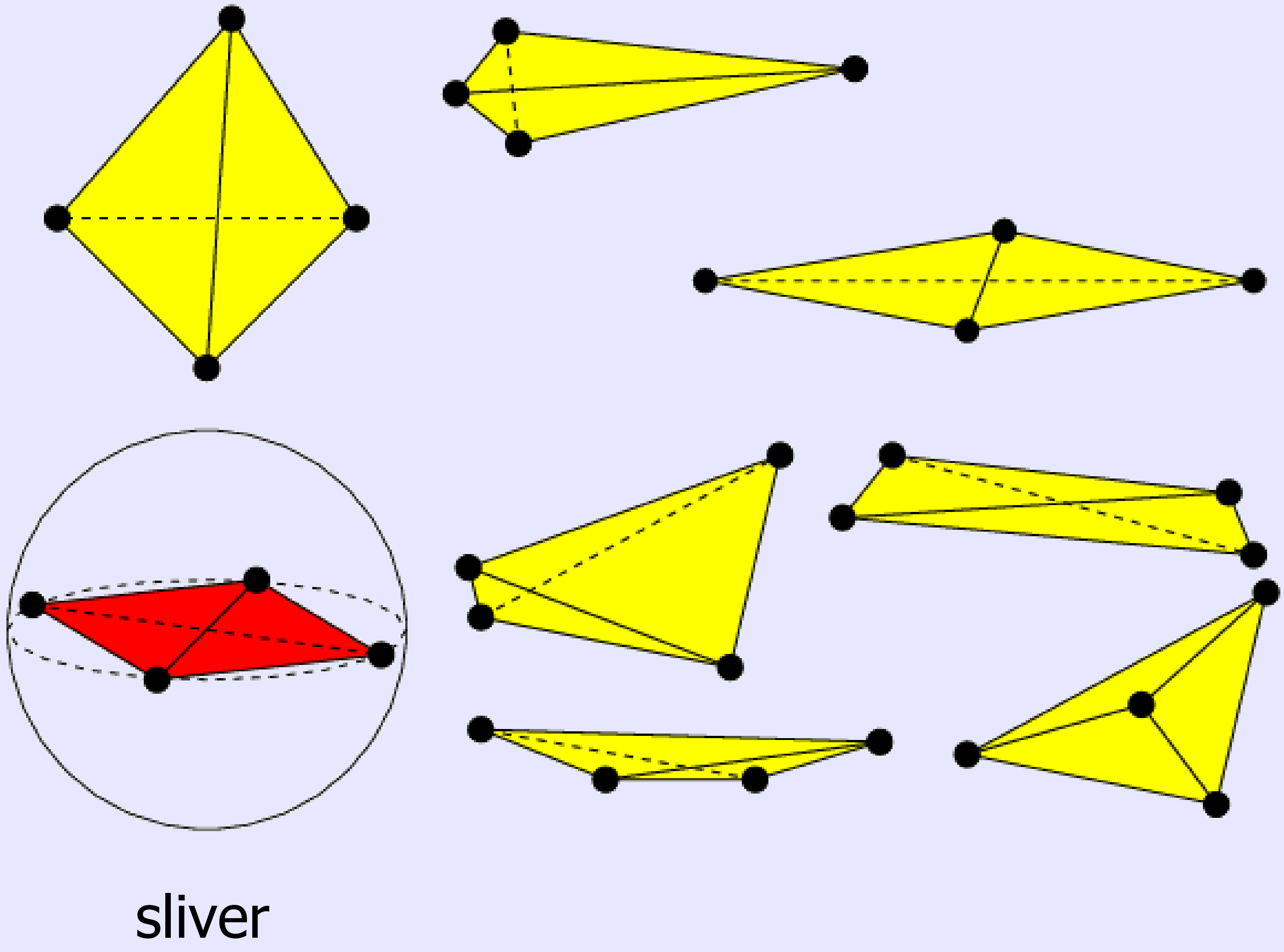
Bad tetrahedron = big or badly shaped



Volume Mesh Generation Algorithm

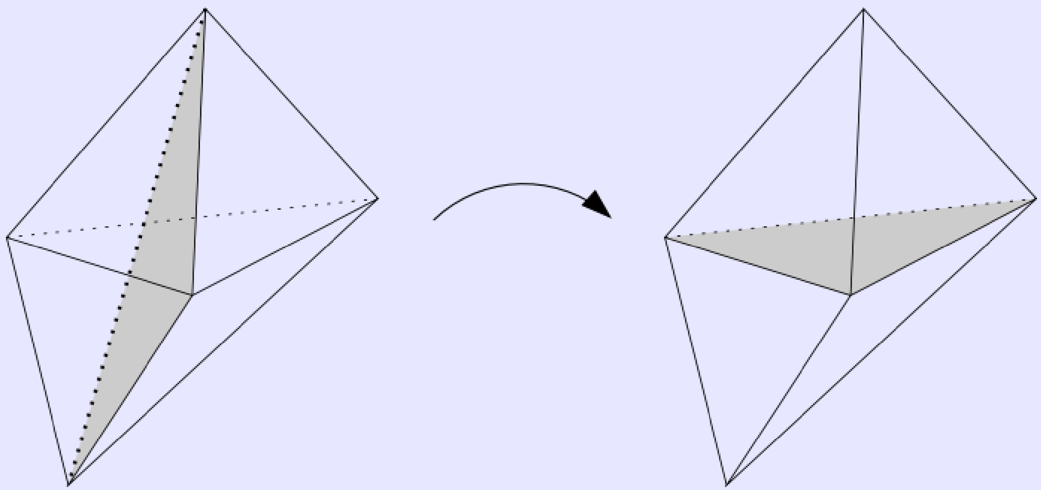
```
repeat
{
pick bad simplex
if(Steiner point encroaches a facet)
    refine facet
else
    refine simplex
update Delaunay triangulation restricted to domain
}
until all simplices are good
Exude slivers
```

Tetrahedron Zoo



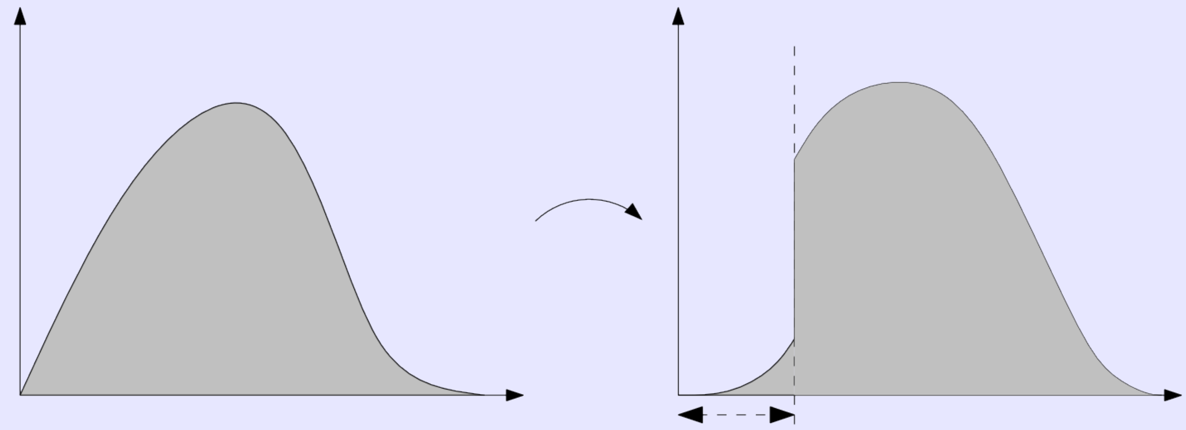
Sliver Exudation [Edelsbrunner-Guoy]

- Delaunay triangulation turned into a regular triangulation with null weights.
- Small increase of weights triggers edge-facets flips to remove slivers.



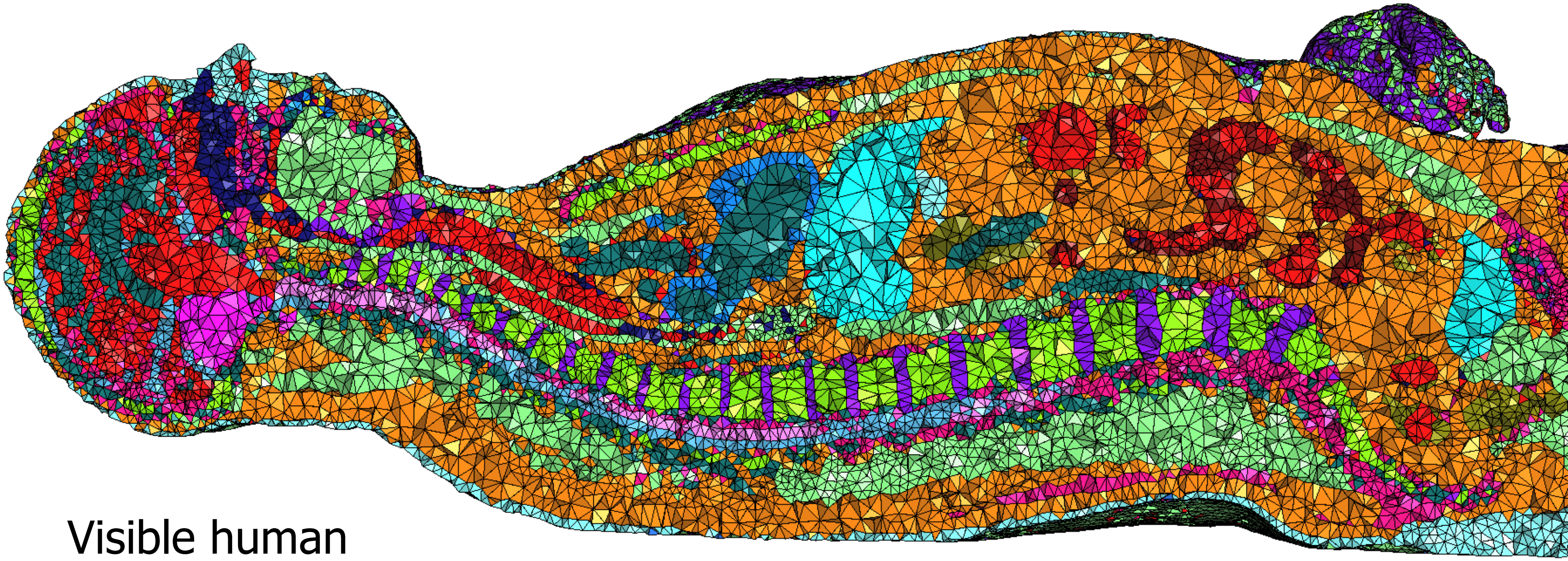
Sliver Exudation Process

- Try improving all tetrahedra with an aspect ratio lower than a given bound
- Never flips a boundary facet



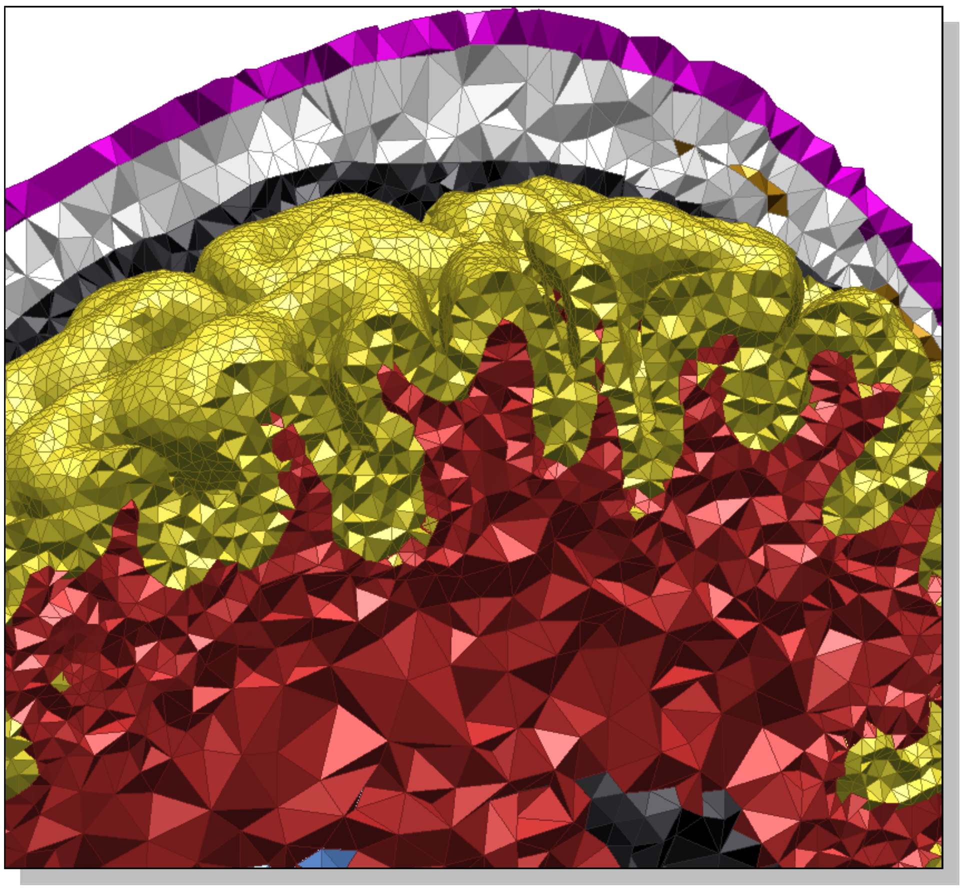
distribution of aspect ratios

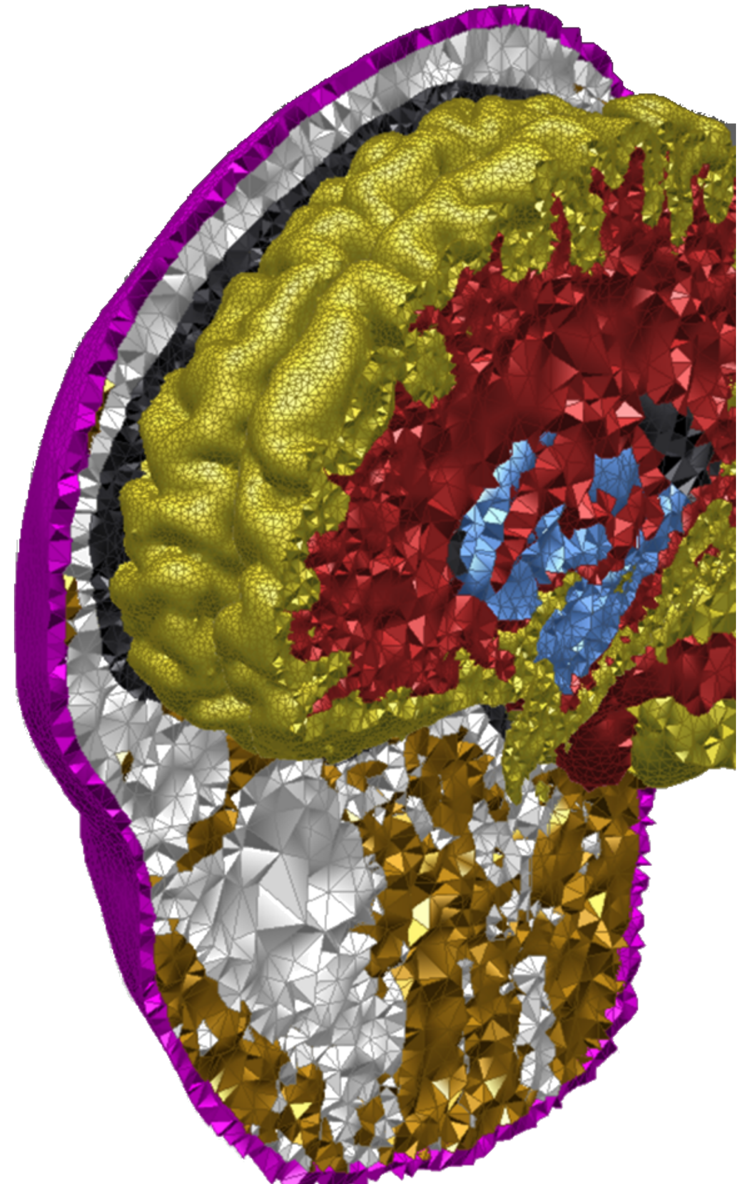
Multi-Domain Volume Mesh



Visible human

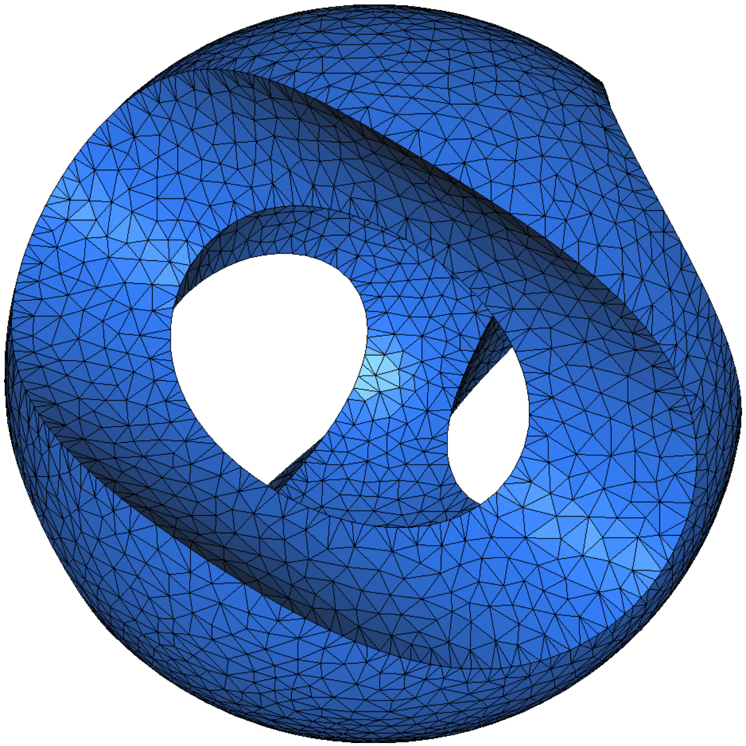
Multi-Domain Volume Mesh

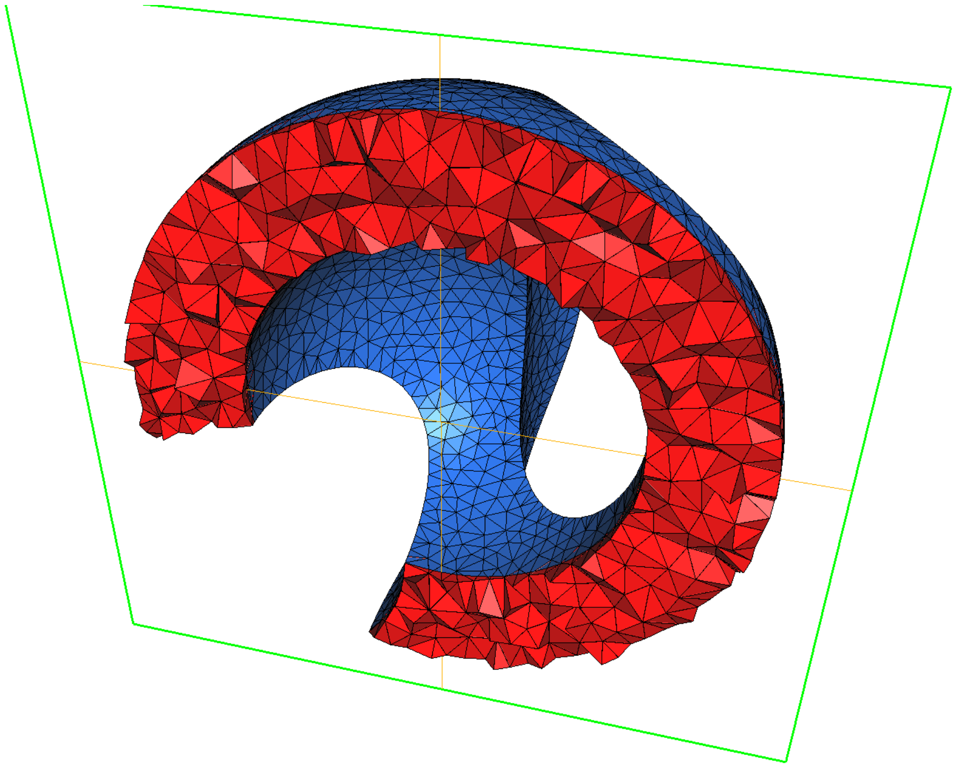




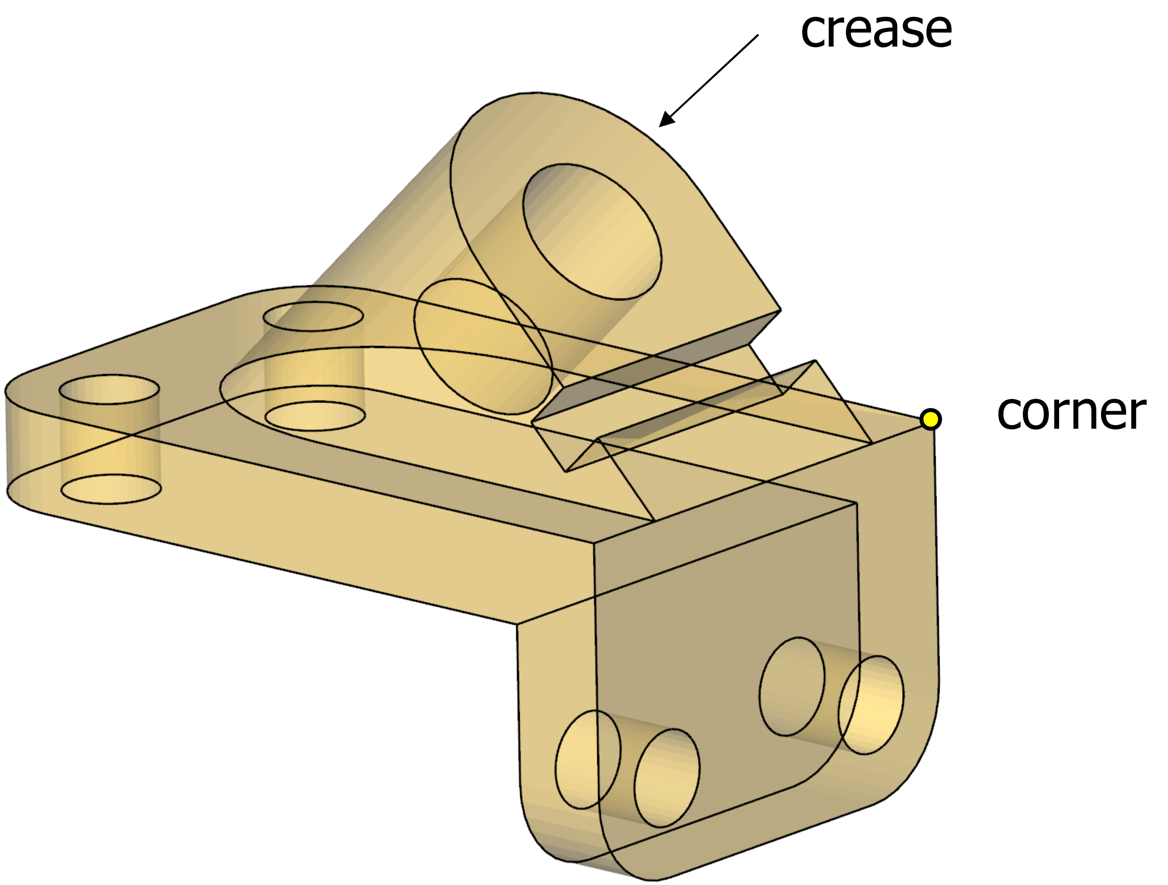
Work in Progress

Piecewise Smooth Surfaces



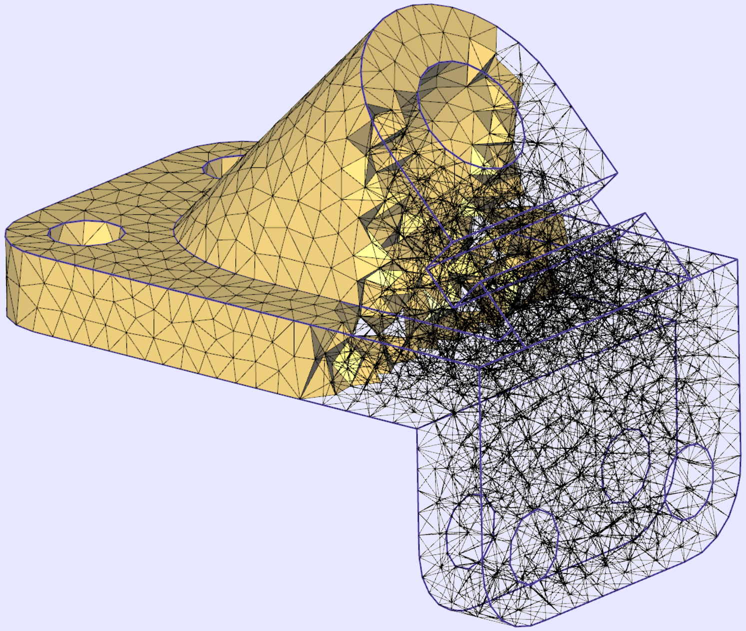


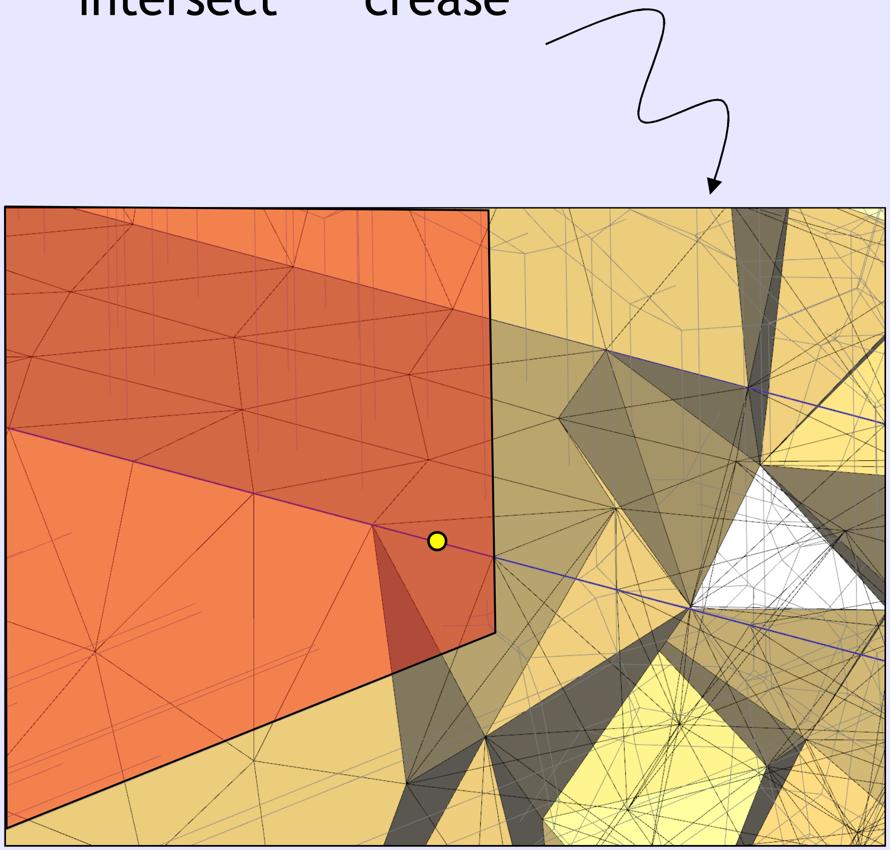
Input: Piecewise smooth complex



Even More Delaunay Filtering

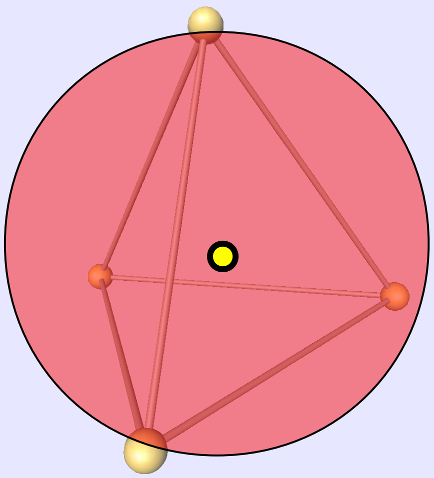
primitive	dual of	test	against
Voronoi vertex	tetrahedron	inside	domain
Voronoi edge	facet	intersect	domain boundary
Voronoi face	edge	intersect	crease

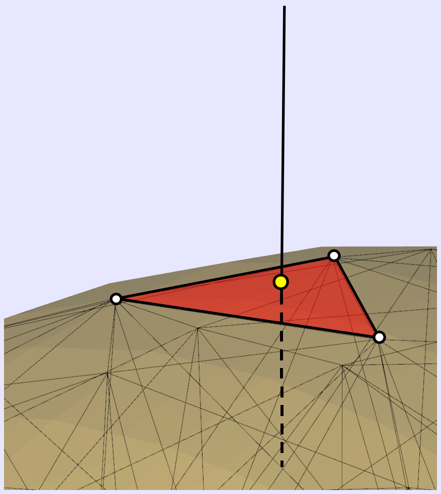


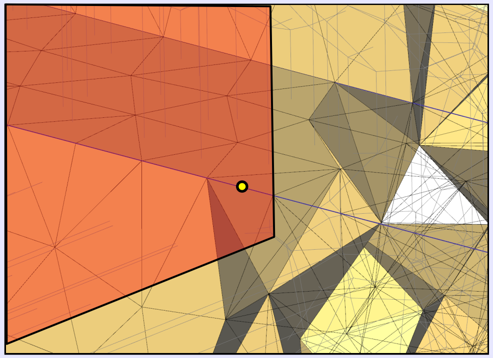


Delaunay Refinement

- Steiner points







Summary: CGAL for Mesh Generation

- 2D mesh generation
  - From triangulation to quality mesh
  - Preserves constraints exactly
- 3D Mesh generation
  - Interpolates boundary
  - Versatile through oracle-based design

Questions and Answers

Andreas Fabri
GeometryFactory

Pierre Alliez
INRIA

Question and Answers

- General Introduction
- CGAL for 2D Vector Graphics
- CGAL for Point Sets
- CGAL for Modeling and Processing of Polyhedral Surfaces
- CGAL for Mesh Generation