Mesh Generation

Andreas Fabri
GeometryFactory

Pierre Alliez
Inria
Outline

● Triangle meshing with CGAL

● 2D Delaunay mesh generation

● 3D Delaunay mesh generation
  ○ Delaunay refinement and filtering
  ○ Multiple types of domains
  ○ Feature preserving
  ○ Meshing from images
  ○ Mesh optimization
  ○ API

● Isotropic tetrahedral remeshing
Triangle Meshing with CGAL
CGAL Packages

- 2D Mesh Generation
- Surface Mesher (obsolete)
- 3D Mesh Generation
- Isotropic Remeshing
2D Mesh Generation
Delaunay Triangulation

A triangulation is a Delaunay triangulation if the circumscribing circle of any facet of the triangulation contains no vertices in its interior.
Constrained Delaunay Triangulation
Conforming Triangulation
Delaunay Mesh Refinement
Parameter: Shape

Lower bound on smallest angle

Input PSLG  
5 deg  
20.7 deg
Parameter: Size

No constraints  Uniform Sizing  Sizing function
Parameter: Seed points

Include and exclude components
Smoothing: Lloyd Iterations
Smoothing: Lloyd Iterations

The graph shows the distribution of angles over iterations of the Lloyd algorithm. The x-axis represents angles ranging from 0 to 180 degrees, and the y-axis represents the frequency of occurrence. The data includes:

- **Before**
- **Lloyd 10 it.**
- **Lloyd 100 it.**

The graph illustrates how the distribution changes with each iteration, indicating the effectiveness of the smoothing process.
```c++
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Constrained_Delaunay_triangulation_2.h>
#include <CGAL/Delaunay_mesher_2.h>

typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef CGAL::Constrained_Delaunay_triangulation_2<Kernel> CDT;
typedef CGAL::Delaunay_mesh_size_criteria_2<CDT> Mesh_criteria;

int main()
{
    CDT cdt;
    // insert points and constraints
    Mesh_criteria mesh_criteria(0.125, 0.5);
    CGAL::refine_Delaunay_mesh_2(cdt, mesh_criteria);
    return 0;
}
```
3D Mesh Generation
3D Mesh Generation
Delaunay Refinement and Filtering
3D Delaunay Mesh Generation
3D Delaunay Mesh Generation
3D Delaunay Mesh Generation
3D Delaunay Mesh Generation
3D Delaunay Mesh Generation
3D Delaunay Mesh Generation
3D Delaunay Mesh Generation
The Algorithm

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

Delaunay triangulation restricted to surface S

Voronoi edge \( \cap \) surface S

dual Voronoi edge

facet
Delaunay Refinement

**Bad facet** = large or badly shaped or large approximation error

![Diagram of Delaunay Refinement with a bad facet highlighted and a refinement point indicated]
Meshing an Implicit Function

```cpp
typedef CGAL::Exact_predicates_inexact_constructions_kernel K;
typedef CGAL::Labeled_mesh_domain_3<K> Mesh_domain;
typedef CGAL::Mesh_triangulation_3<Mesh_domain>::type Tr;
typedef CGAL::Mesh_complex_3_in_triangulation_3<Tr> C3t3;
typedef CGAL::Mesh_criteria_3<Tr> Mesh_criteria;
using CGAL::parameters;

int main()
{
    // Domain
    Mesh_domain domain =
        Mesh_domain::create_implicit_mesh_domain(
            function = [] (const K::Point_3& p)
            {
                return CGAL::squared_distance(p, K::Point_3(CGAL::ORIGIN))-1;
            },
            bounding_object = K::Sphere_3(CGAL::ORIGIN,2));

    // Set mesh criteria
    Mesh_criteria criteria(facet_angle=30, facet_size=0.1,
                            facet_distance=0.025,
                            cell_size=0.1);

    // Mesh generation
    C3t3 c3t3 = CGAL::make_mesh_3<C3t3>(domain, criteria);
}
```
Specifying a Sizing Field

```cpp
struct Spherical_sizing_field
{
    typedef Mesh_domain::Index Index;

    double operator()(const Point_3& p, const int, const Index&)
    {
        double sq_d_to_origin =
            CGAL::squared_distance(p, Point(CGAL::ORIGIN));

        return CGAL::abs(sqrt(sq_d_to_origin)-0.5) / 5. + 0.025;
    }
};
```
Specifying a Sizing Field

typedef CGAL::Labeled_mesh_domain_3<K> Mesh_domain;

typedef CGAL::Mesh_constant_domain_field_3 <
    Mesh_domain::R, Mesh_domain::Index> Sizing_field;

// Domain
Mesh_domain domain =
    Mesh_domain::create_labeled_image_mesh_domain(image);

// Sizing field:
// set global size to 8, kidney size (label 127) to 3
double kidney_size = 3.;
int volume_dimension = 3;
Sizing_field size(8);

size.set_size(kidney_size, volume_dimension,
    domain.index_from_subdomain_index(127));

// Mesh criteria
Mesh_criteria criteria(facet_angle=30, facet_size=6,
    facet_distance=2,
    cell_radius_edge_ratio=3,
    cell_size=size);

// Meshing
C3t3 c3t3 = CGAL::make_mesh_3<C3t3>(domain, criteria);
typedef CGAL::Labeled_mesh_domain_3<K> Mesh_domain;

typedef CGAL::Mesh_constant_domain_field_3 <
    Mesh_domain::R, Mesh_domain::Index> Distance_field;

// Domain
Mesh_domain domain =
    Mesh_domain::create_labeled_image_mesh_domain(image);

// Sizing field:
// set global size to 8, kidney size (label 127) to 3
double kidney_size = 3.;
int volume_dimension = 3;
Distance_field variable_distance;

// Mesh criteria
Mesh_criteria criteria(facet_angle=30,
    facet_size=6,
    facet_distance=variable_distance,
    cell_radius_edge_ratio=3,
    cell_size=6);

// Meshing
C3t3 c3t3 = CGAL::make_mesh_3<C3t3>(domain, criteria);
Manifold Criterion

- Input must be 2-manifold
- Sharp Features must be protected
About Initial Points

Small connected components

Thin feature
3D Mesh Generation
Multiple types of domains
Input Domains

- Level set of an implicit function
- Level set grey level image
- Segmented image
- Polyhedral surfaces
- Polyhedral complexes
- NURBS surfaces (in progress)
Added Value: Shortened Pipeline

**Standard mesh generation pipeline**

- 3D image
- Marching cubes
- Simplification
- Remeshing

**Mesh Generation**

**CGAL mesh generation pipeline**

Mesh 1
Mesh 2
...
Mesh N
Merging
Meshing an Implicit function from a Point Set

Poisson Surface Reconstruction

- Construct *Poisson implicit function* $P$ in ambient space around the input points
- Mesh the 0-surface of $P$
Poisson Surface Reconstruction

- Works well for uneven distribution of points
- Used for reconstruction of geological bodies
Poisson Surface Reconstruction

- Algorithm produces watertight surface
- For large holes use two-pass algorithm
Offset Mesh Generation

- Meshing an Implicit Function
- Can deal with a variable offset distance field
- No direct handling of sharp features
Level Set of Grey Level Image

Input: 3D voxel data for SEG Salt Model
Comparison with Marching Cubes

Delaunay Refinement

Marching Cubes in an octree
Polyhedral Surfaces as Input Domains

- *Surface to remesh*
- Volume delimited by a surface
- Surfaces inside a volume
- Polyhedral complex
Polyhedral Surfaces as Input Domains

- Surface to remesh
- Volume delimited by a surface
- Surfaces inside a volume
- Polyhedral complex
Polyhedral Surfaces as Input Domains

- Surface to remesh
- *Volume delimited by a surface*
- Surfaces inside a volume
- Polyhedral complex
Polyhedral Surfaces as Input Domains

- Surface to remesh
- *Volume delimited by a surface*
- Surfaces inside a volume
- Polyhedral complex
Polyhedral Surfaces as Input Domains

- Surface to remesh
- *Volume delimited by a surface*
- *Surfaces inside a volume*
- Polyhedral complex
Polyhedral Surfaces as Input Domains

- Surface to remesh
- Volume delimited by a surface
- Surfaces inside a volume
- *Polyhedral complex*
NURBS surfaces from CAD (in progress)

Input NURBS model

41 NURBS patches

Feature preserving mesh
Periodic Mesh Generation

- Efficiency: no duplication of points
- No constraint at the border of the domain
- Same options as the 3D Mesh Generation package
  (protection of features, optimization, etc.)
Defining Yet Another Domain

Mesh Generator

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

3D Delaunay triangulation

Query Oracle

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

Construction
- get_intersection_point(Segment s)

User Data

Domain
Criteria

Mesh
Oracle for a Domain

Query Oracle

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

- Construction
  - get_intersection_point(Segment s)

dual Voronoi edge
Develop a Hybrid Domain
Develop a Hybrid Domain

double sphere_function(const Point& p)
{
    return CGAL::squared_distance(p, CGAL::ORIGIN) - 1.0;
}

Implicit_domain domain(sphere_function);

Mesh_criteria criteria( facet_size = 0.1,
                        facet_distance = 0.025 );

C3t3 c3t3 = CGAL::make_mesh_3<C3t3>(domain, criteria);
Develop a Hybrid Domain

class Hybrid_domain {

    Implicit_domain & implicit;
    Polyhedral_domain & polyhedron;

public:
    Hybrid_domain(Implicit_domain & implicit, Polyhedral_domain & polyhedron);

    int is_in_domain(Point_3 p) {
        if(implicit.is_in_domain(p))
            return 2;
        else
            if(polyhedron.is_in_domain(p))
                return 1;
            else
                return 0;
    }
};
3D Mesh Generation
Feature Preserving
Sharp Features

Protecting balls method [Cheng et al. 2007]

- Cover the edges with balls
- Run the weighted version of Delaunay refinement

Segments joining centers of consecutive protecting balls are guaranteed to be edges in the mesh
Meshing an Implicit Function with 1D Features

```cpp
Polylines polylines (1);
Polyline_3& polyline = polylines.front();
for(int i = 0; i < 360; ++i)
{
    Point p (1,
        std::cos(i*CGAL_PI/180),
        std::sin(i*CGAL_PI/180));
    polyline.push_back(p);
}
polyline.push_back(polyline.front());

// Insert edge in domain
domain.add_features (polylines.begin(), polylines.end());
```
Sharp Features
Sharp Features - Adaptive Protecting Balls
Sharp Features
3D Mesh Generation
Meshing from Images
1D Features in Images

- Automatic detection and protection of intersections with the bounding box

Without features protection  
With features protection
1D Features in Images

- Detection of triple lines

[Hege, Hans-Christian, et al., 1997]
1D Features in Images

- Detection and protection of internal features before meshing

[Hege, Hans-Christian, et al., 1997]
Deal with Image in memory

Examples in CGAL

- Mesh_3/random_labeled_image.h
- CGALImageIO/extract_a_sub_image.cpp

Non-documented internal API

```cpp
#include <CGAL/ImageIO.h>

auto *new_image = ::_createImage(new_xdim, new_ydim, new_zdim, 1,
        image->vx, image->vy, image->vz, image->wdim,
        image->wordKind, image->sign);

auto* new_data = static_cast<char*>(new_image->data);

...

auto r = ::_writeImage(new_image, argv[2]);
::_freeImage(new_image);
```
3D Mesh Generation
Mesh Optimization
Delaunay Refinement may produce Slivers
Mesh Quality Optimization - Global

Increase minimal dihedral angle through post-processing
Sliver Perturbation moves vertices to locally improve dihedral angles

Uniform Sizing Field

Lipschitz Sizing Field
Mesh Quality Optimization - Local

Sliver Exudation adds weights to change connectivity
Mesh Quality Optimization - Combined

Original Mesh
(50k vertices, 290k tets, 10 seconds)

ODT smoothing
(global optimization, 110s)

ODT + Sliver perturbation
(local optimization, 40s)
3D Mesh Generation
API
typedef CGAL::Exact_predicates_inexact_constructions_kernel K;
typedef CGAL::Labeled_mesh_domain_3<K> Mesh_domain;
typedef CGAL::Mesh_triangulation_3<Mesh_domain>::type Tr;
typedef CGAL::Mesh_complex_3_in_triangulation_3<Tr> C3t3;
typedef CGAL::Mesh_criteria_3<Tr> Mesh_criteria;

using CGAL::parameters;

int main()
{
    // Domain
    Mesh_domain domain = Mesh_domain::create_implicit_mesh_domain(
        function = [] (const K::Point_3& p)
            { return CGAL::squared_distance(p, K::Point_3(CGAL::ORIGIN))-1; },
        bounding_object = K::Sphere_3(CGAL::ORIGIN,2.));

    // Set mesh criteria
    Mesh_criteria criteria(facet_angle=30, facet_size=0.1,
        facet_distance=0.025, cell_size=0.1);

    // Mesh generation
    C3t3 c3t3 = CGAL::make_mesh_3<C3t3>(domain, criteria);
}

Named Parameters
// Set mesh criteria
Mesh_criteria criteria(
    facet_angle=30,
    facet_size=0.1,
    facet_distance=0.025,
    cell_size=0.1);

// Mesh generation
template<class C3T3 , class MD , class MC >
C3T3 CGAL::make_mesh_3(const MD& domain,
    const MC& criteria,
    parameters::internal::Features_options features =
        parameters::features(domain),
    parameters::internal::Lloyd_options lloyd = parameters::no_lloyd(),
    parameters::internal::Odt_options odt = parameters::no_odt(),
    parameters::internal::Perturb_options perturb = parameters::perturb(),
    parameters::internal::Exude_options exude = parameters::exude(),
    parameters::internal::Manifold_options manifold = parameters::non_manifold())
### Lloyd Optimization

```cpp
parameters::internal::Lloyd_options
CGAL::parameters::lloyd(
    double parameters::time_limit = 0,
    std::size_t parameters::max_iteration_number = 0,
    double parameters::convergence = 0.02,
    double parameters::freeze_bound = 0.01,
    bool parameters::do_freeze = true)
```

#### Usage code example

```cpp
// Mesh generation with lloyd optimization step
C3t3 c3t3 = make_mesh_3<c3t3>(domain,
    criteria,
    parameters::lloyd(parameters::time_limit = 10));
```
```cpp
typedef CGAL::Exact_predicates_inexact_constructions_kernel K;
typedef CGAL::Labeled_mesh_domain_3<K> Mesh_domain;
typedef CGAL::Mesh_triangulation_3<Mesh_domain>::type Tr;
```
Mesh_complex_3_in_triangulation_3

typedef CGAL::Mesh_complex_3_in_triangulation_3<Tr> C3t3;
...
...
C3t3 c3t3 = CGAL::make_mesh_3<C3t3>(domain, criteria);

- A wrapper around a Triangulation_3
- Description of a subset of geometric and combinatorial features
  - 0D - Corner vertices
  - 1D - Feature edges
  - 2D - Surface facets
  - 3D - Volume cells
typedef CGAL::Mesh_complex_3_in_triangulation_3<Tr> C3t3;
C3t3 c3t3;

• API - for each dimension of simplex
  ○ Accessors
    ```cpp
    bool b = c3t3.is_in_complex(simplex);
    Std::size_t n = c3t3.number_of_cells();
    ```
  ○ Modificators
    ```cpp
c3t3.add_to_complex(simplex, simplex_index);
c3t3.remove_from_complex(simplex);
    ```
  ○ Iterators
    ```cpp
c3t3.cells_in_complex_begin();
c3t3.cells_in_complex_end();
    ```

• C3t3 describes a complex of each dimension 0, 1, 2, 3
Mesh_complex_3_in_triangulation_3 - API

- Two Concepts
  - MeshComplex_3InTriangulation_3
    - Cells and Facets
  - MeshComplexWithFeatures_3InTriangulation_3
    - Edges and Vertices

- One Class
  - CGAL::Mesh_complex_3_in_triangulation_3
typedef CGAL::Mesh_complex_3_in_triangulation_3<Tr> C3t3;
C3t3 c3t3;

- Export facets complex
  - to CGAL::Surface_mesh
    - `CGAL::facets_in_complex_3_to_triangle_mesh(c3t3, surface_mesh);`
  - to OFF file
    - `c3t3.output_boundary_to_off(ostream);`
Isotropic Tetrahedral Remeshing
Isotropic Tetrahedral Remeshing

- Post-processing
- CGAL::Triangulation_3
  - Triangulation of the convex hull
  - Non-Delaunay

[Faraj et al. 2016]
Isotropic Tetrahedral Remeshing

- Post-processing
- CGAL::Triangulation_3

- Iterative improvements of tetrahedral meshes by consecutive atomic operations
  - Edge split
  - Edge collapse
  - Edge flip
  - Global smoothing using vertex relocations
  - Re-projection of boundary vertices
Isotropic Tetrahedral Remeshing

- Post-processing
- CGAL::Triangulation_3

- Iterative improvements of tetrahedral meshes by consecutive atomic operations
  - Edge split
  - Edge collapse
  - Edge flip
  - Global smoothing using vertex relocations
  - Re-projection of boundary vertices

- Deals with
  - Multi-domains,
  - Boundaries,
  - Features
  - Preserves the geometry of subdomains throughout the remeshing process.[Faraj et al. 2016]
Isotropic Tetrahedral Remeshing

Input (Delaunay Mesh)

Dihedral angle distribution

Input Remeshed
Isotropic Tetrahedral Remeshing

Input (Delaunay Mesh)

Input Remeshed

Dihedral angle distribution
Isotropic Tetrahedral Remeshing

Input (Delaunay Mesh)

Dihedral angle distribution

Input Remeshed
Isotropic Tetrahedral Remeshing
Isotropic Tetrahedral Remeshing
Isotropic Tetrahedral Remeshing - API

```cpp
CGAL::Triangulation_3 tr;
...
CGAL::tetrahedral_isotropic_remeshing(tr,
  target_edge_length);
```

Optional Named Parameters

- `number_of_iterations`
- `remesh_boundaries`
- `smooth_constrained_edges`
- `edge_is_constrained_map`
- `facet_is_constrained_map`
- `cell_is_selected_map`
Thank you.