Mission Statement

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

CGAL EU Project Proposal, 1996

Project = « Planned Undertaking »

- Project partners make a long term commitment:
- INRIA, Tel-Aviv U, Max-Planck Institute, ETH Zurich, cnrs-LIRIS, GeometryFactory,...

- CGAL Editorial Board
  - Steers and animates the project
  - Reviews submissions
  - Release manager, Review manager, ...

- Development infrastructure

CGAL in Numbers

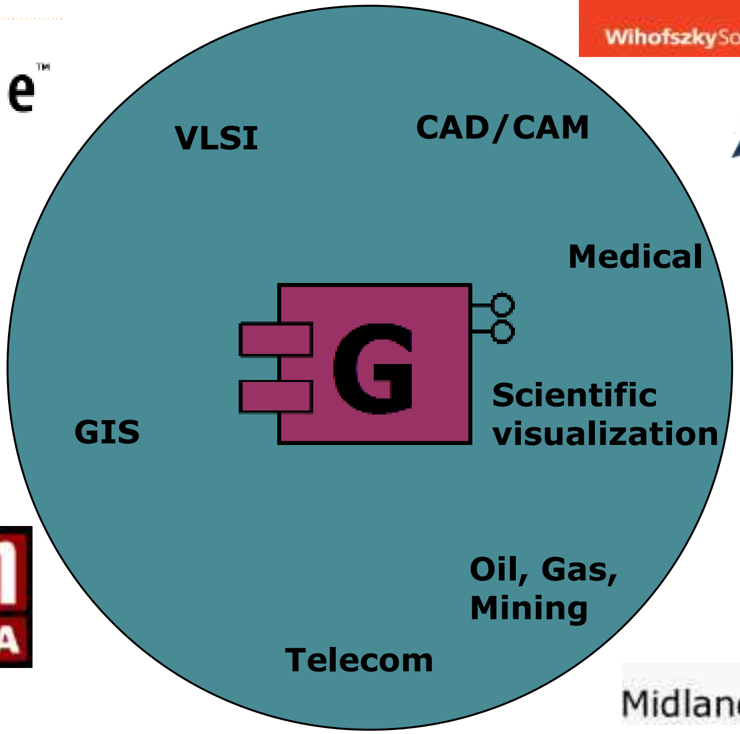
600,000	lines of C++ code
10,000	downloads/year (+ package managers)
4,500	manual pages
3,000	subscribers to cgal-announce
1,000	subscribers to cgal-discuss
200	commercial users
	software components
120	active developers
20	months release cycle
6	licenses: Open Source + Commercial
2	

Licenses

- Open Source License:
  - LGPL for Foundation Layer
  - GPL for other packages

- Commercial Licenses
  - Annual Research Licenses for entire CGAL
  - Commercial License for components
  -

Some Commercial CGAL Users



VLSI

CAD/CAM

Medical

Scientific visualization

GIS

Oil, Gas, Mining

Telecom

Orbotech

pulsic Finish First

ECT

TOSHIBA

cadence™

Agilent Technologies

BAE SYSTEMS

QinetiQ

erdas

Virtualwind

SAFE SOFTWARE

rm DATA

ARCHI VIDEO

ESRI

TruePosition

FNVITIA

Trimble

orange™

BT

protomold®

Autodesk®

Exa CORPORATION

CMET シーメット株式会社

WihofszkySoftware

MPC THEMOVINGPICTURECOMPANY

ECL

schaerermayfield WHEREVER YOU OPERATE

GE Healthcare

TOMTEC IMAGING SYSTEMS

ST. JUDE MEDICAL

PRESAGIS

School of Biomedical Sciences Excellence in Teaching 5* Research

INDUSTRIAL RESEARCH LIMITED

The MathWorks

Midland Valley

roxar

Agip

TOTAL

Schlumberger

BR PETROBRAS

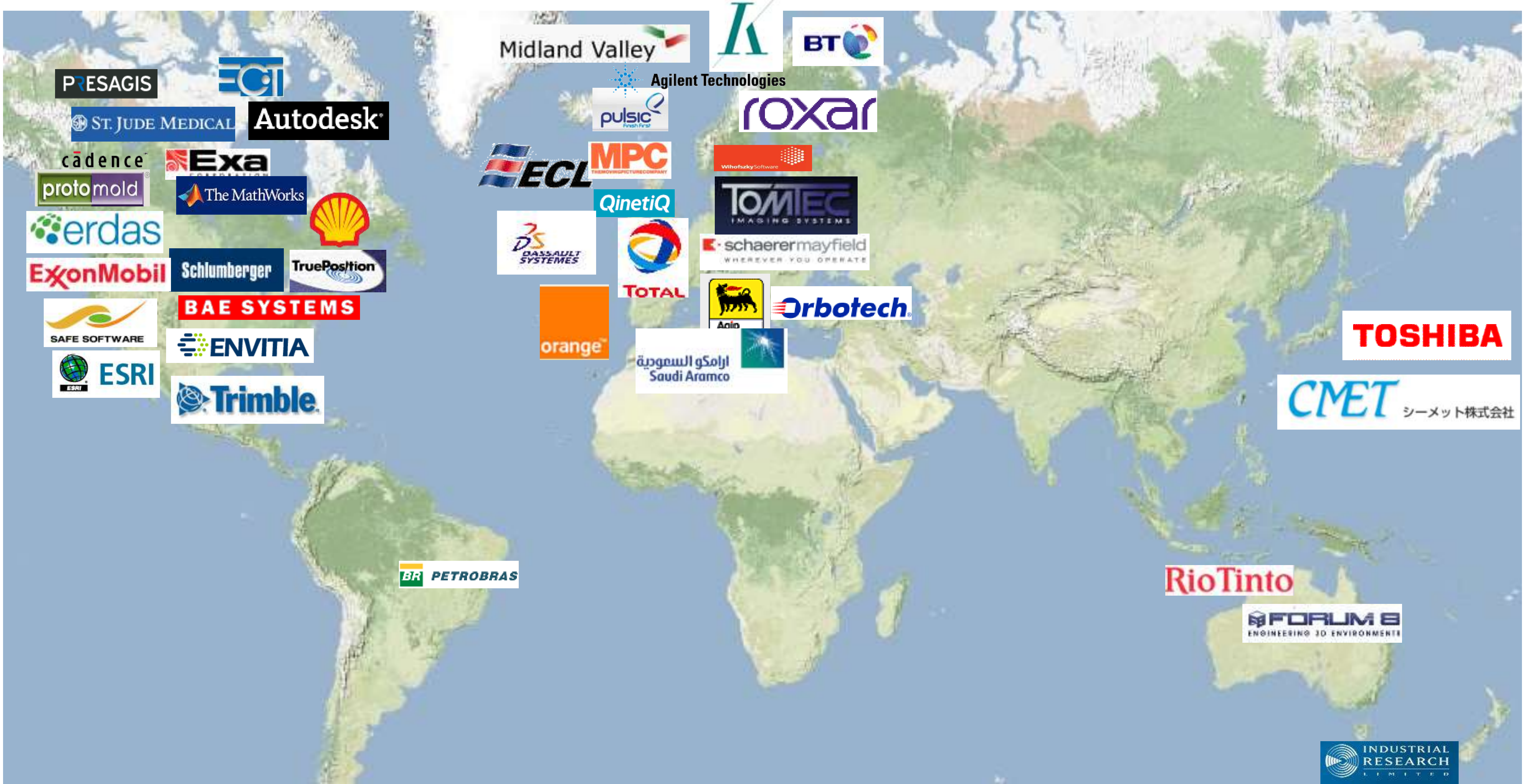
ارامكو السعودية Saudi Aramco

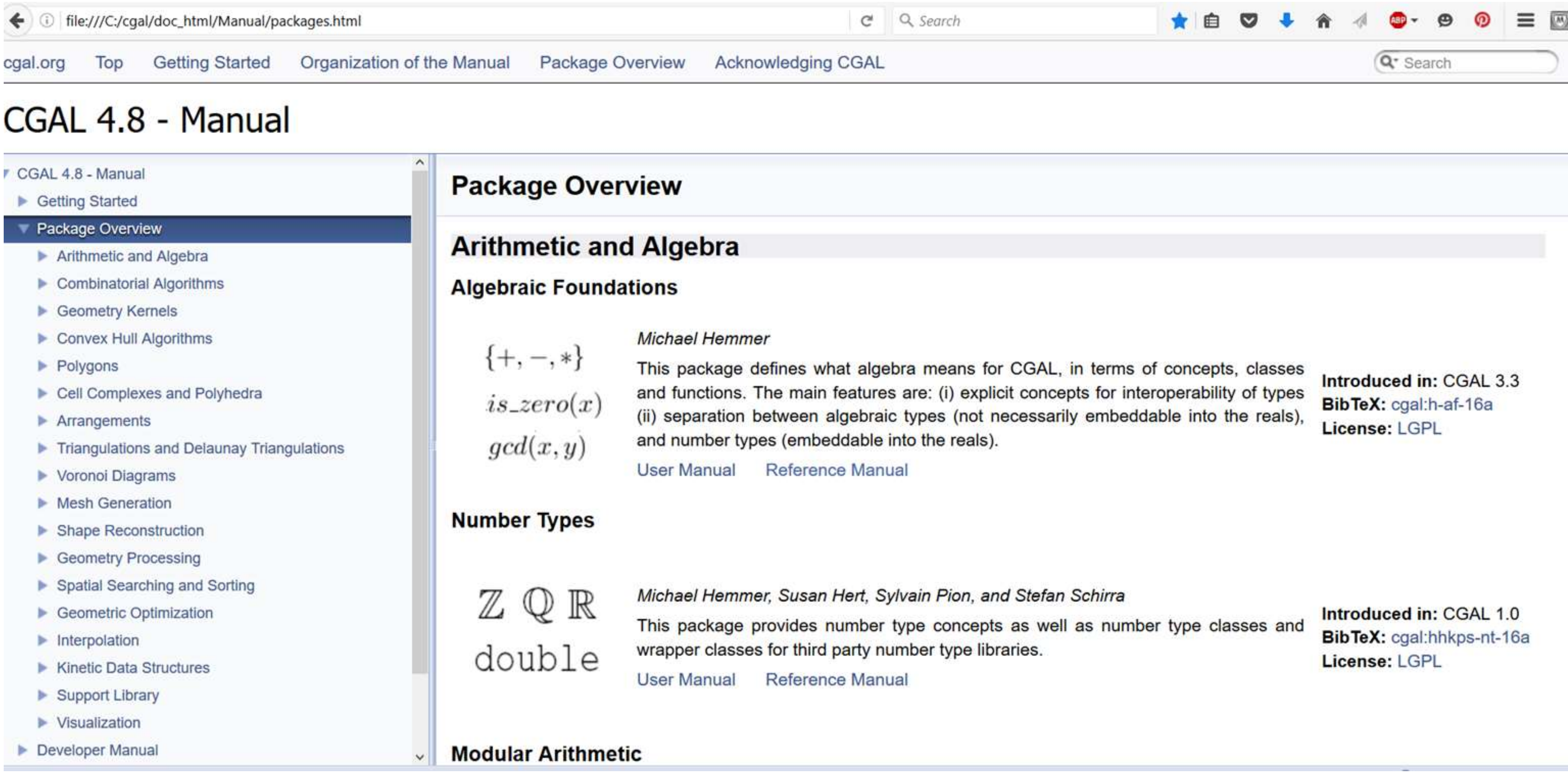
RioTinto

brgm

ExxonMobil

Some Commercial CGAL Users





CGAL 4.8 - Manual

Package Overview

Arithmetic and Algebra

Algebraic Foundations

$\{+,-,*\}$
$is\_zero(x)$
$gcd(x,y)$

Michael Hemmer

This package defines what algebra means for CGAL, in terms of concepts, classes and functions. The main features are: (i) explicit concepts for interoperability of types (ii) separation between algebraic types (not necessarily embeddable into the reals), and number types (embeddable into the reals).

User Manual   Reference Manual

Introduced in: CGAL 3.3
BibTeX: cgal:h-af-16a
License: LGPL

Number Types

$\mathbb{Z}$ $\mathbb{Q}$ $\mathbb{R}$
double

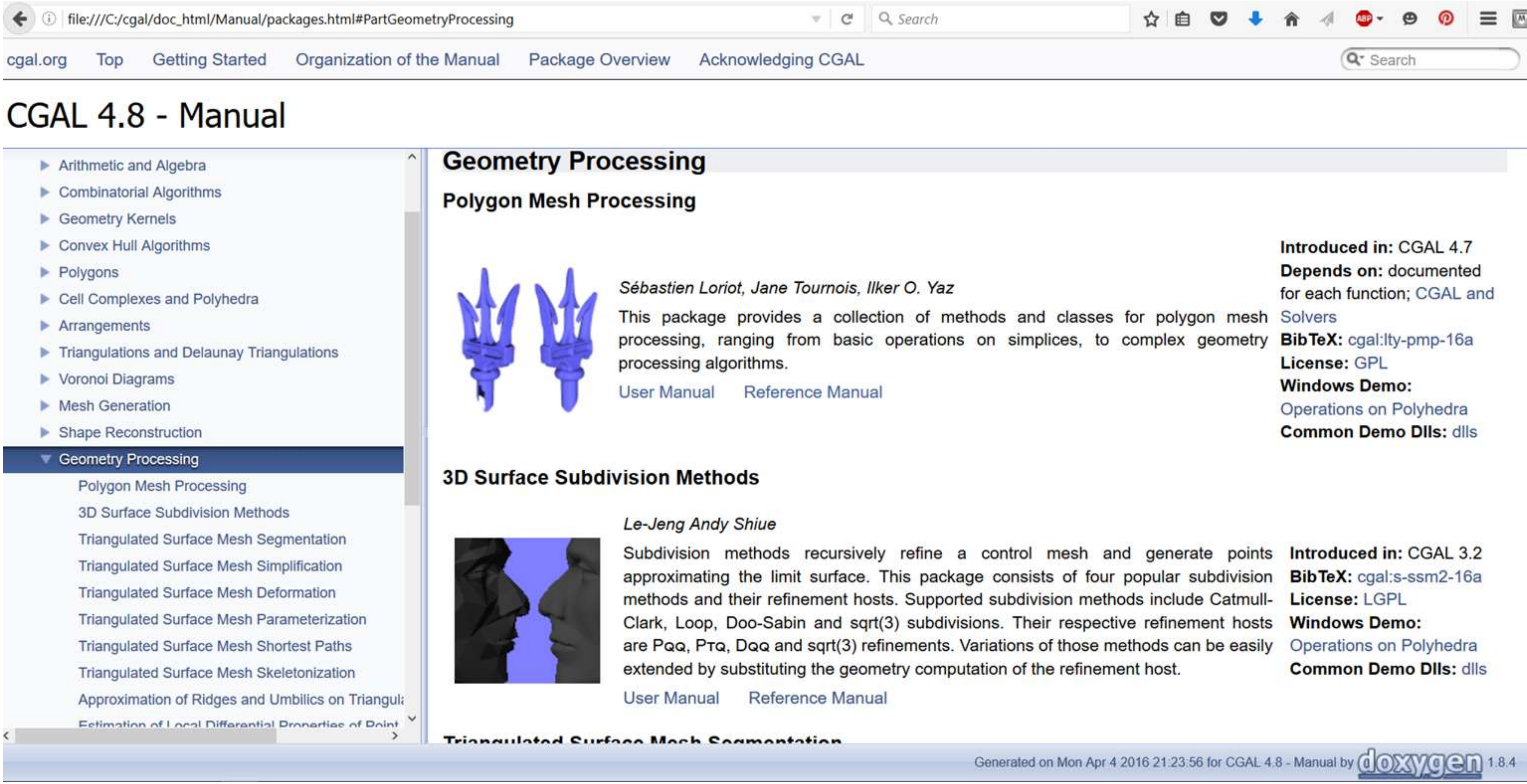
Michael Hemmer, Susan Hert, Sylvain Pion, and Stefan Schirra

This package provides number type concepts as well as number type classes and wrapper classes for third party number type libraries.

User Manual   Reference Manual

Introduced in: CGAL 1.0
BibTeX: cgal:hhkps-nt-16a
License: LGPL

Modular Arithmetic


Search
cgal.org
Top
Getting Started
Organization of the Manual
Package Overview
Acknowledging CGAL
Search
Arithmetic and Algebra
Combinatorial Algorithms
Geometry Kernels
Convex Hull Algorithms
Polygons
Cell Complexes and Polyhedra
Arrangements
Triangulations and Delaunay Triangulations
Voronoi Diagrams
Mesh Generation
Shape Reconstruction
Geometry Processing
Polygon Mesh Processing
3D Surface Subdivision Methods
Triangulated Surface Mesh Segmentation
Triangulated Surface Mesh Simplification
Triangulated Surface Mesh Deformation
Triangulated Surface Mesh Parameterization
Triangulated Surface Mesh Shortest Paths
Triangulated Surface Mesh Skeletonization
Approximation of Ridges and Umbilics on Triangul
Estimation of Local Differential Properties of Point

file:///C:/cgal/doc_html/Manual/packages.html#PartGeometryProcessing

CGAL 4.8 - Manual

Geometry Processing

Polygon Mesh Processing

Sébastien Loriot, Jane Tournois, Ilker O. Yaz

This package provides a collection of methods and classes for polygon mesh processing, ranging from basic operations on simplices, to complex geometry processing algorithms.

User Manual Reference Manual

Introduced in: CGAL 4.7
Depends on: documented for each function; CGAL and Solvers
BibTeX: cgal:lty-pmp-16a
License: GPL
Windows Demo: Operations on Polyhedra
Common Demo Dlls: dlls

3D Surface Subdivision Methods

Le-Jeng Andy Shiue

Subdivision methods recursively refine a control mesh and generate points approximating the limit surface. This package consists of four popular subdivision methods and their refinement hosts. Supported subdivision methods include Catmull-Clark, Loop, Doo-Sabin and sqrt(3) subdivisions. Their respective refinement hosts are PQQ, PTQ, DQQ and sqrt(3) refinements. Variations of those methods can be easily extended by substituting the geometry computation of the refinement host.

User Manual Reference Manual

Introduced in: CGAL 3.2
BibTeX: cgal:s-ssm2-16a
License: LGPL
Windows Demo: Operations on Polyhedra
Common Demo Dlls: dlls

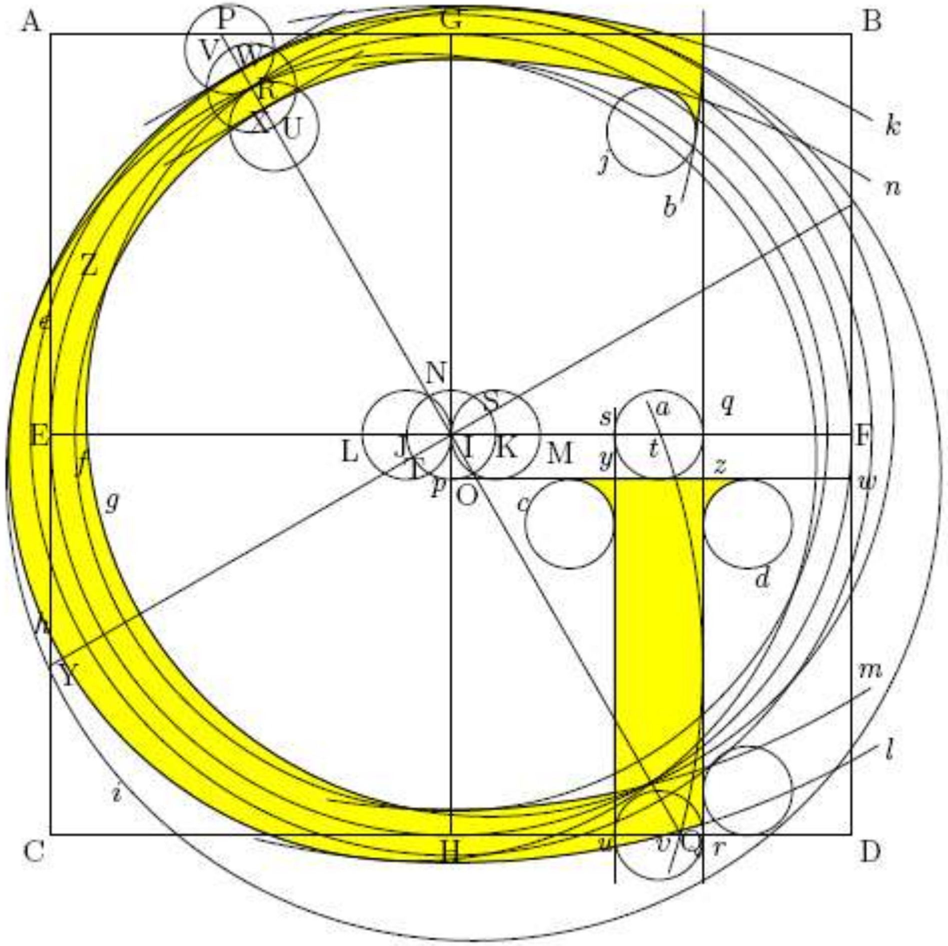
Triangulated Surface Mesh Segmentation

Generated on Mon Apr 4 2016 21:23:56 for CGAL 4.8 - Manual by doxygen 1.8.4

Outline

- Point Set Processing
- Polygon Mesh Processing
- Mesh Generation

- Use, Participate, Contribute



Point Set Processing

Background

cgal.org

KD Tree

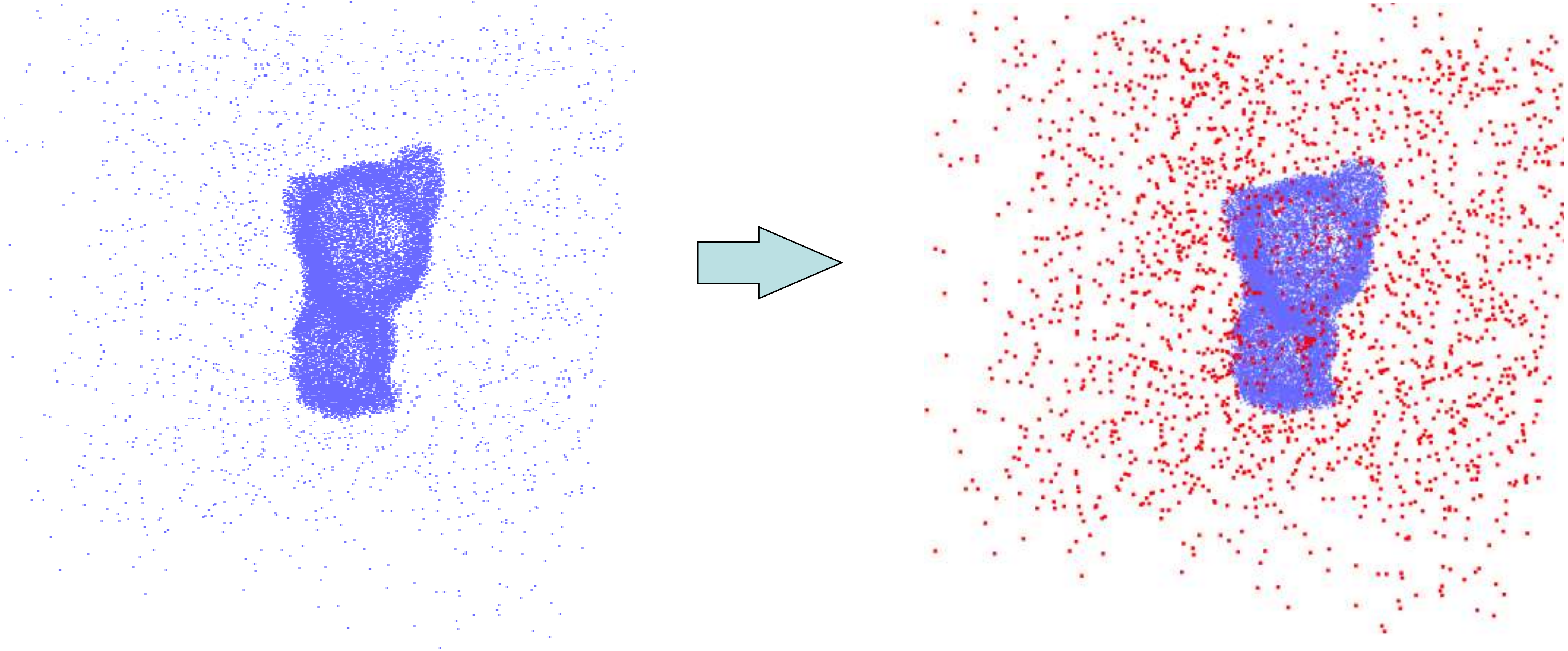
- Range queries:
- Get points that lie in a query iso-cuboid, or sphere

- Distance queries:
- Get m closest points to a query point

cgal.org

Outlier Removal

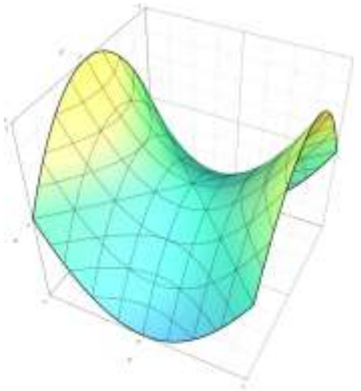
- Sort points by average squared distances to m nearest neighbors
- Cut at specified percentile

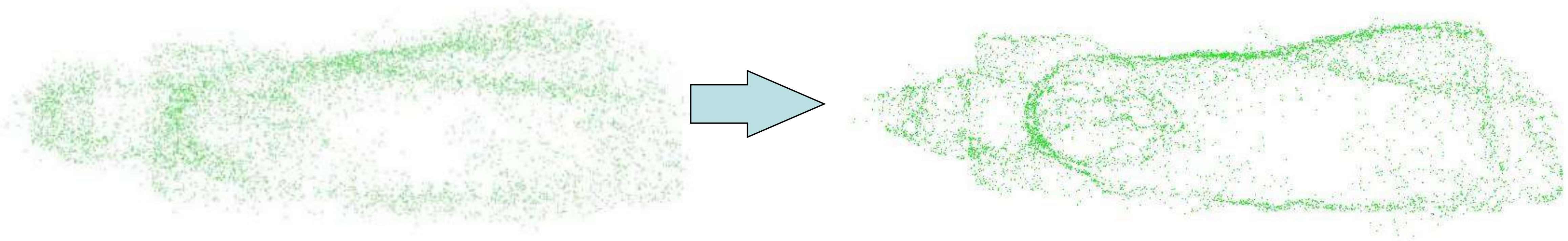


cgal.org

Smoothing

- For each point
  - Find m nearest neighbors
  - Fit plane or jet (smooth parametric surface)
  - Project





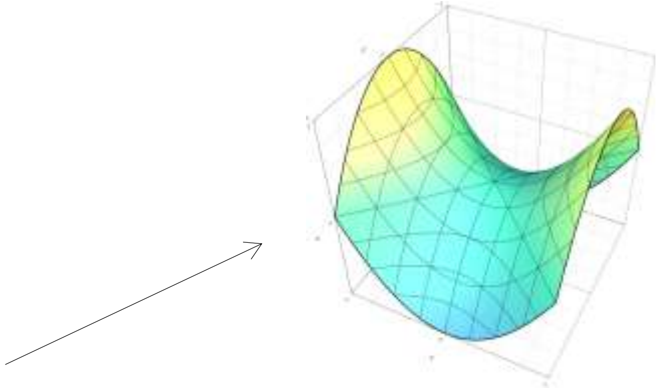
(noisy point set)

(smoothed point set)

cgal.org

Normal Estimation

- For each point
  - Find m nearest neighbors
  - Fit plane or smooth jet surface
  - Project

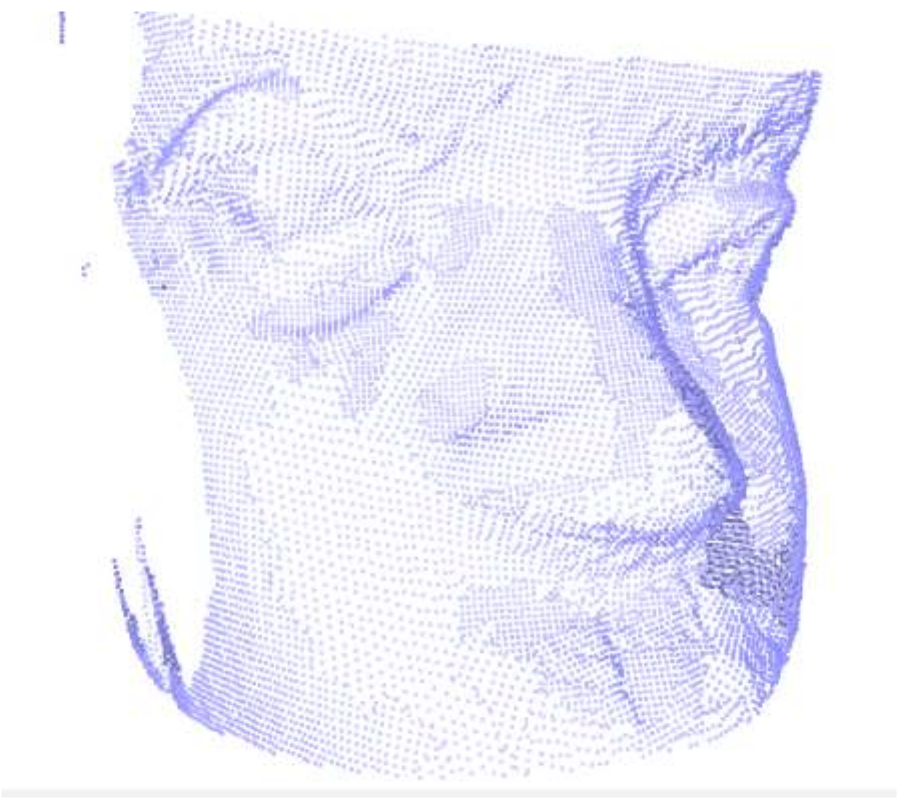


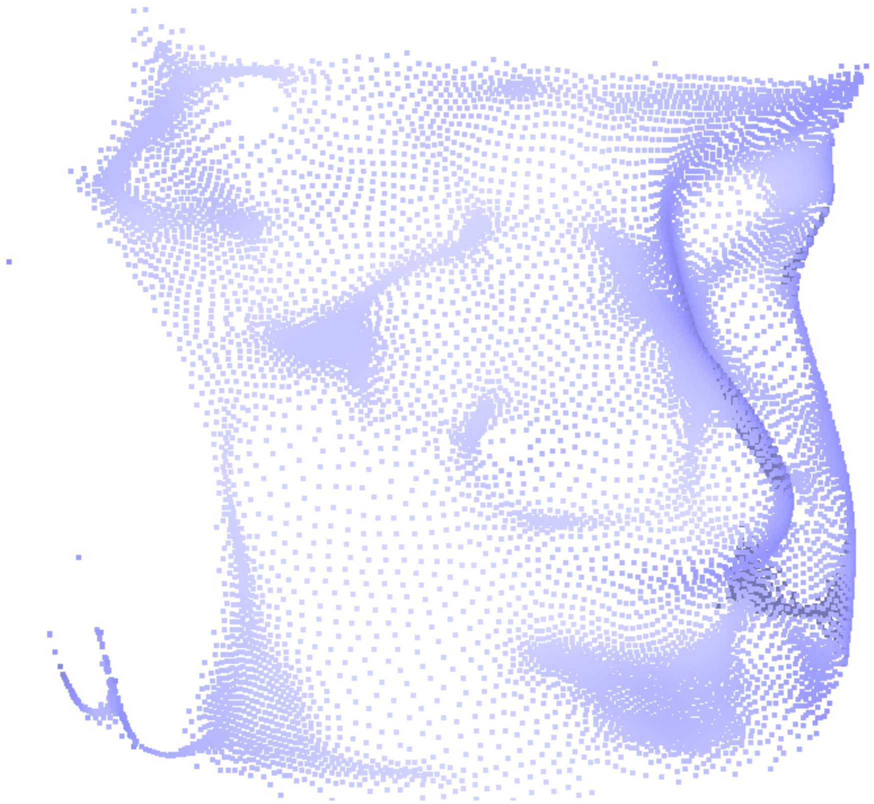
- Orient normals by propagation in Riemannian graph (edges between k NN)

cgal.org

WLOP Simplification

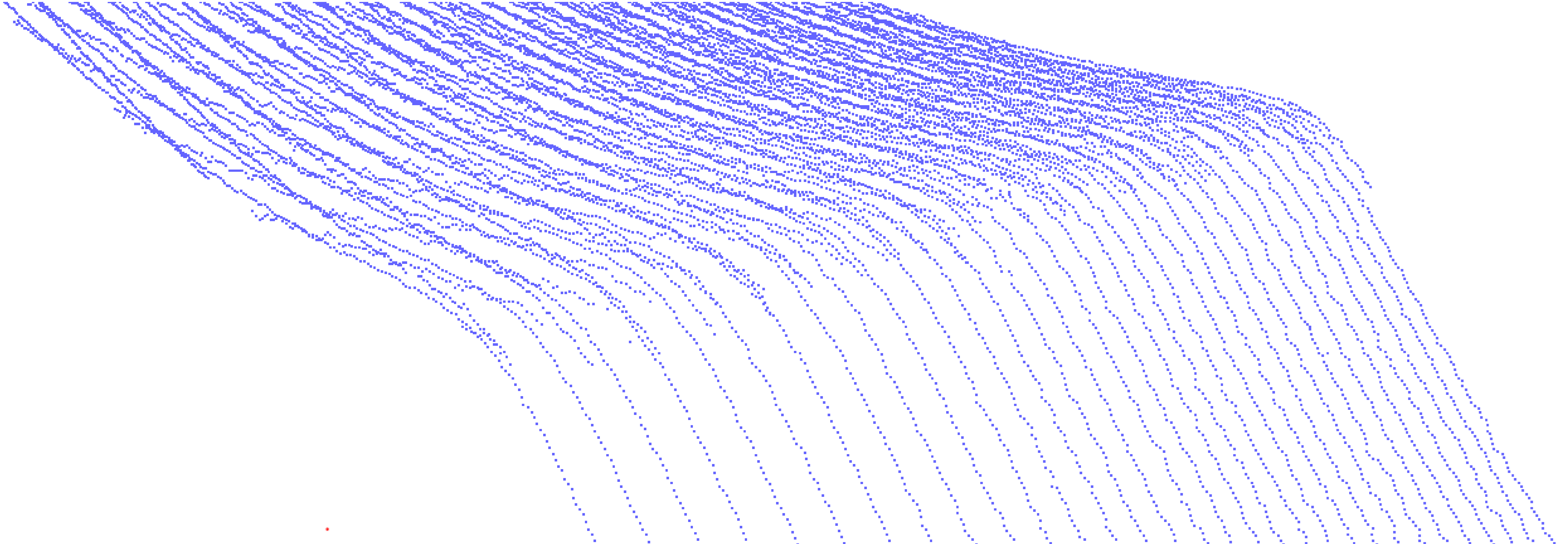
- Weighted Locally Optimal Projection [Wu et al.]
- Distributes particles applying contraction and repulsion forces





Warning Concerning All Algorithms

The m nearest neighbors may be on same scan line

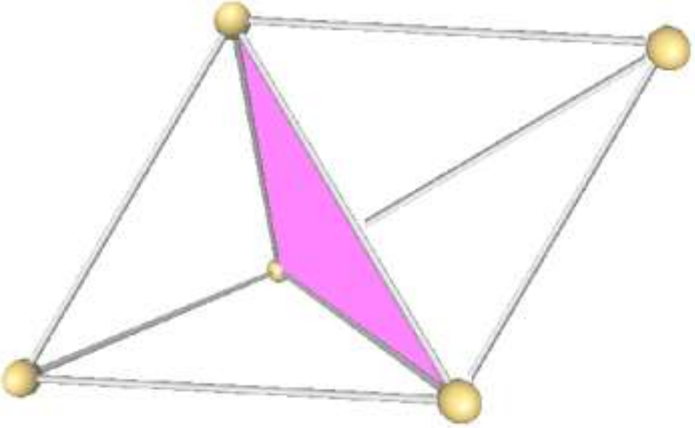


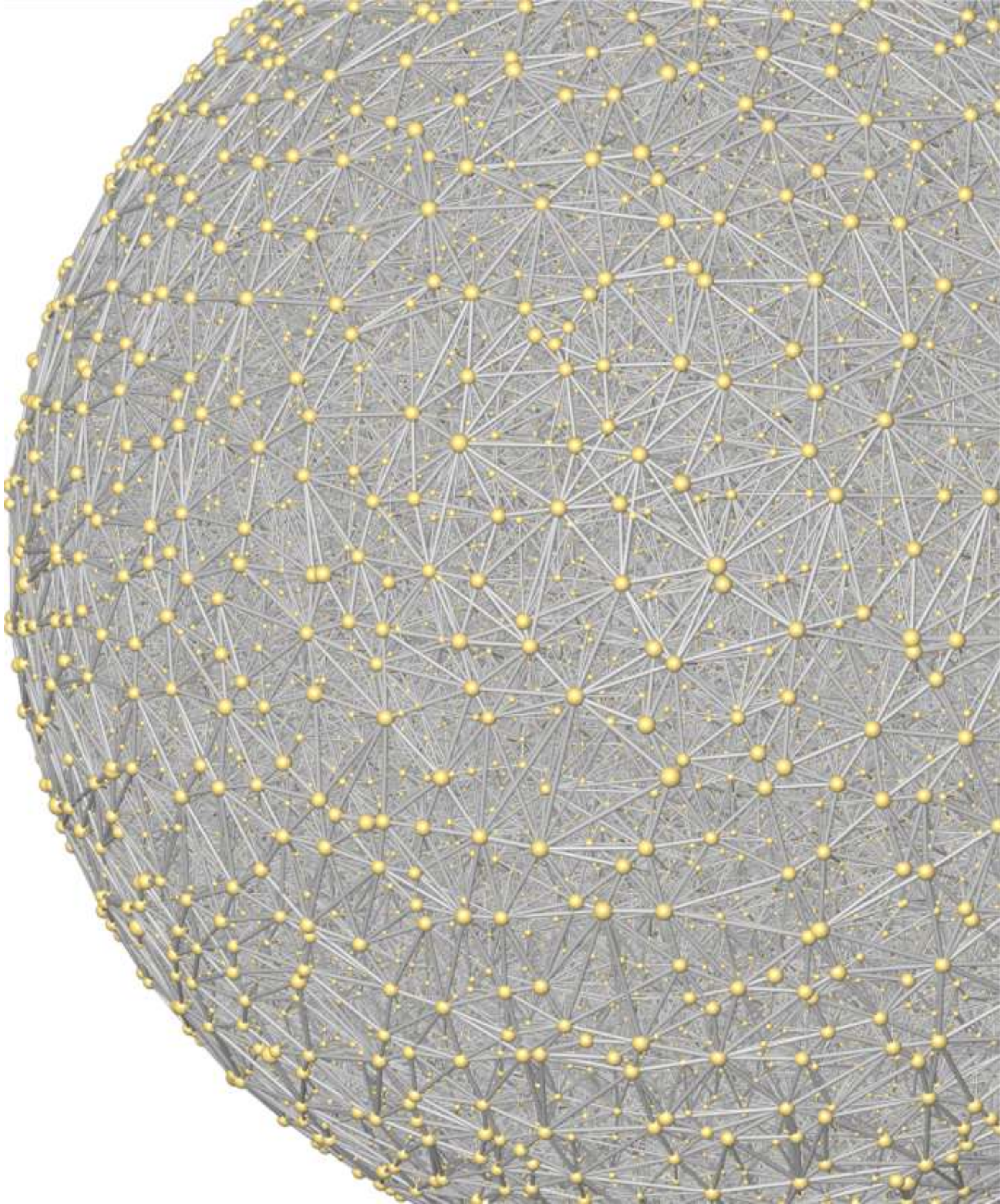
Surface Reconstruction

Surface Reconstruction

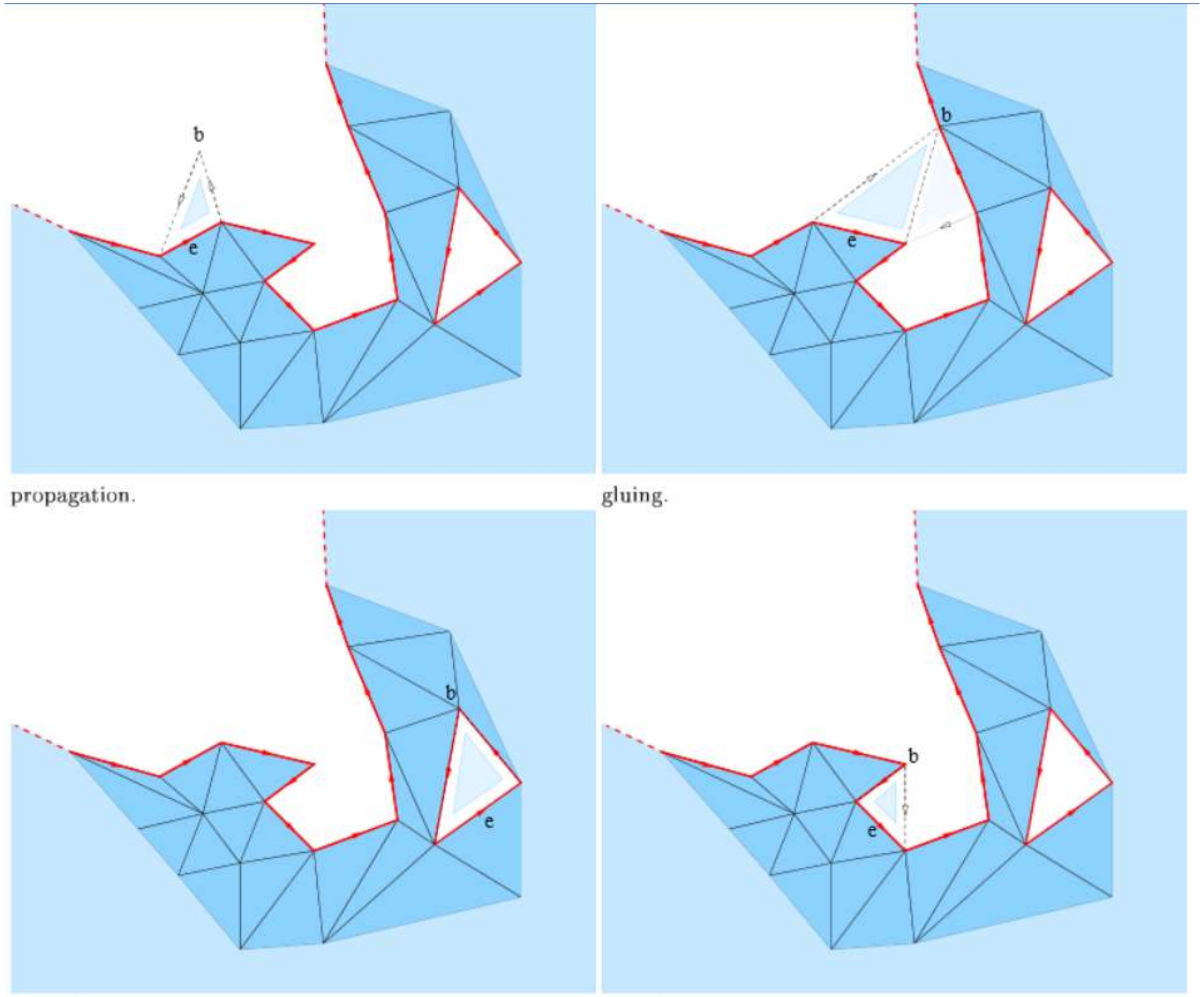
- Three methods
  - Advancing Front cgal.org
  - Poisson cgal.org
  - Scale Space cgal.org

- Common tool: Delaunay_3
-





Advancing Front Reconstruction

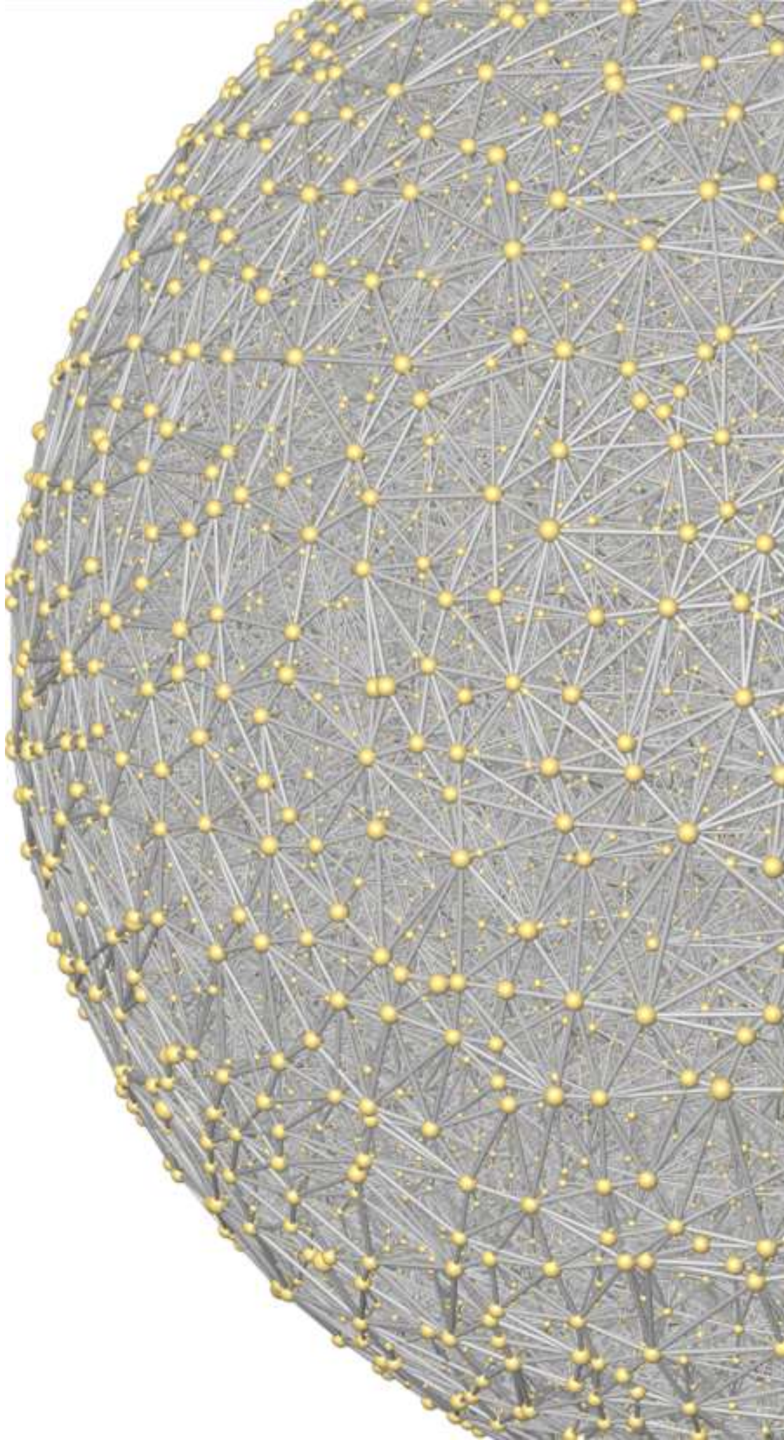


propagation.

gluing.

hole filling.

ear filling.



cgal.org

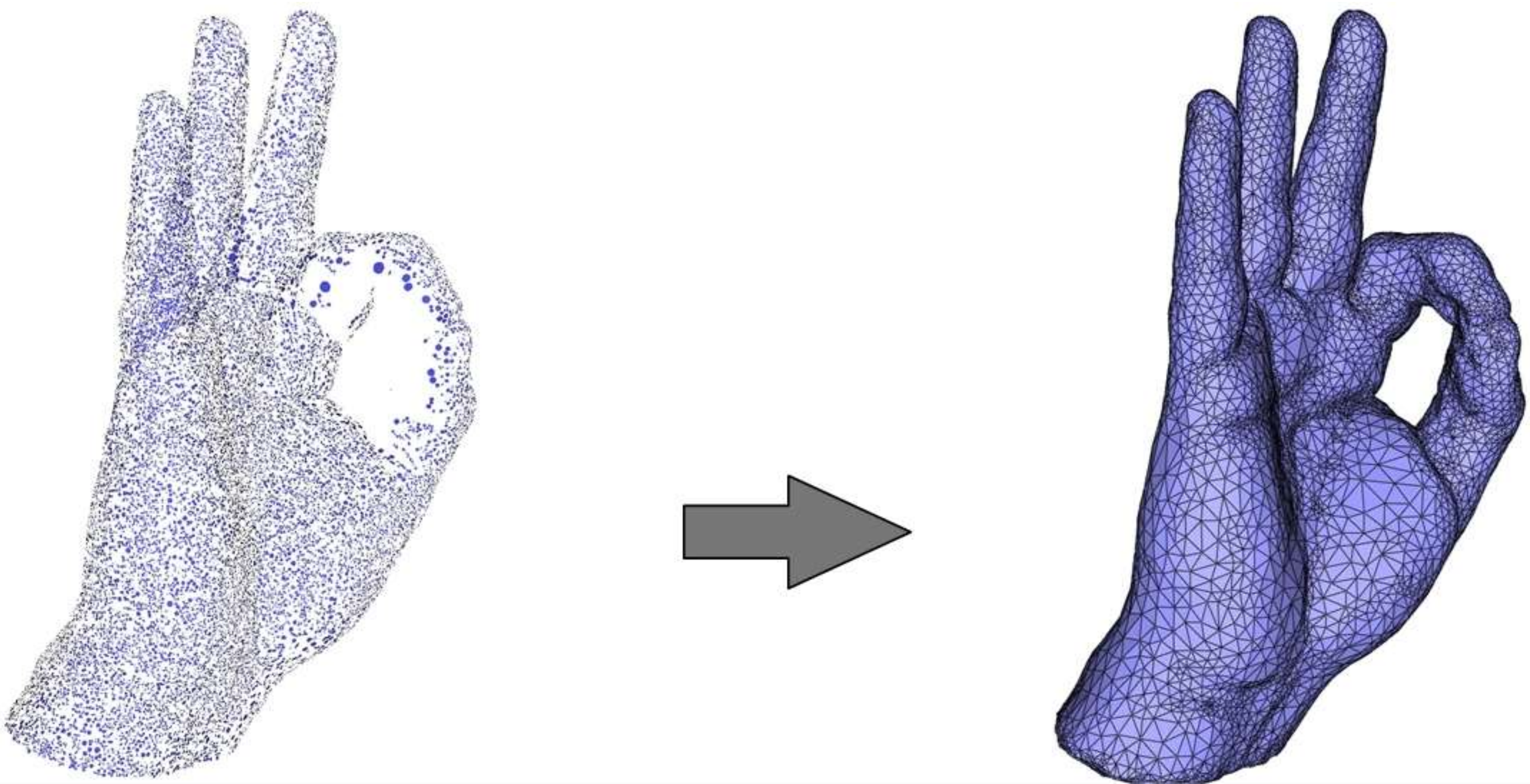
Poisson Surface Reconstruction [Kazhdan et al. 2006]

- Approximate implicit surface
- Replace octree by 3D tetrahedral mesh of ambient space
- Compute iso-surface with CGAL::Surface_mesher

Impossible d'afficher l'image.

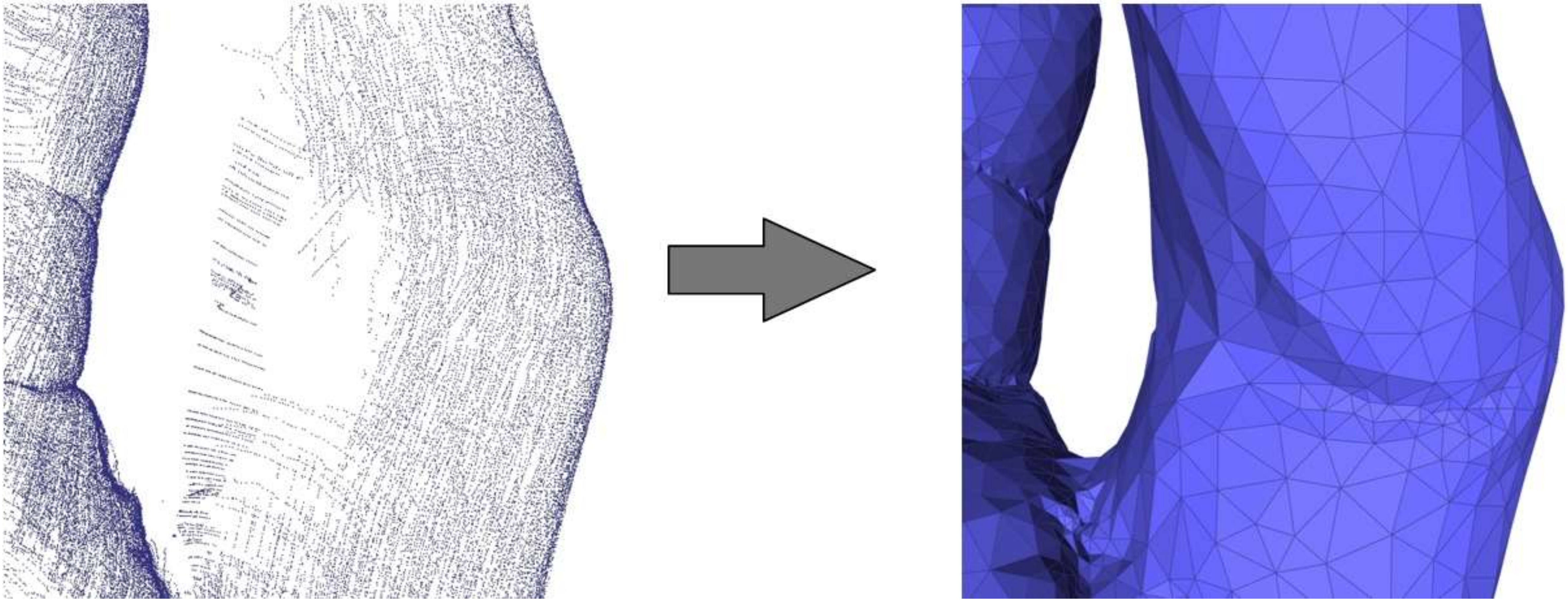
cgal.org

Poisson Surface Reconstruction



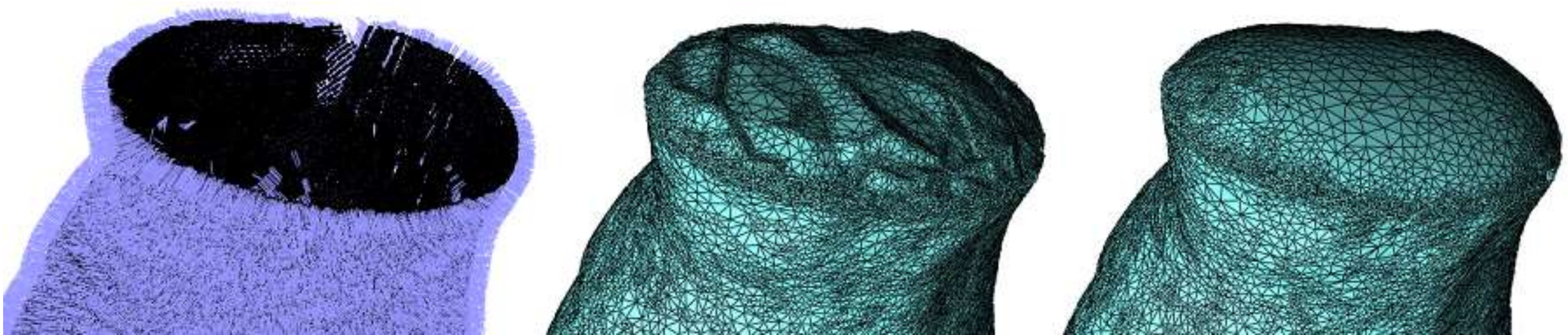
Poisson Surface Reconstruction

- Works well for uneven distribution of points



Poisson Surface Reconstruction

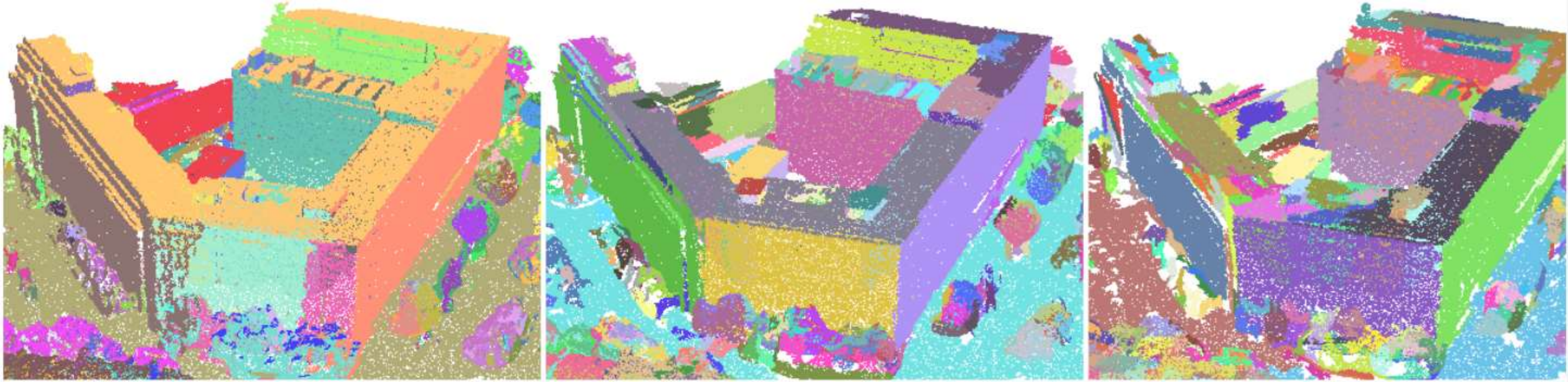
- Algorithm produces water tight surfaces
- For large holes use two-pass algorithm



Shape Detection

Efficient RANSAC – Main Parameters

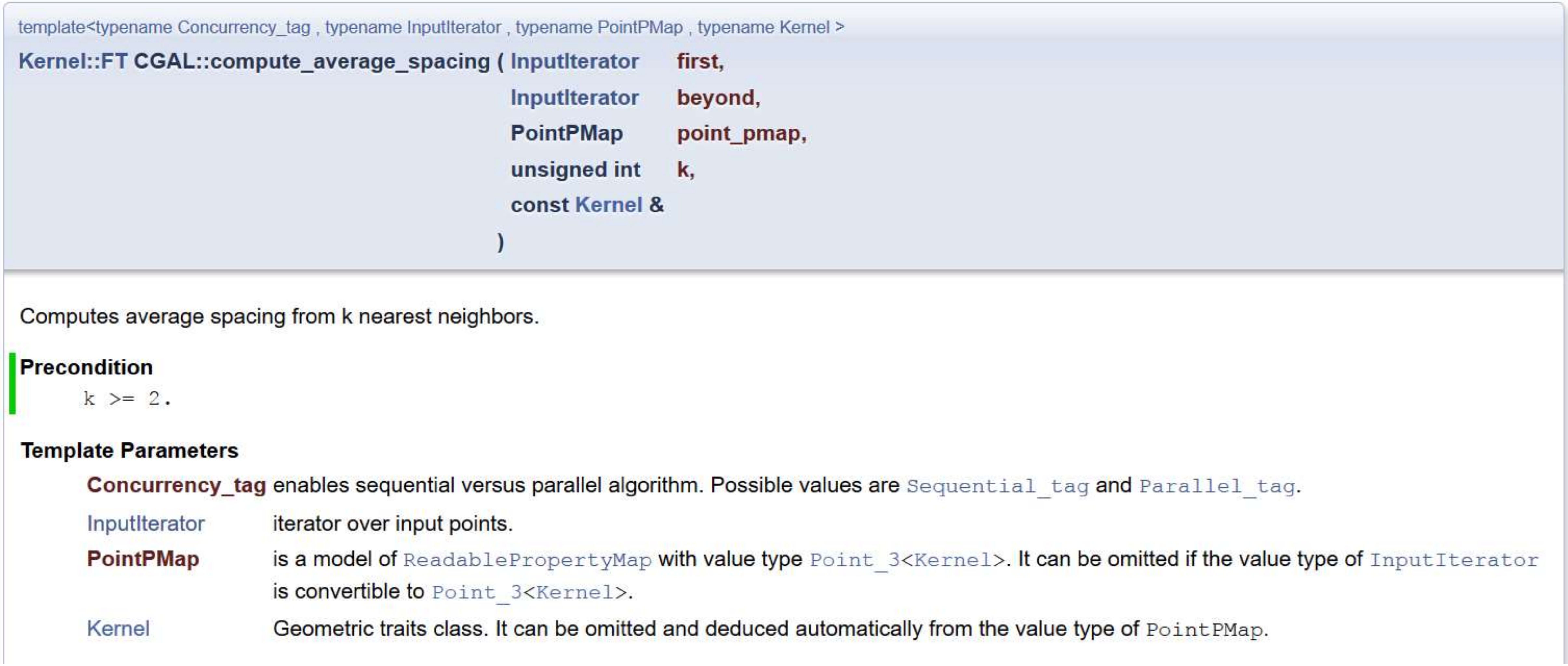
- Shape types : plane, cone, cylinder, sphere, torus
- Maximal distance



API

cgal.org

API



```
template<typename Concurrency_tag , typename InputIterator , typename PointPMap , typename Kernel >
Kernel::FT CGAL::compute_average_spacing ( InputIterator    first,
                                           InputIterator    beyond,
                                           PointPMap        point_pmap,
                                           unsigned int     k,
                                           const Kernel &
                                         )
```

Computes average spacing from k nearest neighbors.

Precondition

`k >= 2.`

Template Parameters

Concurrency_tag enables sequential versus parallel algorithm. Possible values are `Sequential_tag` and `Parallel_tag`.

InputIterator iterator over input points.

PointPMap is a model of `ReadablePropertyMap` with value type `Point_3<Kernel>`. It can be omitted if the value type of `InputIterator` is convertible to `Point_3<Kernel>`.

Kernel Geometric traits class. It can be omitted and deduced automatically from the value type of `PointPMap`.

Example: Just Points

```
typedef CGAL::Simple_cartesian<double>::Point_3 Point_3;
std::vector<Point_3> points;

double as = compute_average_spacing(points.begin(), points.end(),..);
```

Functions can operate on any range of objects :

```
template <class Iterator, class PointPropertyMap>
compute_average_spacing(Iterator first,
                        Iterator beyond,
                        PointPropertyMap point_pmap);
```

PointPropertyMap must map value_type of Iterator to Point_3

Example: Point and Vector in a Pair

```
typedef std::pair<Point_3,Vector_3> Pwn;
Std::vector<Pwn> points;

double as = compute_average_spacing(points.begin(),
                                    points.end(),
                                    CGAL::First_of_pair_property_map<Pwn>());
```

Example: Point and Vector in a Pair

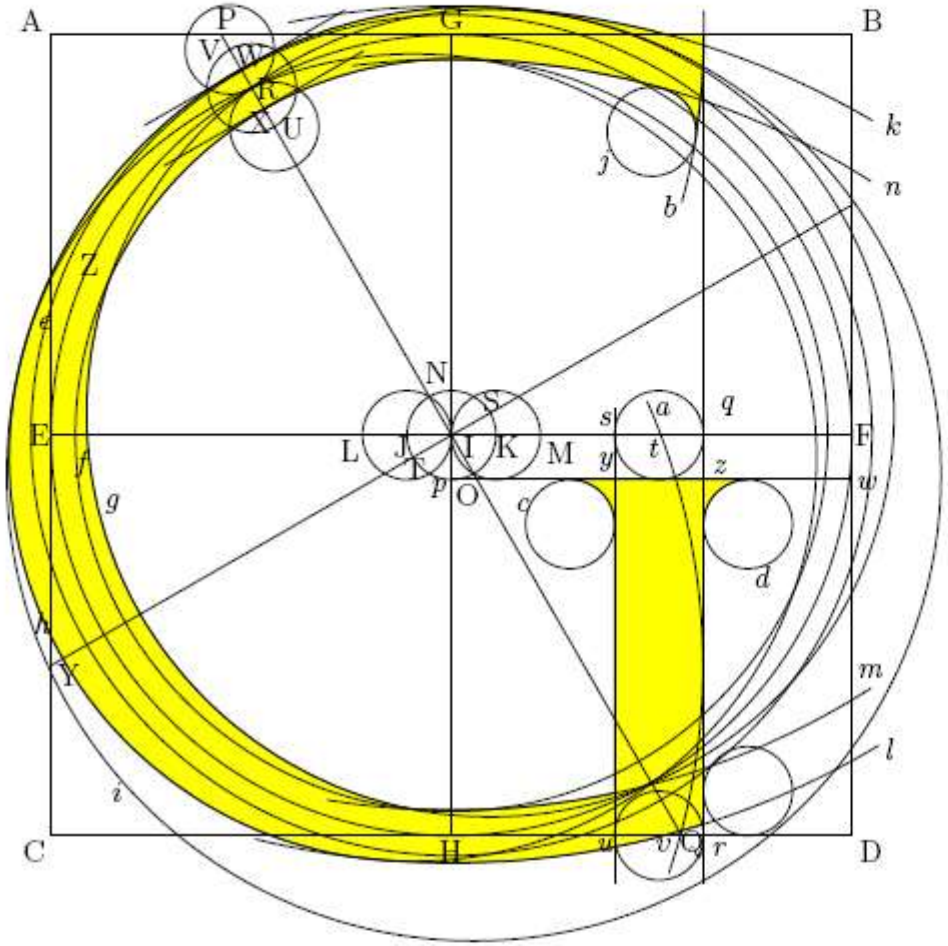
```
typedef std::pair<Point_3,Vector_3> Pwn;
Std::vector<Pwn> points;

estimate_normals(points.begin(),
                 points.end(),
                 CGAL::First_of_pair_property_map<Pwn>(),
                 CGAL::Second_of_pair_property_map<Pwn>());
```

Example: Writing Vectors into a Hash Map

```
std::vector<Point_3> points;
std::unordered_map<Point_3,Vector_3> pvm;


estimate_normals(points.begin(),
                 points.end(),
                 CGAL::Identity_property_map<Point_3>(),
                 boost::make_associative_property_map(pvm));
```

A property map must be light weight as it gets copied

`std::unordered_map` is hence **not** a property map



Polygon Mesh Processing

Background

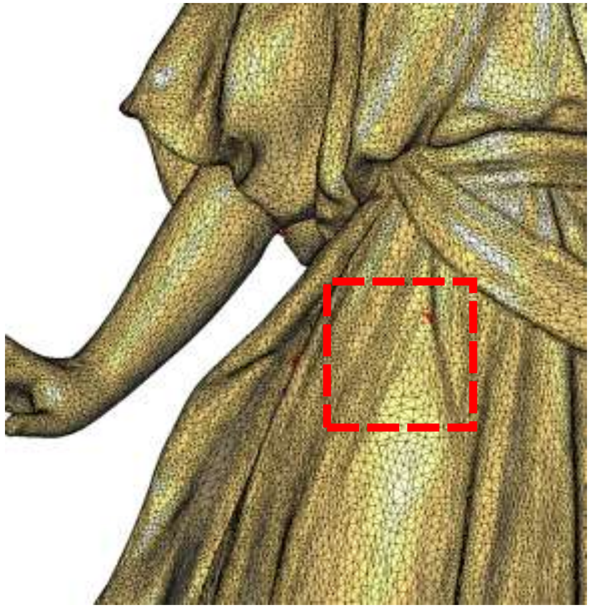
cgal.org

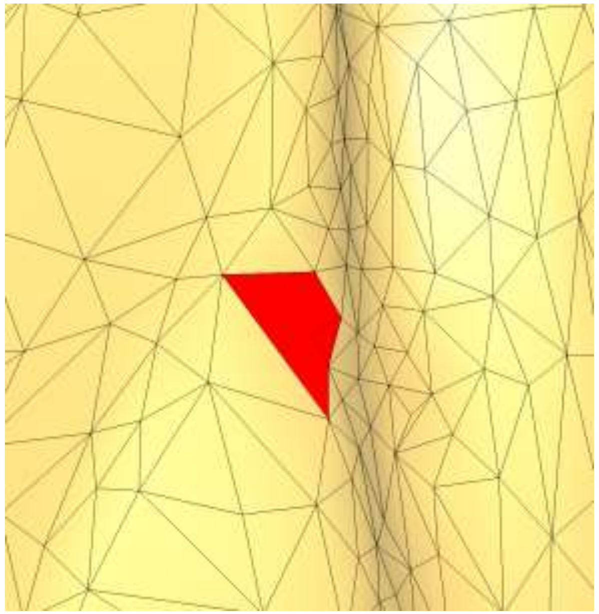
Intersection Detection

Based on `CGAL::box_intersection_d`

Applies user-defined callback on all intersecting pairs of boxes.

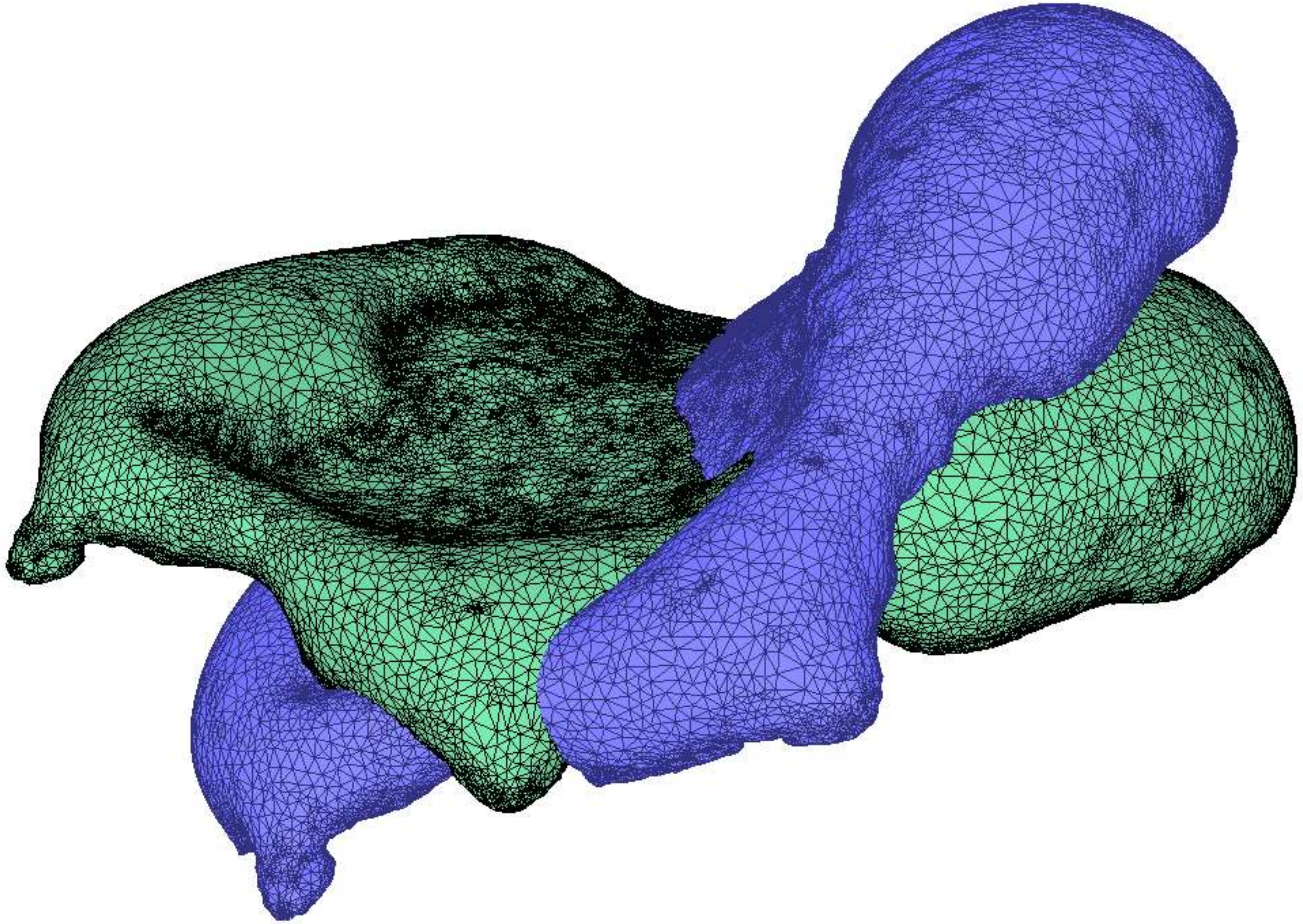
Generic programming : **Box** is a template argument, and can contain objects of any type





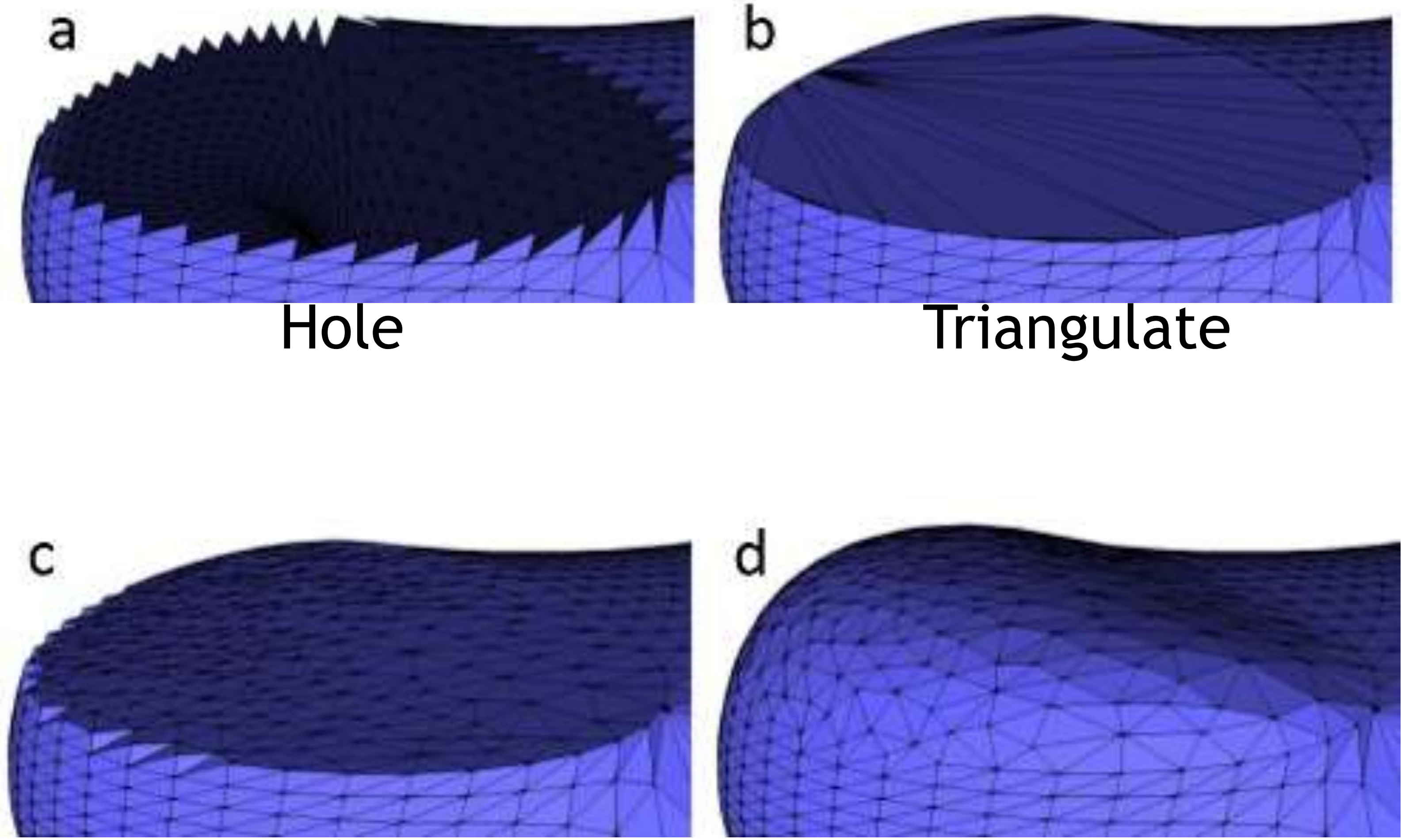
Intersection Test

0.2s for two surfaces with 100k triangles



cgal.org

Hole filling [Liepa 2003], [Zou et al. 2013]



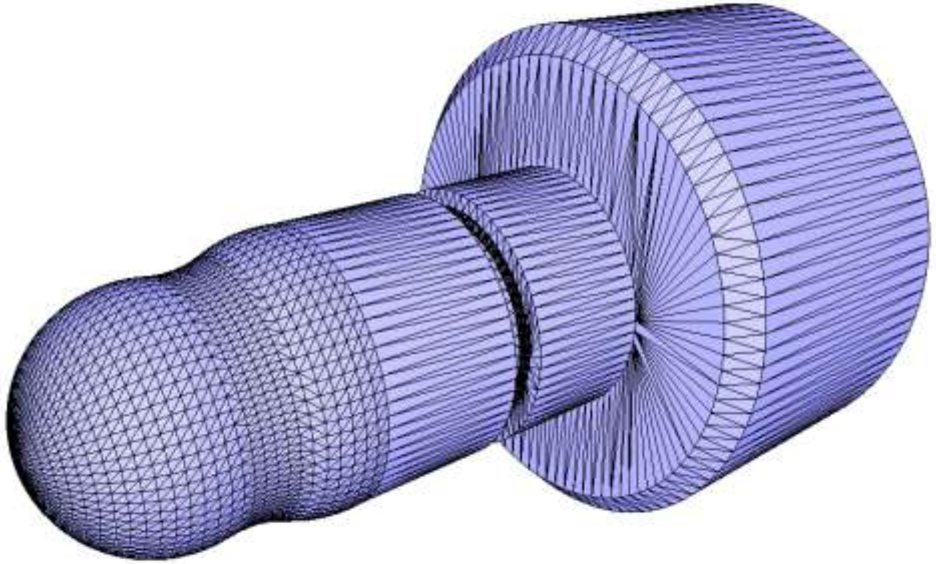
Hole

Triangulate

Refine

Fair

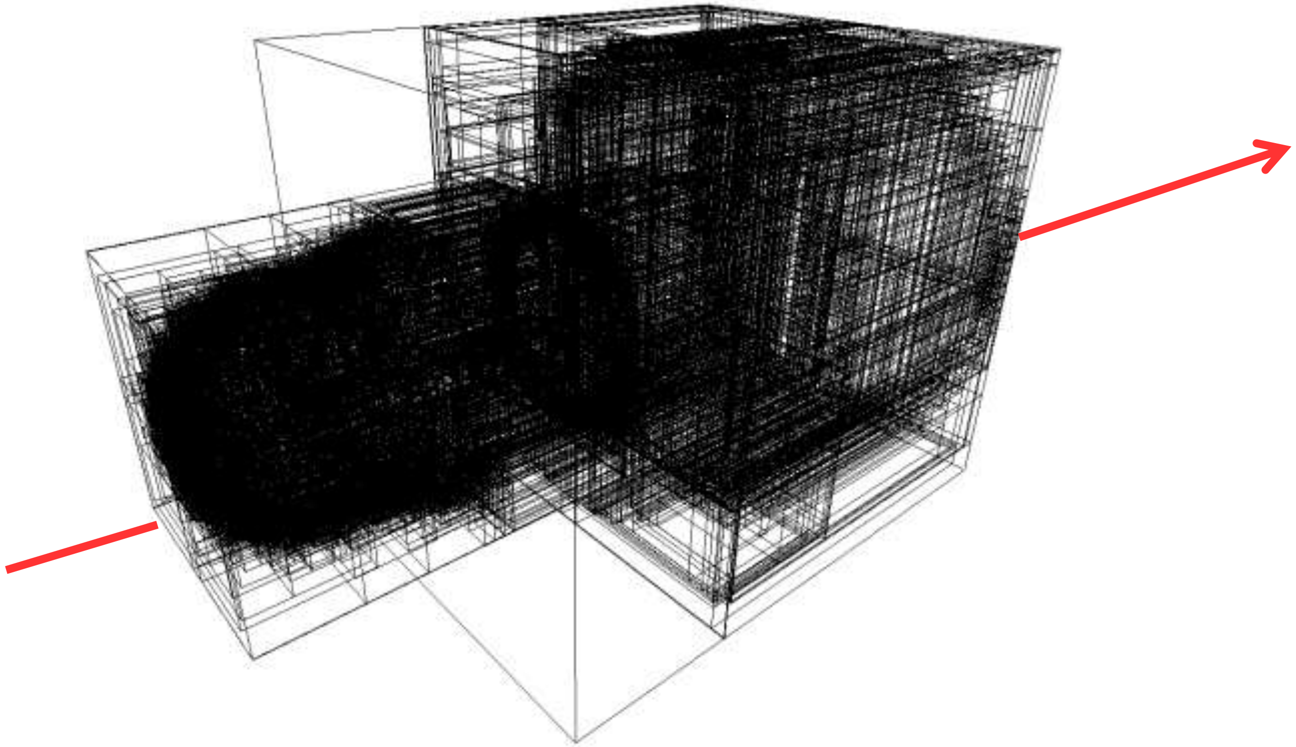
Distances and Intersections



Based on CGAL::AABB_tree

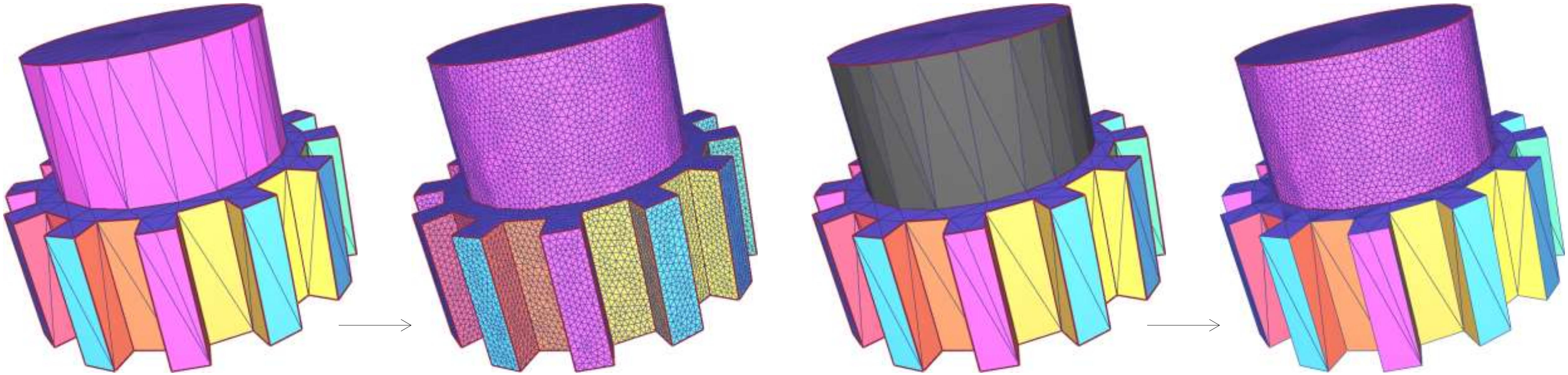
Generic programming :

Box is a template argument, and can contain objects of any type



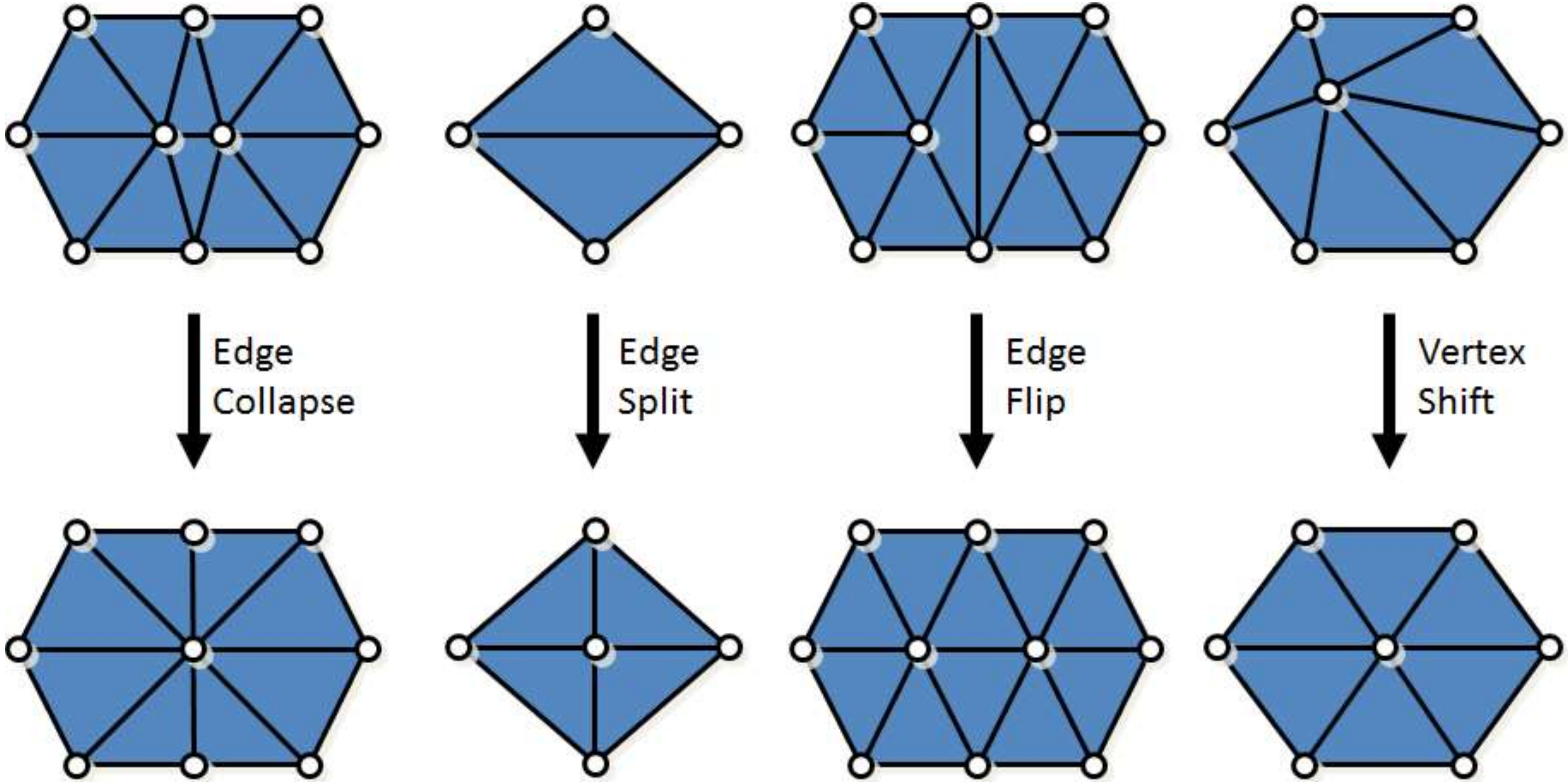
cgal.org

Isotropic Remeshing [Botsch-Kobbelt 2004]



Feature Preserving / Selection

Isotropic Remeshing

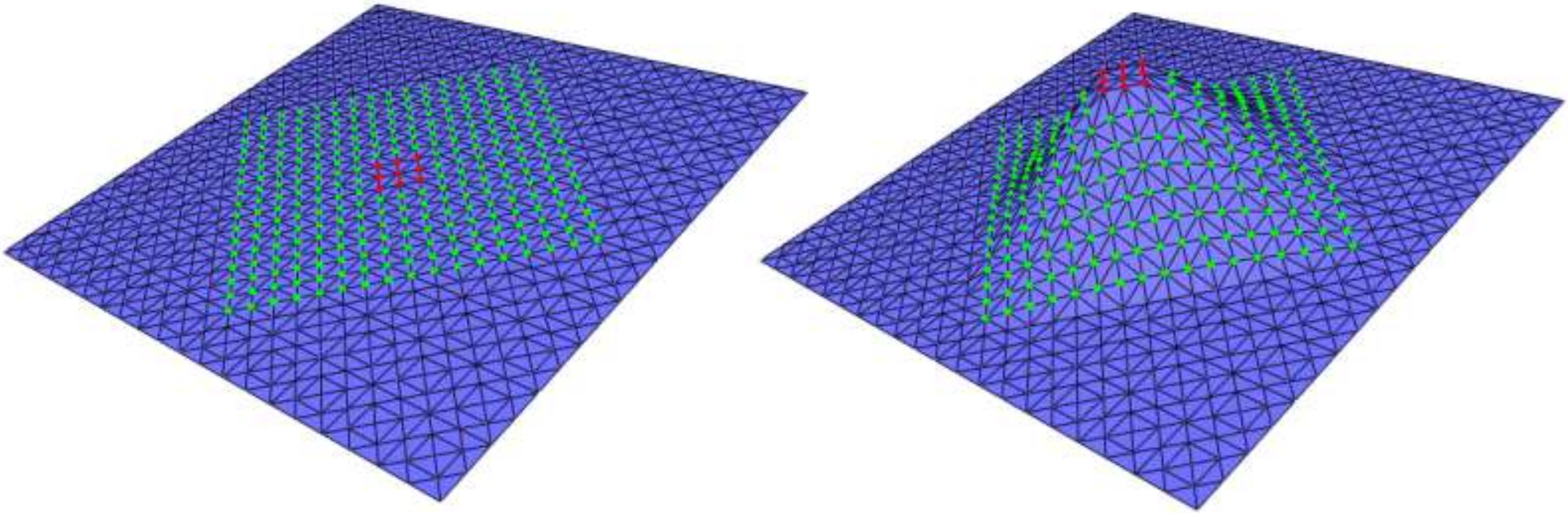


Local remeshing operators

cgal.org

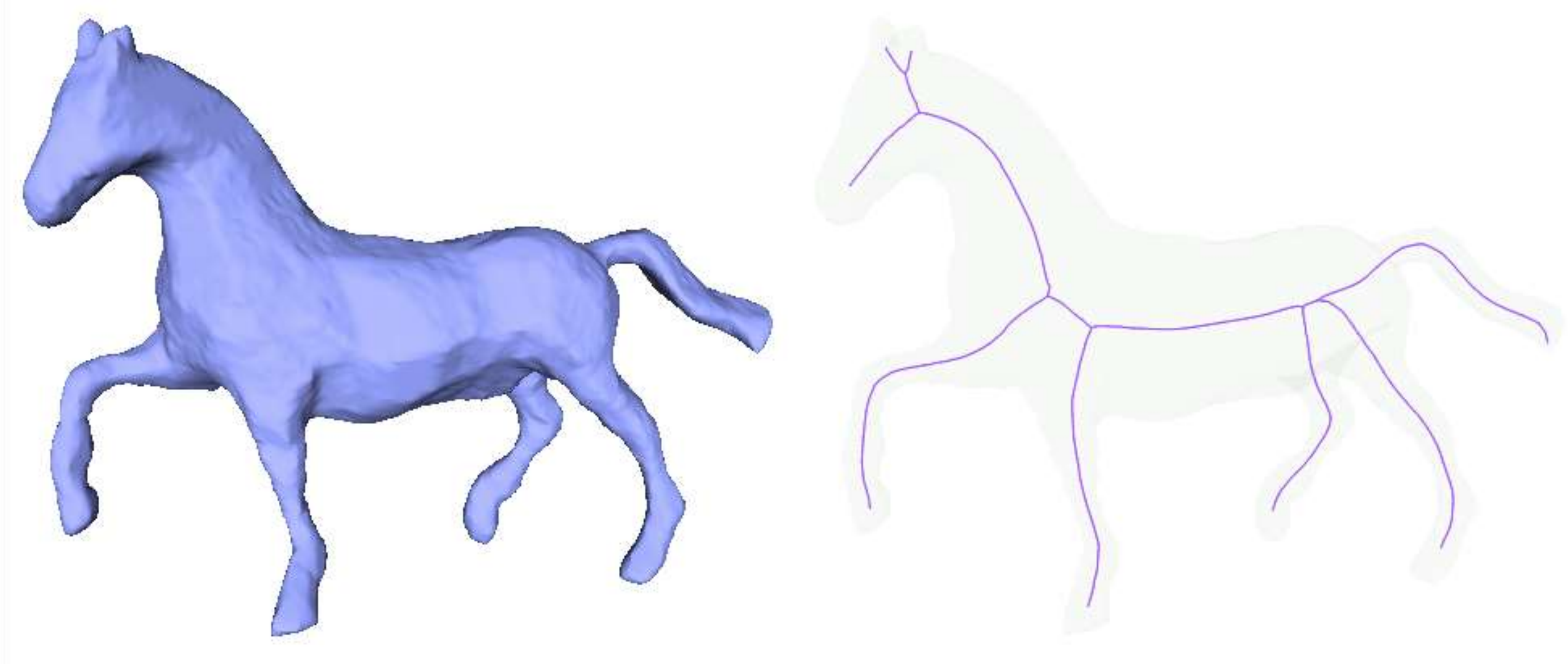
Deformation [Sorkine-Alexa 2007]

As Rigid as Possible ("ARAP")



Skeletonization [Tagliasacchi et al. 2012]

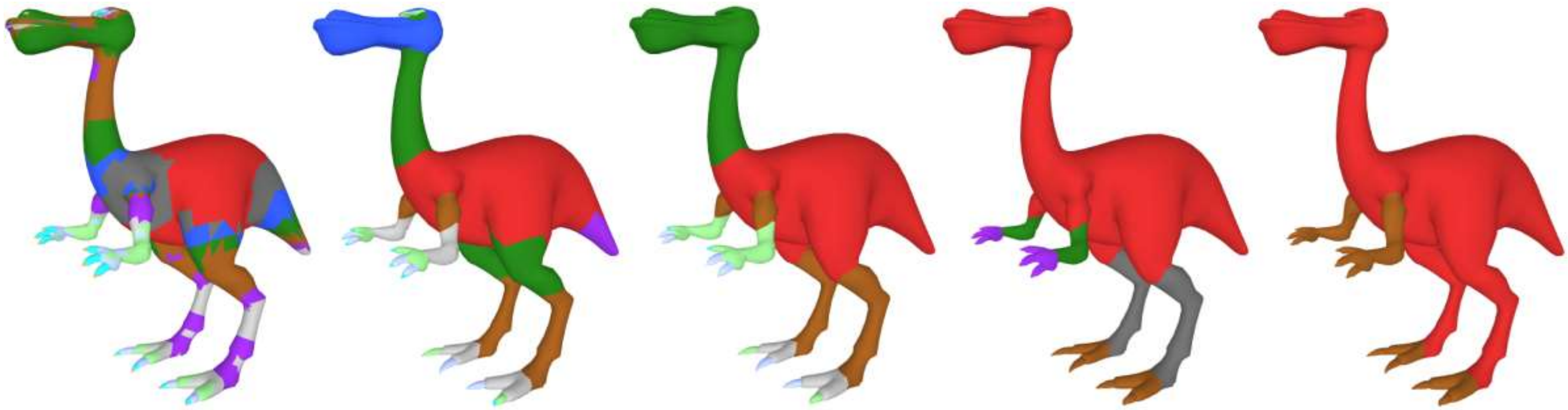
Mean Curvature Flow skeletonization



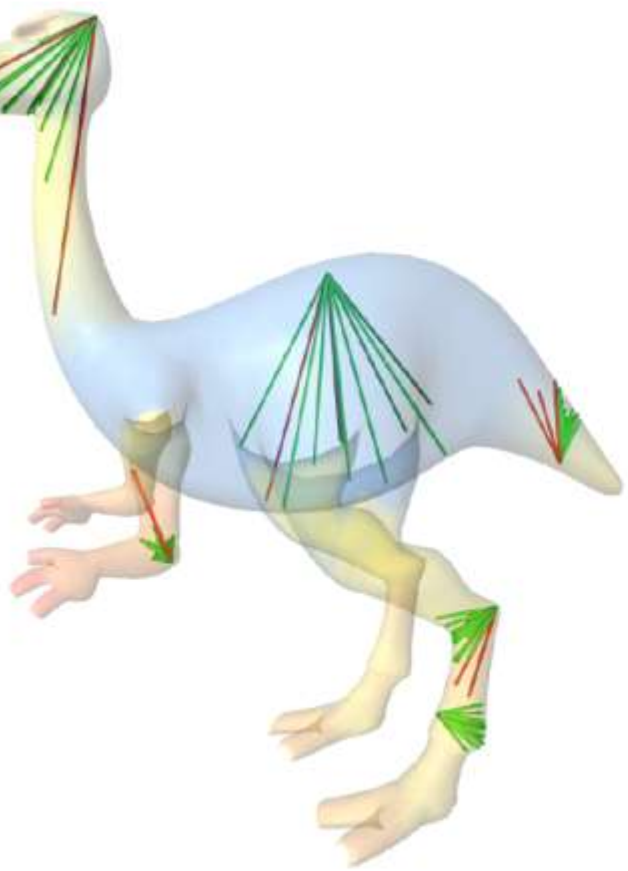
cgal.org

Segmentation [Shapira et al. 2008]

- Segment surface into k patches
- Based on «shape diameter» estimate



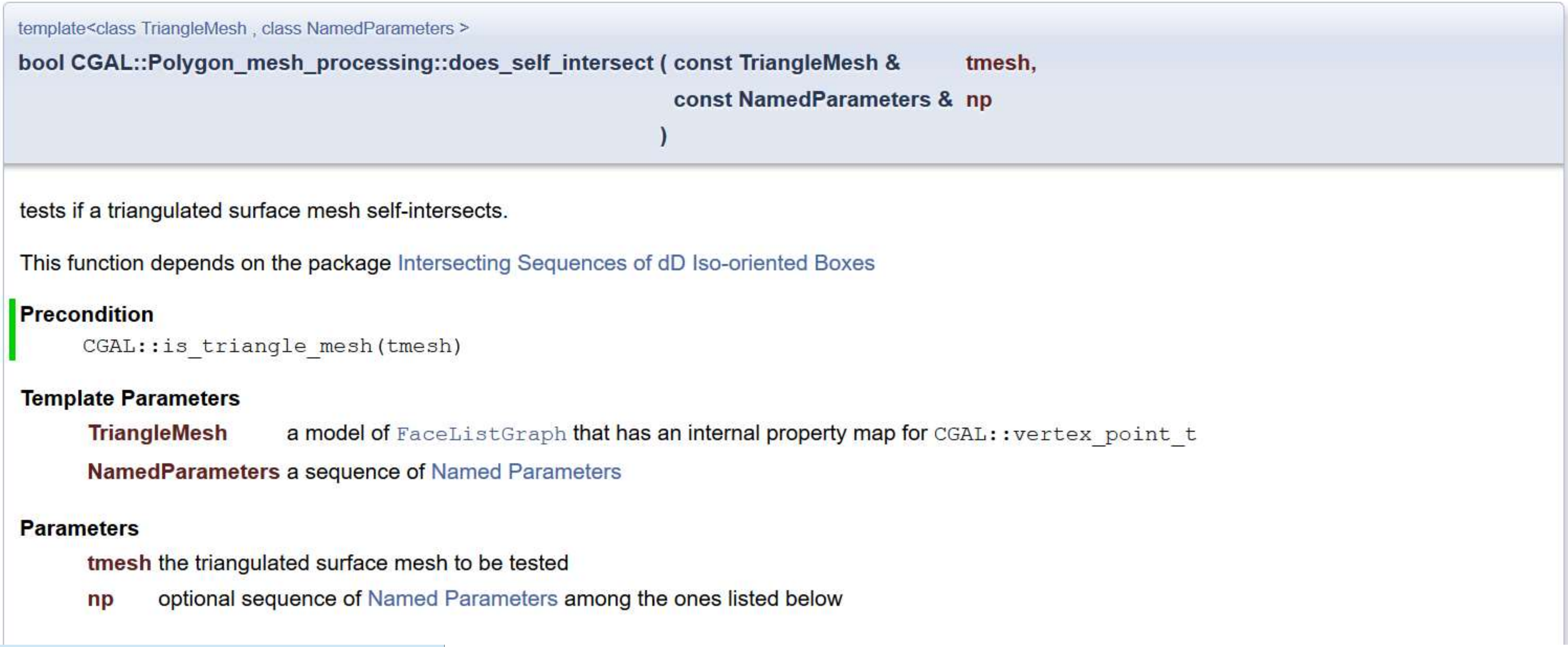
Segmentation



Shape diameter function

API

API



template<class TriangleMesh , class NamedParameters >

```
bool CGAL::Polygon_mesh_processing::does_self_intersect ( const TriangleMesh &       tmesh,
                                                          const NamedParameters &  np
                                                        )
```

tests if a triangulated surface mesh self-intersects.

This function depends on the package Intersecting Sequences of dD Iso-oriented Boxes

Precondition

`CGAL::is_triangle_mesh(tmesh)`

Template Parameters

TriangleMesh	a model of `FaceListGraph` that has an internal property map for `CGAL::vertex_point_t`
NamedParameters	a sequence of Named Parameters

Parameters

tmesh	the triangulated surface mesh to be tested
np	optional sequence of Named Parameters among the ones listed below

Example : Using a CGAL Mesh

```
typedef CGAL::Exact_predicates_inexact_constructions_kernel::Point_3 Point;

typdef CGAL::Surface_mesh<Point> Mesh;

int main()
{
  Mesh mesh;
  Std::ifstream in(“mesh.off”);
  in >> mesh;

  if( ! CGAL::Polygon_mesh_processing::does_self_intersect(mesh))
    do_something(mesh);
}
```

Example: Using User Defined Mesh

```
typedef CGAL::Exact_predicates_inexact_constructions_kernel::Point_3 Point;

typdef CGAL::Surface_mesh<Point> Mesh;

int main()
{
  MyLab::HalfedgeDS myhds;
  Mesh mesh = convert_to_CGAL(myhds);

  if(! CGAL::Polygon_mesh_processing::does_self_intersect(mesh))
     do_something(myhds);
}
```

Goal: Avoid the conversion by making

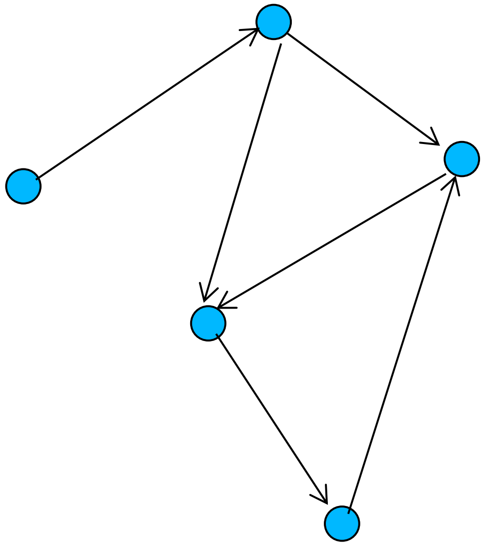
`MyLab::HalfedgeDS` a model of `TriangleMesh`

CGAL and the Boost Graph Library

BGL - Boost Graph Library

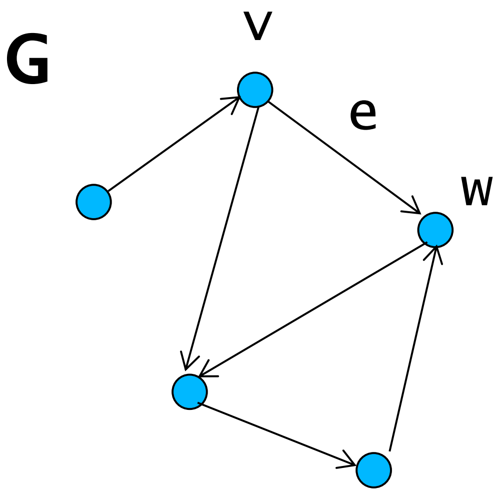
- Rich collection of graph algorithms
  shortest paths, minimum spanning tree, flow, ...
- BGL design
  - separates data structure from algorithm
  - links them through a thin glue layer
- BGL and CGAL
  - we provide glue layer for CGAL
  - Extension: we order edges incident to a vertex
    inducing the notion of faces

BGL Glue Layer : Traits Class



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

BGL Glue Layer : Free Functions



```
vertex_descriptor v, w;
edge_descriptor e;

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


std::pair<vertex_iterator, vertex_iterator> ipair;

ipair = vertices(G);
```

CGAL::Surface_mesh as Graph

CGAL provides partial specializations of boost::graph_traits<..>

```
template <typename P>
struct boost::graph_traits<CGAL::Surface_mesh<P> > {

  typedef   Surface_mesh<P>::Vertex_index    vertex_descriptor;


  typedef   Surface_mesh<P>::Vertex_iterator> vertex_iterator;


};
```

CGAL::Surface_mesh as Graph

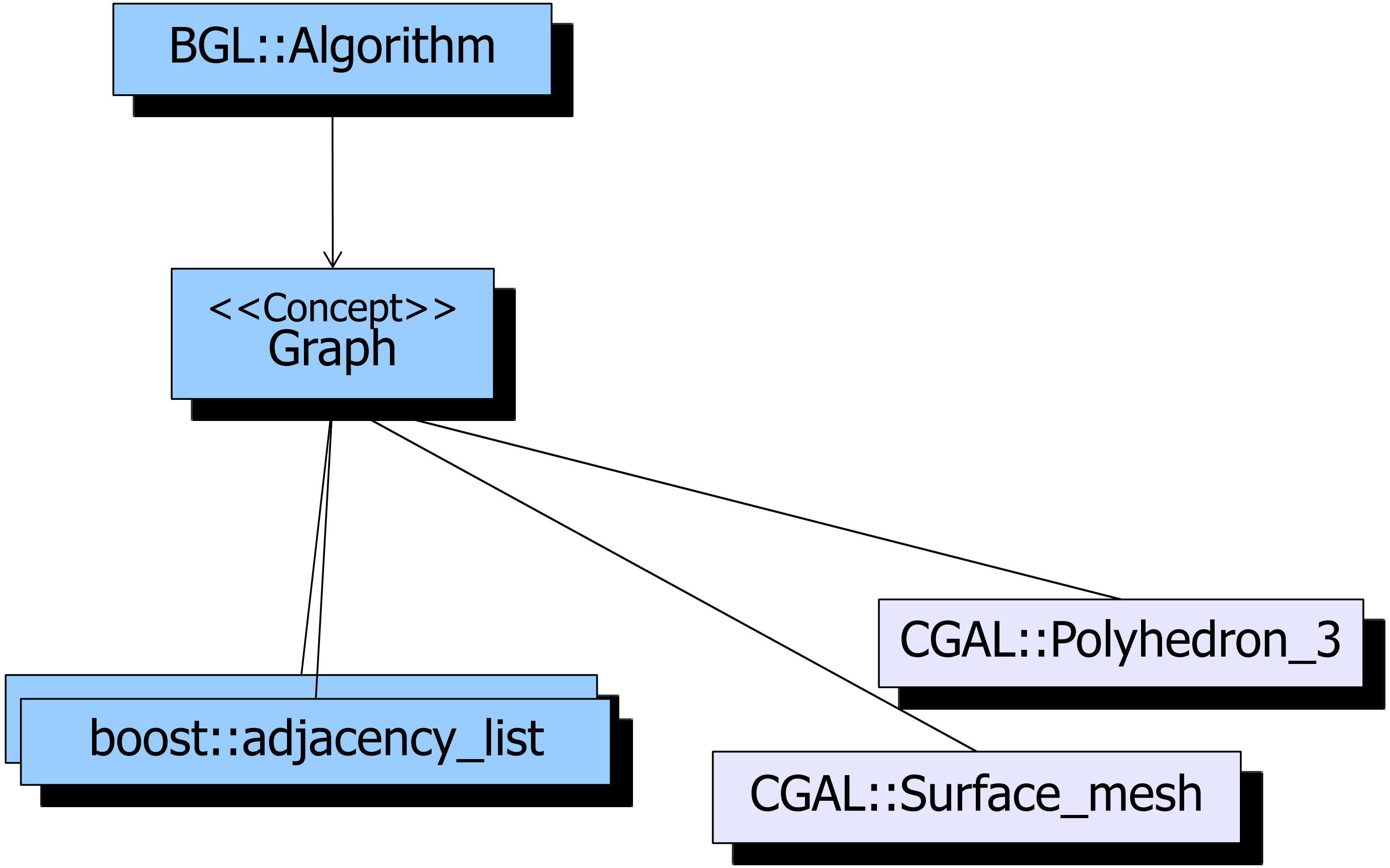
```
template <typename P>
vertex_descriptor
target(edge_descriptor e,
       Surface_mesh<P>& graph)
{
  return graph.target(e) ;
}
```

Users can run

```
boost::kruskal_mst(sm);
```

Courtesy: P.Schroeder, Caltech

From A BGL Glue Layer for CGAL



BGL::Algorithm

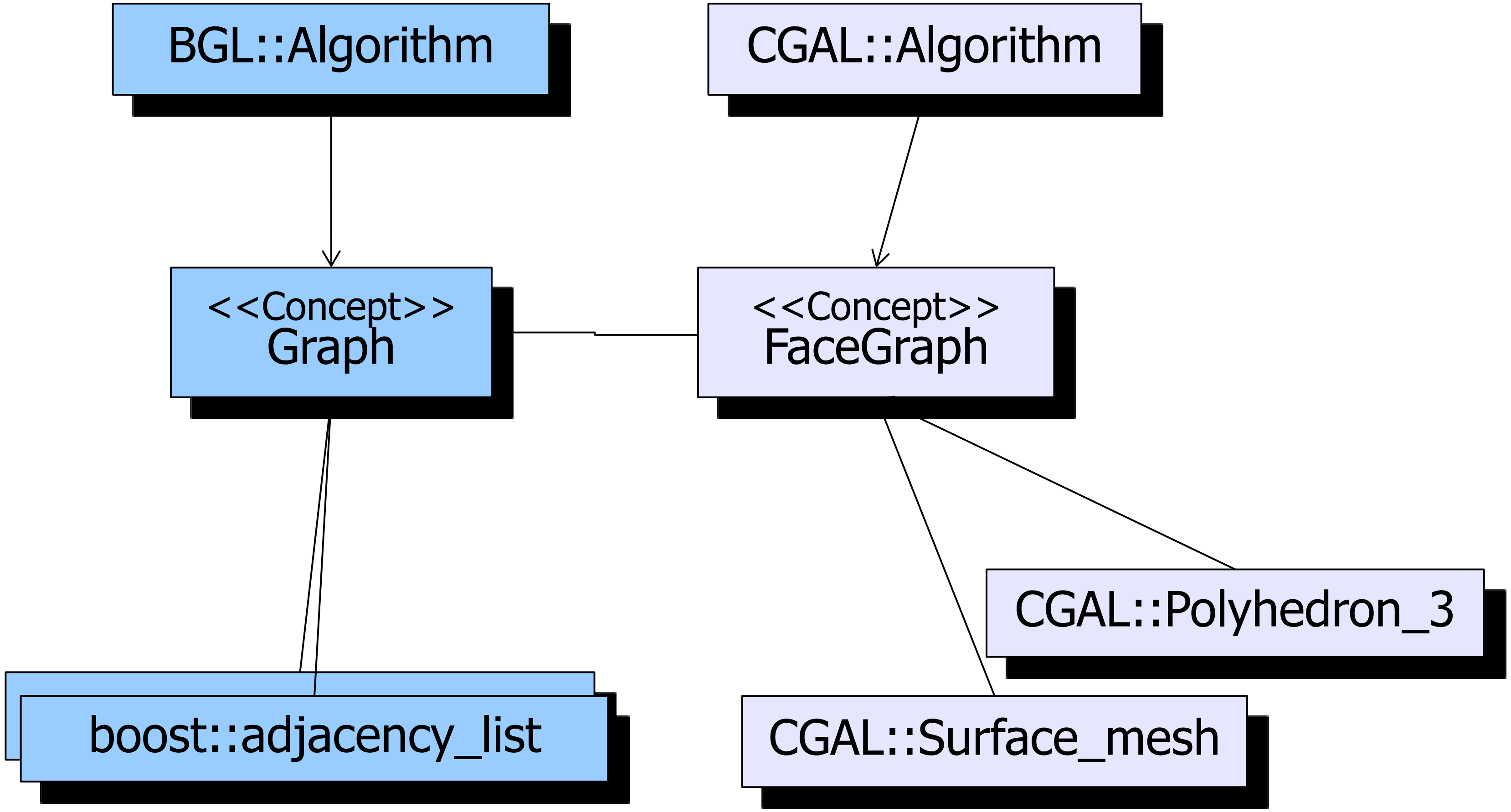
<<Concept>>
Graph

CGAL::Polyhedron_3

boost::adjacency_list

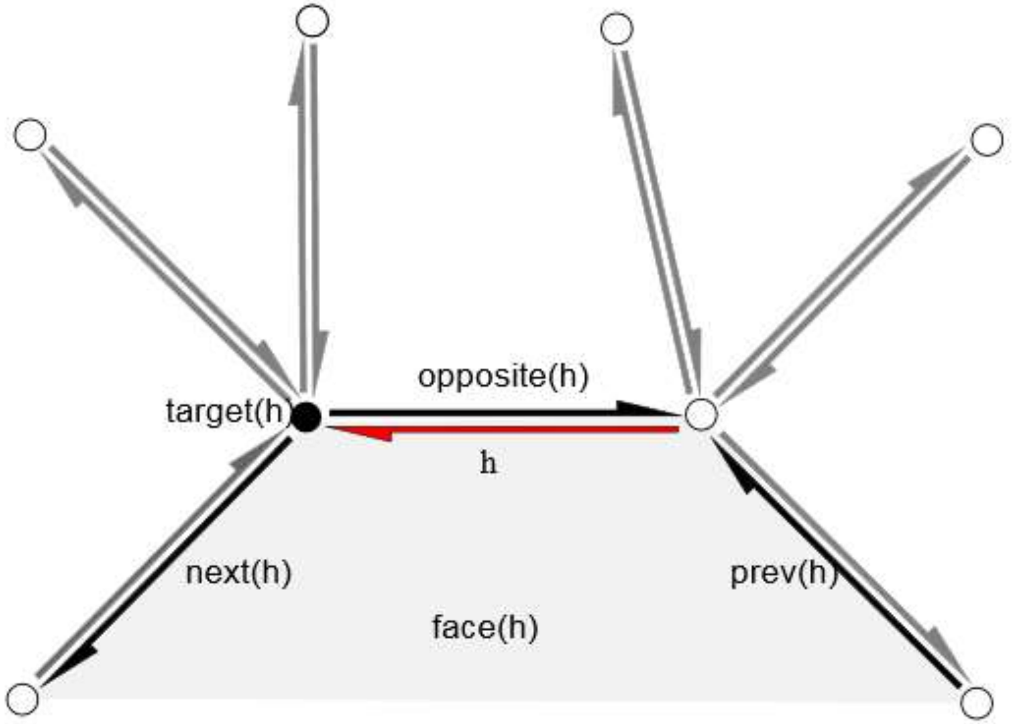
CGAL::Surface_mesh

To BGL Style CGAL Algorithms



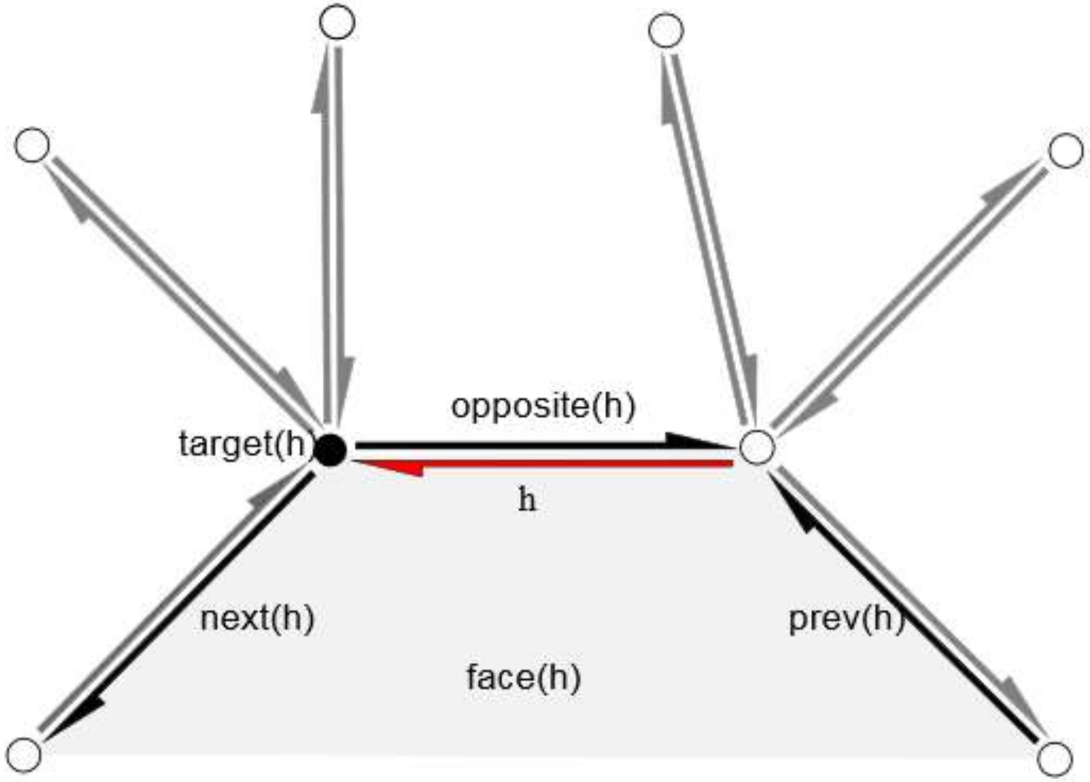
Extension of the Traits Class

```
template <typename FaceGraph >
struct boost::graph_traits {
    typedef ... vertex_descriptor;
    typedef ... edge_descriptor;
    typedef ... halfedge_descriptor;
    typedef ... face_descriptor;
};
```



Extension of the Free Functions

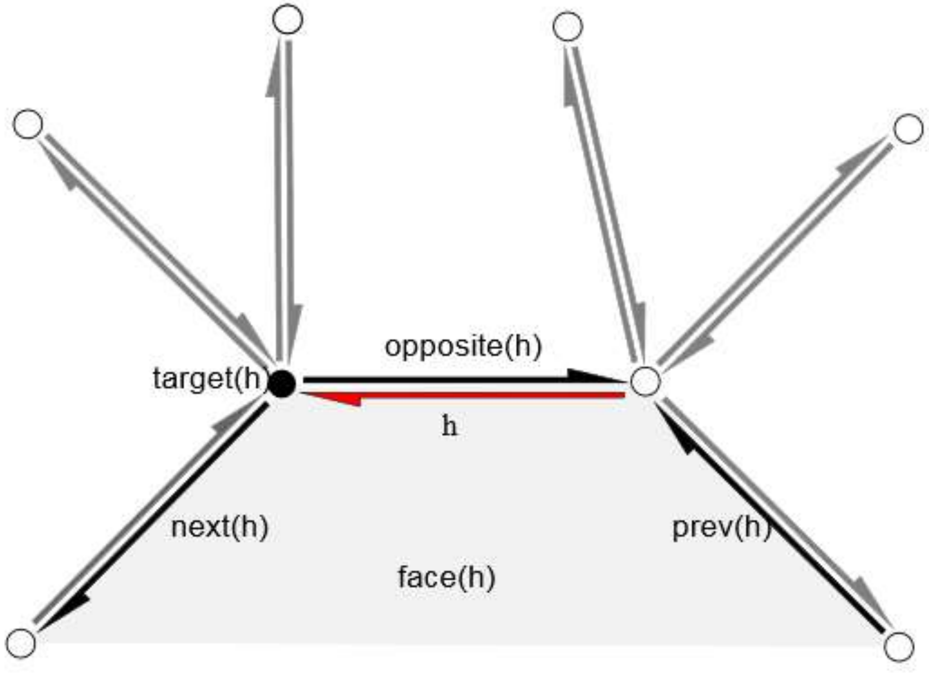
```
vertex_descriptor v, w;
edge_descriptor e;
halfedge_descriptor h,hn,hopp;
face_descriptor f;

hopp = opposite(h,G);
hn = next(h,G);

h = halfedge(e,G);
e = edge(h,G);

f = face(h,G);
h = halfedge(f,G);
```



Generic Iterators

`CGAL::Halfedge_around_face_iterator`

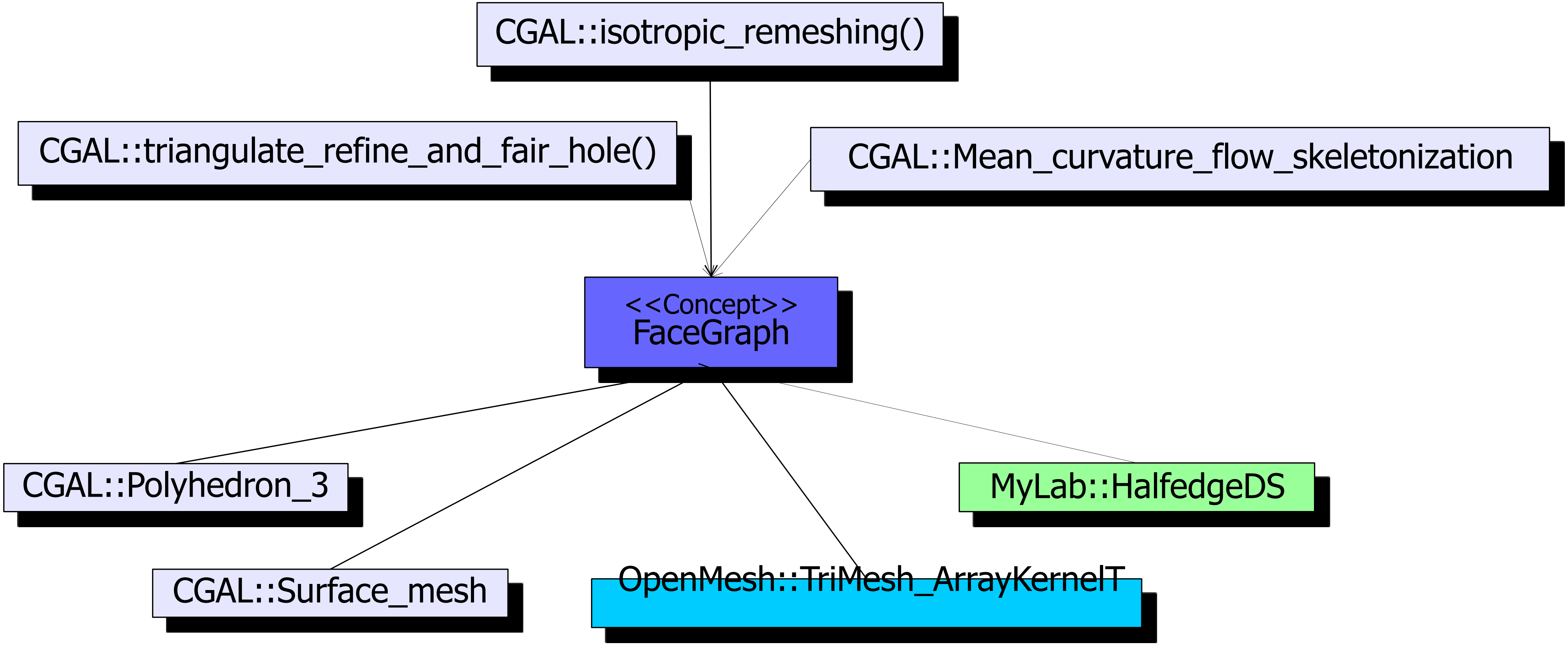
- `stores h and G* g`
- `calls h = next(h,*g) on operator++()`

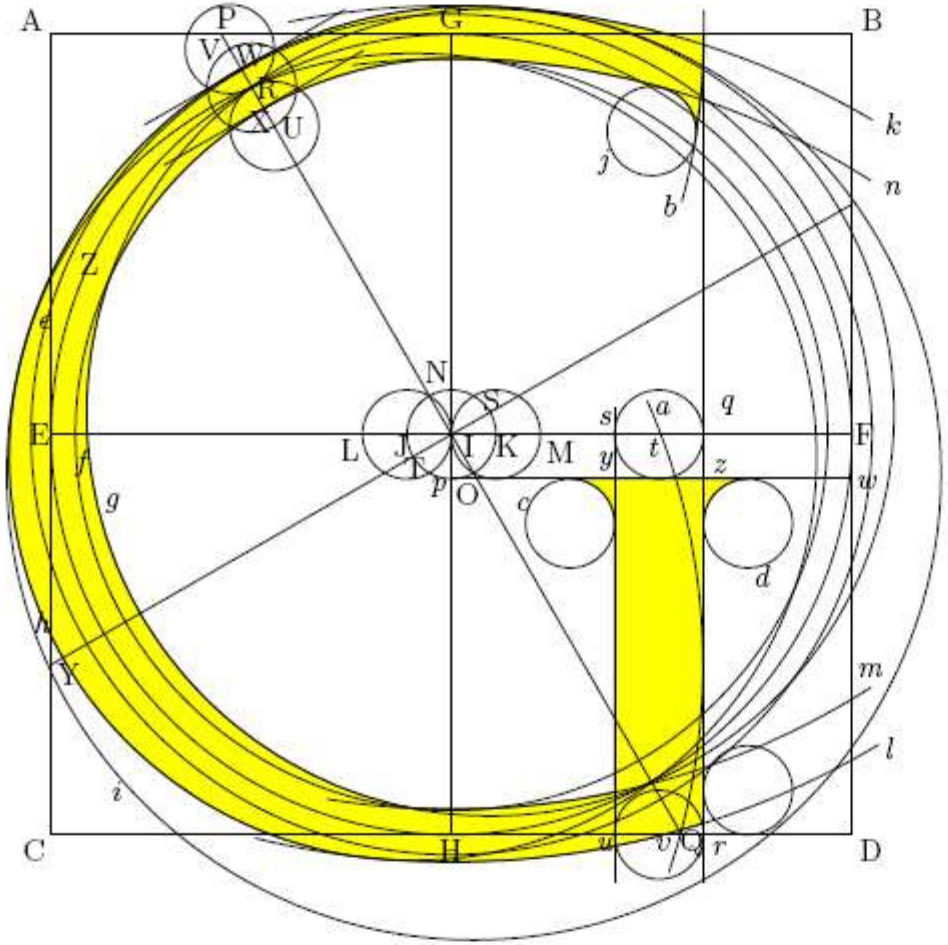


returns an `Iterator_range` ~= begin/end pair

```
for(auto h : halfedges_around_face(h,g)){
  ...
}
```

Generic Polygon Mesh Processing

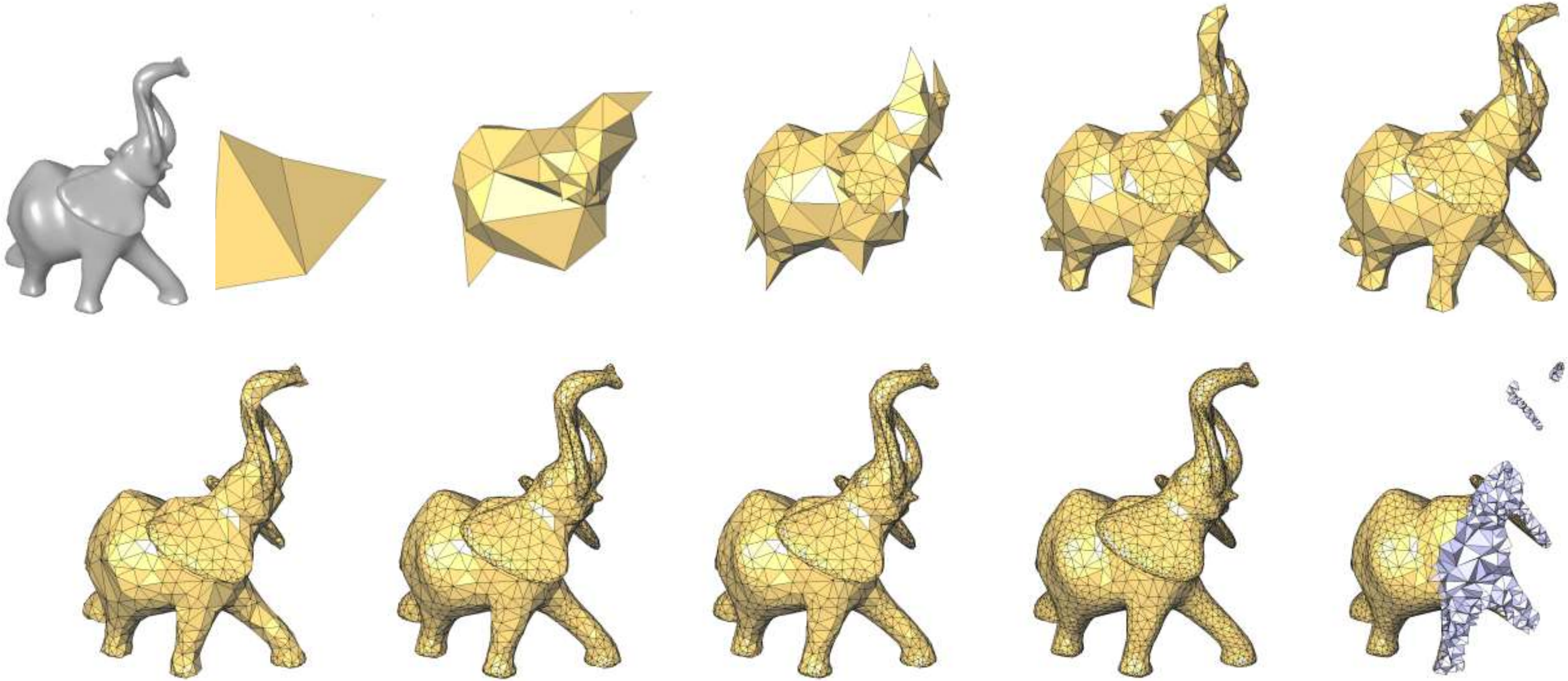

CGAL::isotropic_remeshing()
CGAL::triangulate_refine_and_fair_hole()
CGAL::Mean_curvature_flow_skeletonization
<<Concept>>
FaceGraph
CGAL::Polyhedron_3
MyLab::HalfedgeDS
CGAL::Surface_mesh
OpenMesh::TriMesh_ArrayKernelT



Mesh Generation

Background

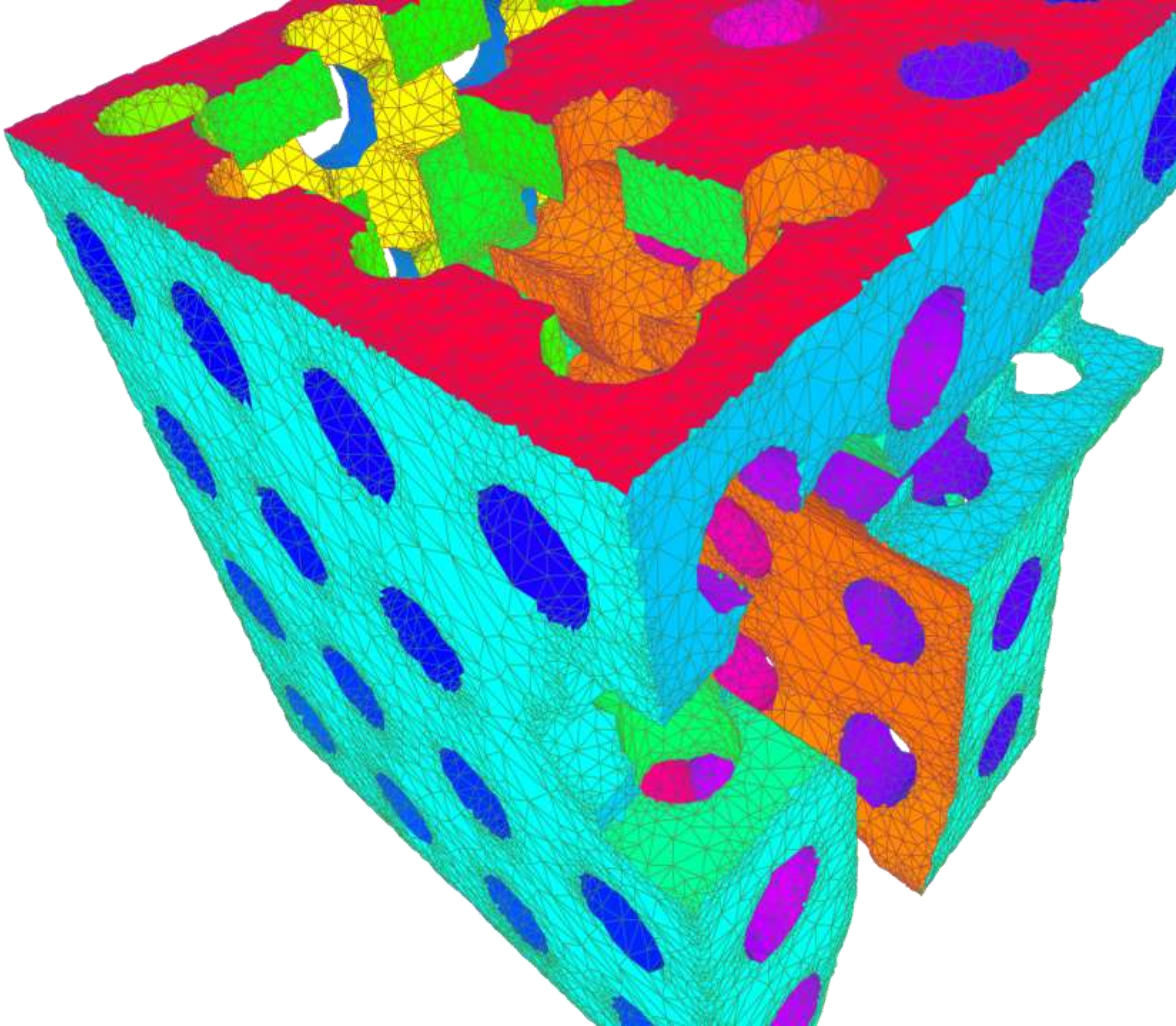
Delaunay Mesh Generation



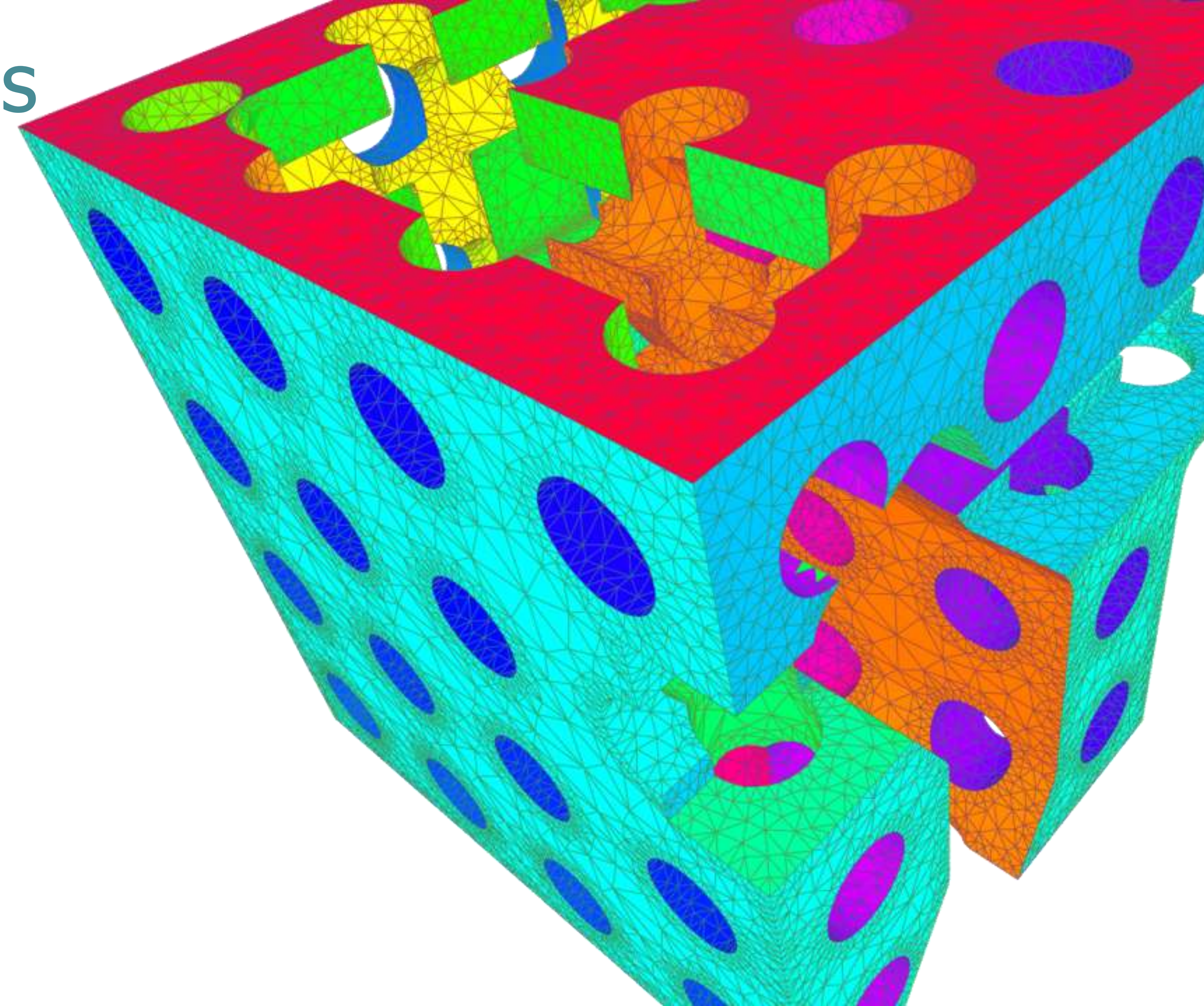
Delaunay Filtering and Refinement

```
repeat
{
pick worst facet f
insert   $\text{dual}(f) \cap S$   in Delaunay triangulation
update Delaunay triangulation restricted to S
}
until all facets are good
```

Sharp Features



Sharp Features



Multi Domain Volume Meshes

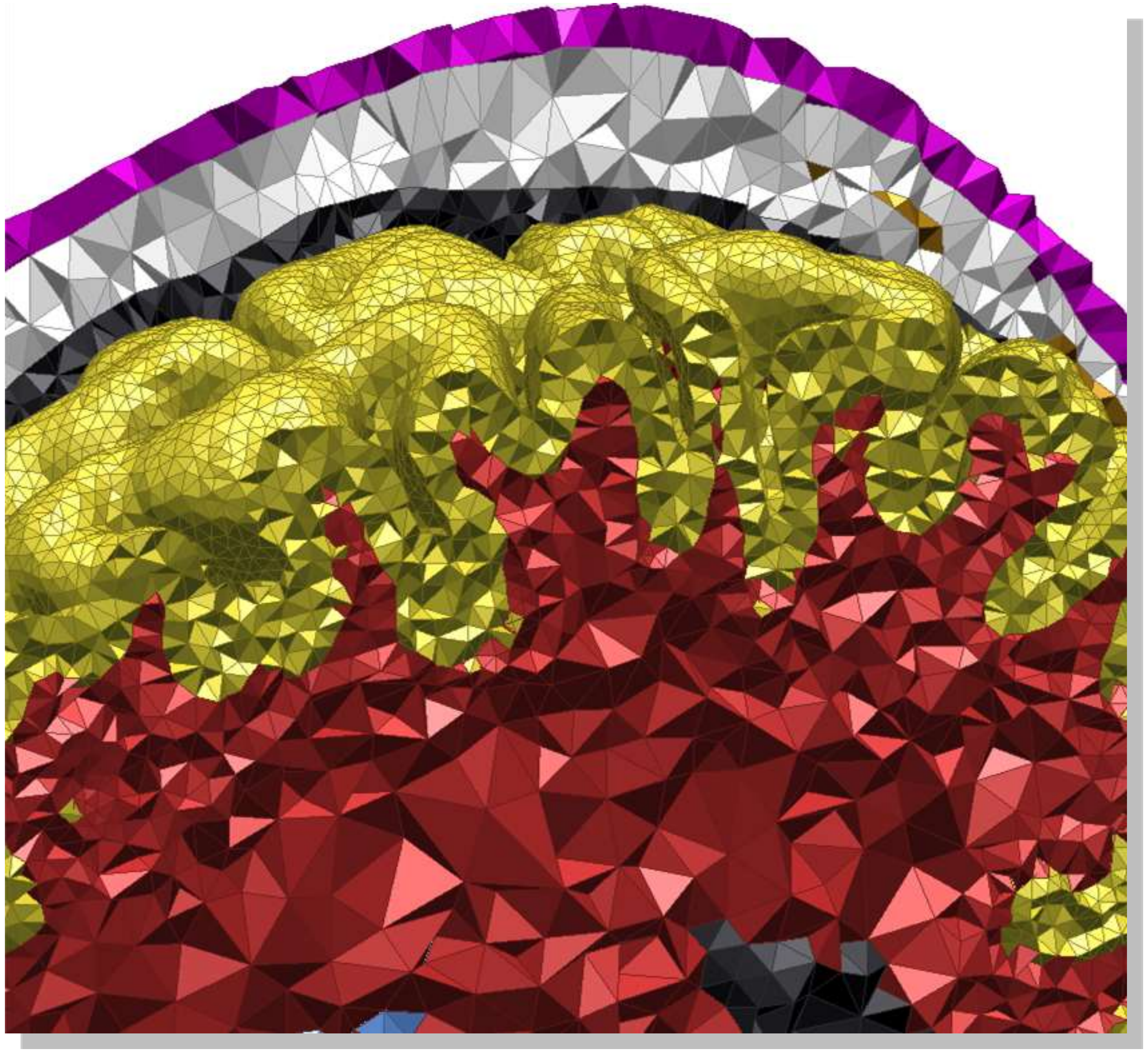
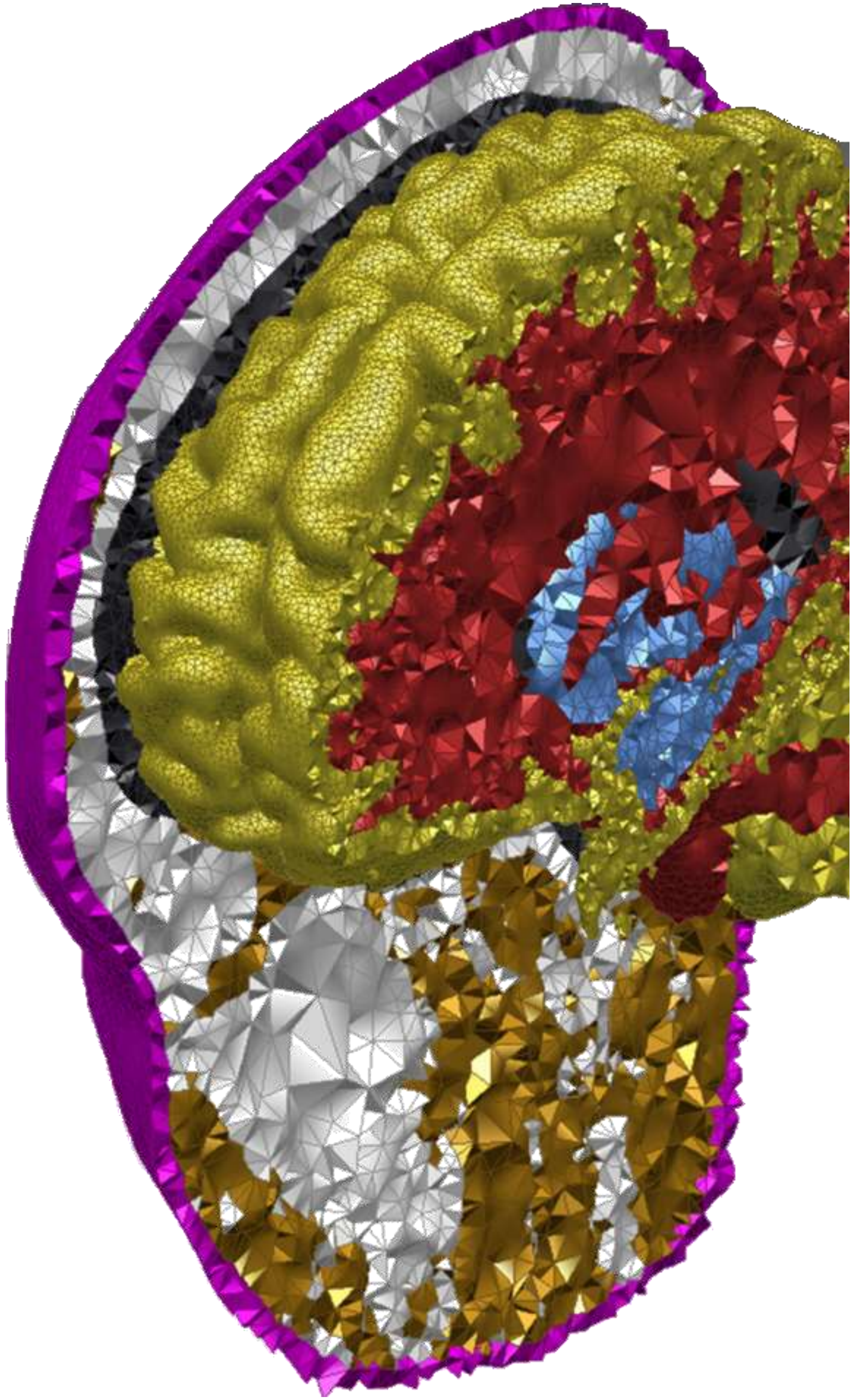
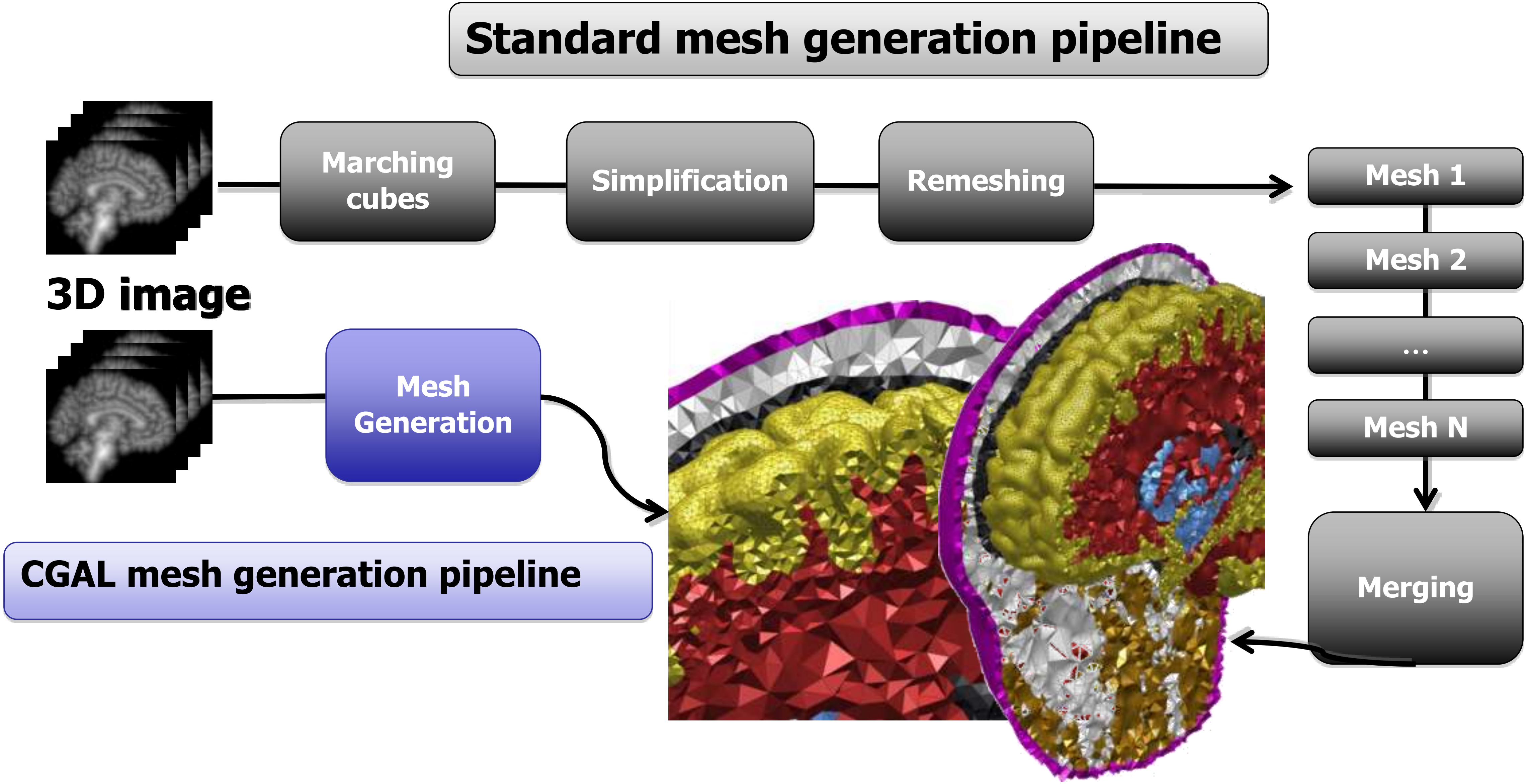


Image from Pons et al.



Added Value: Shortened Pipeline



Standard mesh generation pipeline

3D image

Marching cubes

Simplification

Remeshing

Mesh 1

Mesh 2

...

Mesh N

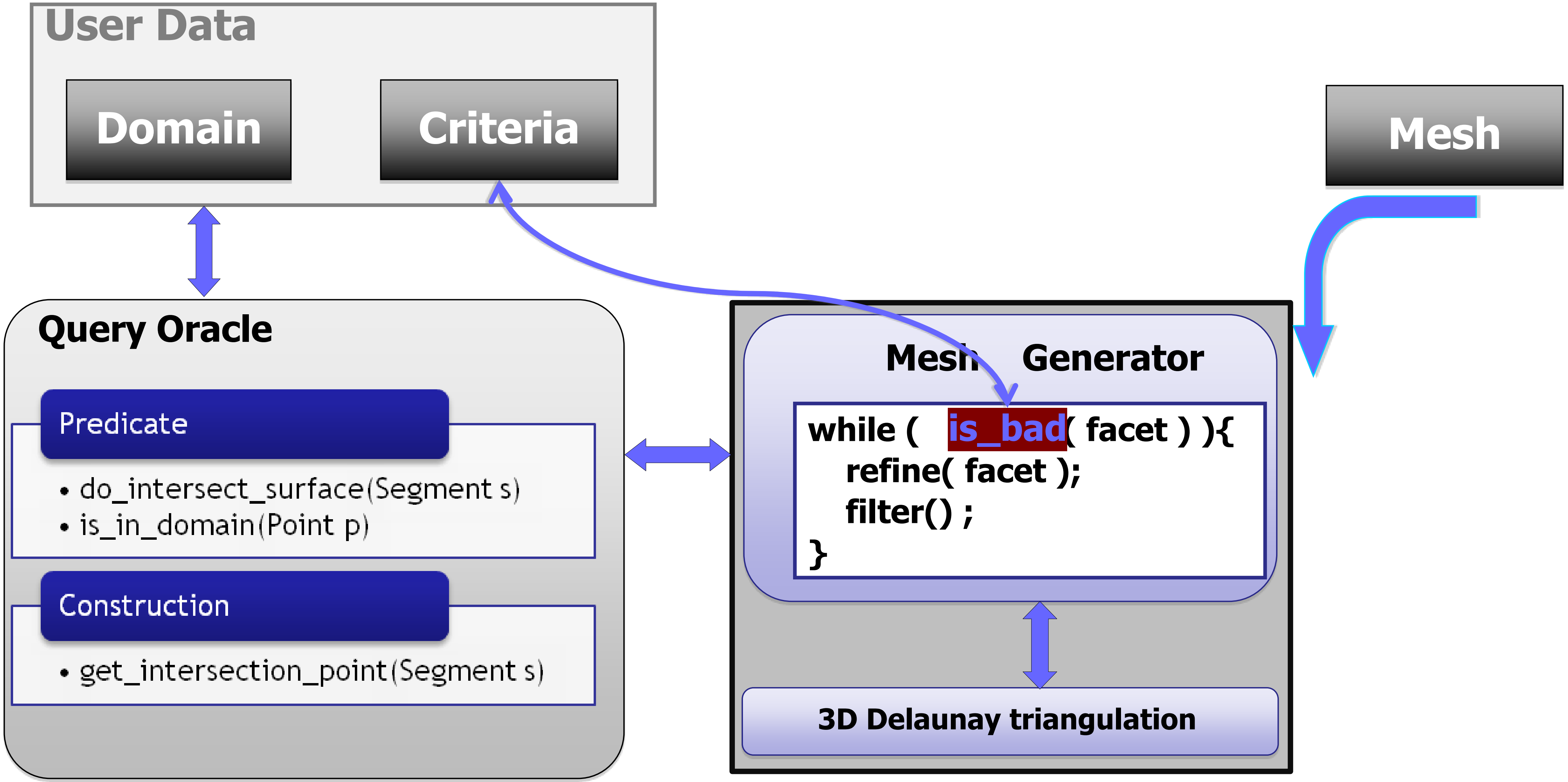
Merging

Mesh Generation

CGAL mesh generation pipeline

API

Overall Design



User Data

Domain

Criteria

Mesh

Query Oracle

Predicate

- do_intersect_surface(Segment s)
- is_in_domain(Point p)

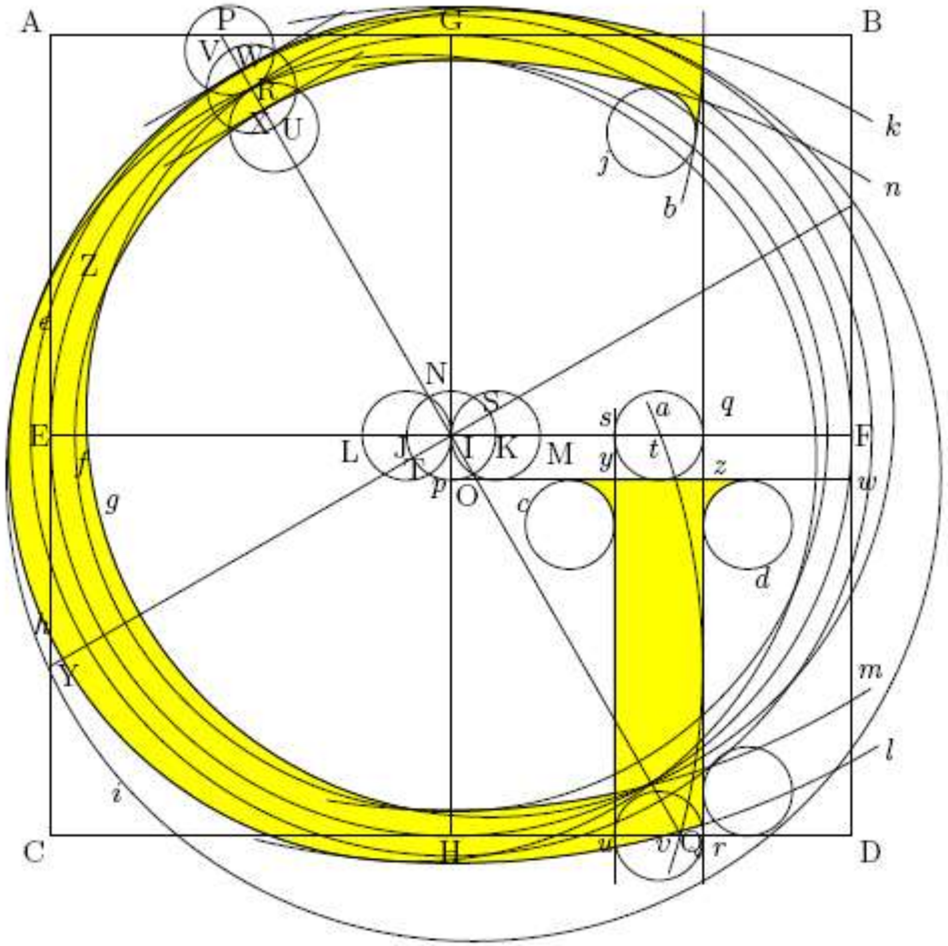
Construction

- get_intersection_point(Segment s)

Mesh Generator

```
while (  is_bad( facet ) ){
   refine( facet );
   filter() ;
}
```

3D Delaunay triangulation



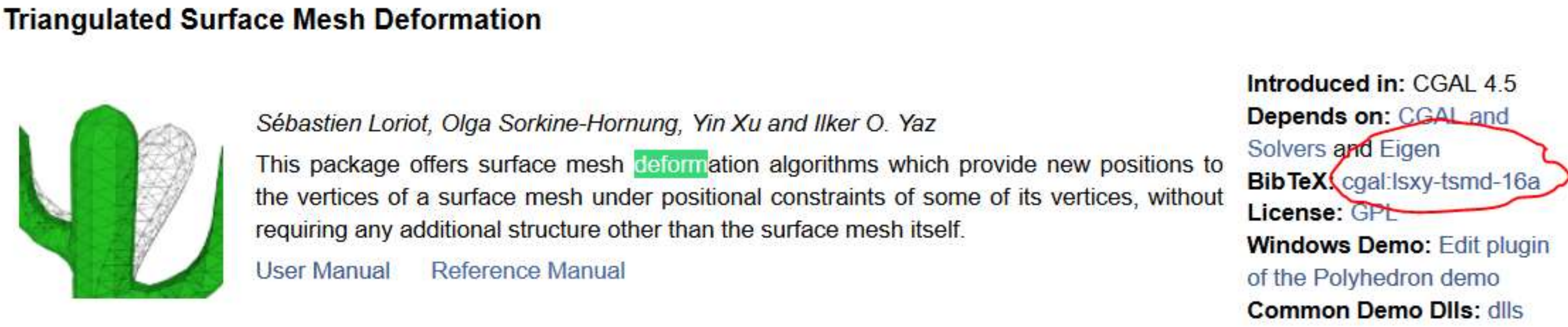
Use

Participate

Contribute

Acknowledge CGAL

cgal.org



Triangulated Surface Mesh Deformation

Sébastien Loriot, Olga Sorkine-Hornung, Yin Xu and Ilker O. Yaz

This package offers surface mesh deformation algorithms which provide new positions to the vertices of a surface mesh under positional constraints of some of its vertices, without requiring any additional structure other than the surface mesh itself.

User Manual Reference Manual

Introduced in: CGAL 4.5
Depends on: CGAL and Solvers and Eigen
BibTeX: cgal:lsxy-tsmd-16a
License: GPL
Windows Demo: Edit plugin of the Polyhedron demo
Common Demo Dlls: dlls

```
@incollection{cgal:lsxy-tsmd-16a,
  author = {S{\'e}bastien Loriot and Olga Sorkine-Hornung and Yin Xu and Ilker O. Yaz},
  title = {Triangulated Surface Mesh Deformation},
  publisher = {{CGAL Editorial Board}},
  edition = {{4.8.1}},
  booktitle = {{CGAL} User and Reference Manual},
  url = {http://doc.cgal.org/4.8.1/Manual/packages.html#PkgSurfaceMeshDeformationSummary},
  year = 2016
}
```

Bug Reports

- Report them so that we can fix them

- Be precise – We have no crystal ball
- Issue tracker on https://github.com/CGAL/cgal/issues
- Create a gist https://gist.github.com/

- If you have trivial fixes make a pull request (PR)

Contribute to CGAL

- New functionality for an existing class
- The prototype for your latest ECCV paper

- Review in your field of expertise

- Interface with another software project

Contributors ...

- stay owner of contributed package
- profit
  - from a well defined workflow
  - from shared infrastructure and maintenance
  - from peer-review
- get recognition and visibility

GeometryFactory

- 7 engineers, whereof 5 with a PhD

- Commercialization agreements with academic partners
- Sales of CGAL software components

- Development of new components
- Improvements of existing components