Computational Geometry

Efi Fogel

Tel Aviv University

Computational Geometry Algorithm Library
May. 11th, 2020
Outline

1 Cgal
   • Introduction
   • Content
   • Literature
   • Geometry Factory
   • Details

2 Arrangement
   • Minimum Area Triangle
   • Spherical Gaussian Map
Outline

1. **CGAL**
   - Introduction
   - Content
   - Literature
   - Geometry Factory
   - Details

2. **Arrangement**
   - Minimum Area Triangle
   - Spherical Gaussian Map
Cgal: Mission

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

Cgal Project Proposal, 1996
Cgal Facts

- A collection of software packages written in C++
- Adheres the *generic programming* paradigm
- Development started in 1995
- An open source library
- Several active contributor sites
- High search-engine ranking for www.cgal.org

- Used in a diverse range of domains
  - e.g., computer graphics, scientific visualization, computer aided design and modeling, additive manufacturing, geographic information systems, molecular biology, medical imaging, and VLSI
- The de-facto standard in applied Computational Geometry
Cgal in Numbers

600,000 lines of C++ code
10,000 downloads per year not including Linux distributions
4,500 manual pages (user and reference manual)
1,000 subscribers to user mailing list
200 commercial users
120 packages
30 active developers
6 months release cycle
2 licenses: Open Source and commercial
## CGAL History

<table>
<thead>
<tr>
<th>Year</th>
<th>Version Released</th>
<th>Other Milestones</th>
</tr>
</thead>
<tbody>
<tr>
<td>1996</td>
<td></td>
<td><strong>CGAL founded</strong></td>
</tr>
<tr>
<td>1998</td>
<td>July 1.1</td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td></td>
<td><strong>Work continued after end of European support</strong></td>
</tr>
<tr>
<td>2001</td>
<td>Aug 2.3</td>
<td><strong>Editorial Board</strong> established</td>
</tr>
<tr>
<td>2002</td>
<td>May 2.4</td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>Nov 3.0</td>
<td><strong>Geometry Factory</strong> founded</td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td><strong>CMake</strong></td>
</tr>
<tr>
<td>2009</td>
<td>Jan 3.4, Oct 3.5</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Mar 3.6, Oct 3.7</td>
<td><strong>Google Summer of Code (GSoC) 2010</strong></td>
</tr>
<tr>
<td>2011</td>
<td>Apr 3.8, Aug 3.9</td>
<td><strong>GSoC 2011</strong></td>
</tr>
<tr>
<td>2012</td>
<td>Mar 4.0, Oct 4.1</td>
<td><strong>GSoC 2012</strong></td>
</tr>
<tr>
<td>2013</td>
<td>Mar 4.2, Oct 4.3</td>
<td><strong>GSoC 2013, Doxygen</strong></td>
</tr>
<tr>
<td>2014</td>
<td>Apr 4.4, Oct 4.5</td>
<td><strong>GSoC 2014</strong></td>
</tr>
<tr>
<td>2015</td>
<td>Apr 4.6, Oct 4.7</td>
<td><strong>GitHub, HTML5, Main repository</strong> made public</td>
</tr>
<tr>
<td>2016</td>
<td>Apr 4.8, Sep 4.9</td>
<td><strong>Only headers, 20th anniversary</strong></td>
</tr>
<tr>
<td>2017</td>
<td>May 4.10, Sep 4.11</td>
<td><strong>CTest, GSoC 2017</strong></td>
</tr>
<tr>
<td>2018</td>
<td>Apr 4.12</td>
<td><strong>GSoC 2018</strong></td>
</tr>
<tr>
<td>2019</td>
<td>Nov 5.0</td>
<td><strong>C++14, GSoC 2019</strong></td>
</tr>
<tr>
<td>2020</td>
<td>5.1</td>
<td><strong>GSoC 2020</strong></td>
</tr>
</tbody>
</table>
CGAL Properties

- **Reliability**
  - Explicitly handles degeneracies
  - Follows the Exact Geometric Computation (EGC) paradigm

- **Efficiency**
  - Depends on leading 3rd party libraries
    - e.g., **Boost**, **Gmp**, **Mpfr**, **Qt**, **Eigen**, **Tbb**, and **Core**
  - Adheres to the generic-programming paradigm
    - Polymorphism is resolved at compile time

→ The best of both worlds ←
Cgal Properties, Cont

- **Flexibility**
  - Adaptable, e.g., graph algorithms can directly be applied to Cgal data structures
  - Extensible, e.g., data structures can be extended

- **Ease of Use**
  - Has didactic and exhaustive Manuals
  - Follows standard concepts (e.g., C++ and STL)
  - Has a modular structure, e.g., geometry and topology are separated
  - Characterizes with a smooth learning-curve
Outline

1 Cgal
   • Introduction
   • Content
   • Literature
   • Geometry Factory
   • Details

2 Arrangement
   • Minimum Area Triangle
   • Spherical Gaussian Map
2D Algorithms and Data Structures

- Triangulations
- Mesh Generation
- Polyline Simplification
- Voronoi Diagrams
- Arrangements
- Boolean Operations
- Neighborhood Queries
- Minkowski Sums
- Straight Skeleton
3D Algorithms and Data Structures

- Triangulations
- Mesh Generation
- Polyhedral Surface
- Deformation
- Boolean Operations
- Mesh Simplification
- Skeleton
- Segmentation
- Classification
- Hole Filling
Outline

1 CGAL
   • Introduction
   • Content
   • Literature
   • Geometry Factory
   • Details

2 Arrangement
   • Minimum Area Triangle
   • Spherical Gaussian Map
The CGAL Project. 
*CGAL User and Reference Manual.*

Efi Fogel, Ron Wein, and Dan Halperin. 
*CGAL Arrangements and Their Applications, A Step-by-Step Guide.*

Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Levy. 
*Polygon Mesh Processing.*

A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr. 
On the design of CGAL a computational geometry algorithms library. 

A. Fabri and S. Pion. 
A generic lazy evaluation scheme for exact geometric computations. 
In *2nd Library-Centric Software Design Workshop, 2006.*

M. H. Overmars. 
Designing the computational geometry algorithms library CGAL. 

Many Many Many papers
Outline

1 **CGAL**
   - Introduction
   - Content
   - Literature
   - Geometry Factory
   - Details

2 Arrangement
   - Minimum Area Triangle
   - Spherical Gaussian Map
Some CGAL Commercial Customers
CGAL Commercial Customers, Geographic Segmentation
Outline

1. **CGAL**
   - Introduction
   - Content
   - Literature
   - Geometry Factory
   - Details

2. **Arrangement**
   - Minimum Area Triangle
   - Spherical Gaussian Map
CGAL Structure

Basic Library
Algorithms and Data Structures
e.g., Triangulations, Surfaces, and Arrangements

Kernel
Elementary geometric objects
Elementary geometric computations on them

Support Library
Configurations, Assertions,…

Visualization
Files
I/O
Number Types
Generators
**CGAL Kernel Concept**

- Geometric objects of constant size.
- Geometric operations on object of constant size.

<table>
<thead>
<tr>
<th>Primitives 2D, 3D, dD</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>point</td>
<td>comparison</td>
</tr>
<tr>
<td>vector</td>
<td>orientation</td>
</tr>
<tr>
<td>triangle</td>
<td>containment</td>
</tr>
<tr>
<td>iso rectangle</td>
<td>intersection</td>
</tr>
<tr>
<td>circle</td>
<td>squared distance</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
CGAL Kernel Affine Geometry

point - origin $\rightarrow$ vector
point - point $\rightarrow$ vector
point + vector $\rightarrow$ point

point + point $\leftarrow$ Illegal

midpoint(a, b) = $a + 1/2 \times (b - a)$
CGAL Kernel Classification

- **Dimension:** 2, 3, arbitrary
- **Number types:**
  - **Ring:** $+, -, \times$
  - **Euclidean ring** (adds integer division and gcd) (e.g., CGAL::Gmpz).
  - **Field:** $+, -, \times, /$ (e.g., CGAL::Gmpq).
  - **Exact sign evaluation for expressions with roots** (Field_with_sqr).
- **Coordinate representation**
  - **Cartesian**—requires a field number type or **Euclidean ring** if no constructions are performed.
  - **Homogeneous**—requires **Euclidean ring**.
- **Reference counting**
- **Exact, Filtered**
## CGAL Kernels and Number Types

### Cartesian representation

| point | $x = \frac{hx}{hw}$ | $y = \frac{hy}{hw}$ |

### Homogeneous representation

| point | $hx$ | $hy$ | $hw$ |

### Intersection of two lines

\[
\begin{align*}
\begin{cases}
    a_1 x + b_1 y + c_1 = 0 \\
    a_2 x + b_2 y + c_2 = 0
\end{cases}
\end{align*}
\]

\[
(x, y) = \left( \left| \begin{array}{cc}
    b_1 & c_1 \\
    b_2 & c_2 \\
    a_1 & b_1 \\
    a_2 & b_2 \\
\end{array} \right|, - \left| \begin{array}{cc}
    a_1 & c_1 \\
    a_2 & c_2 \\
    a_1 & b_1 \\
    a_2 & b_2 \\
\end{array} \right| \right)
\]

### Field operations

\[
(hx, hy, hw) = \left( \left| \begin{array}{cc}
    b_1 & c_1 \\
    b_2 & c_2 \\
\end{array} \right|, - \left| \begin{array}{cc}
    a_1 & c_1 \\
    a_2 & c_2 \\
\end{array} \right|, \left| \begin{array}{cc}
    a_1 & b_1 \\
    a_2 & b_2 \\
\end{array} \right| \right)
\]

### Ring operations
#if 1
    typedef CORE:: Expr NT;
    typedef CGAL:: Cartesian<NT> Kernel;
    NT sqrt2 = CGAL:: sqrt(NT(2));
#else
    typedef double NT;
    typedef CGAL:: Cartesian<NT> Kernel;
    NT sqrt2 = sqrt(2);
#endif
Kernel::Point_2 p(0,0), q(sqrt2, sqrt2);
Kernel::Circle_2 C(p, 4);
assert (C.has_on_boundary(q));

- OK if NT supports exact sqrt.
- **Assertion violation** otherwise.
CGAL Pre-defined Cartesian Kernels

- Support construction of points from `double` Cartesian coordinates.
- Support exact geometric predicates.
- Handle geometric constructions differently:
  - `CGAL::Exact_predicates_inexact_constructions_kernel`
    - Geometric constructions may be inexact due to round-off errors.
    - It is however more efficient and sufficient for most CGAL algorithms.
  - `CGAL::Exact_predicates_exact_constructions_kernel`
  - `CGAL::Exact_predicates_exact_constructions_kernel_with_sqrt`
    - Its number type supports the exact square-root operation.
CGAL Special Kernels

- Filtered kernels
- 2D circular kernel
- 3D spherical kernel

Refer to CGAL’s manual for more details.
CGAL Basic Library

- Generic data structures are parameterized with Traits
  - Separates algorithms and data structures from the geometric kernel.
- Generic algorithms are parameterized with iterator ranges
  - Decouples the algorithm from the data structure.
CGAL Components Developed at Tel Aviv University

- 2D Arrangements
- 2D Regularized Boolean Set-Operations
- 2D Minkowski Sums
- 2D Envelopes
- 3D Envelopes
- 2D Snap Rounding
- 2D Set Movable Separability (2D Casting)
- 3D Set Movable Separability (3D Casting)
- Inscribed Areas / 2D Largest empty iso rectangle
- CGAL Python bindings for the above
Outline

1 Cgal
   - Introduction
   - Content
   - Literature
   - Geometry Factory
   - Details

2 Arrangement
   - Minimum Area Triangle
   - Spherical Gaussian Map
2D Arrangements

Definition (Arrangement)

Given a collection $C$ of curves on a surface, the arrangement $\mathcal{A}(C)$ is the partition of the surface into vertices, edges and faces induced by the curves of $C$.

An arrangement of circles in the plane
An arrangement of lines in the plane
An arrangement of great-circle arcs on a sphere
Arrangement_2<Traits, Dcel>

- Is the main component in the 2D Arrangements package.
- An instance of this class template represents 2D arrangements.
- The representation of the arrangements and the various geometric algorithms that operate on them are separated.
- The topological and geometric aspects are separated.
  - The Traits template-parameter must be substituted by a model of a geometry-trait concept, e.g., ArrangementBasicTraits_2.
    - Defines the type X_monotone_curve_2 that represents $x$-monotone curves.
    - Defines the type Point_2 that represents two-dimensional points.
    - Supports basic geometric predicates on these types.
  - The Dcel template-parameter must be substituted by a model of the ArrangementDcel concept, e.g., Arr_default_dcel<Traits>.
Arrangement Background

- Arrangements have numerous applications
  - robot motion planning, computer vision, GIS, optimization, computational molecular biology

A planar map of the Boston area showing the top of the arm of Cape Cod.

Raw data comes from the US Census 2000 TIGER/line data files
Outline

1 CGAL
   • Introduction
   • Content
   • Literature
   • Geometry Factory
   • Details

2 Arrangement
   • Minimum Area Triangle
   • Spherical Gaussian Map
Point-Line Duality Transform

- Points and lines are transformed into lines and points, respectively.
  
  **Primal Plane**    **Dual Plane**
  
  the point $p : (a, b)$    the line $p^* : y = ax - b$
  
  the line $l : y = cx + d$    the point $l^* : (c, -d)$

- This duality transform does not handle vertical lines!

- The transform is incidence preserving.
- The transform preserves the above/below relation.
- The transform preserves the vertical distance between a point and a line.
Application: Minimum-Area Triangle

Application (Minimum-Area Triangle)

Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of $n$ points in the plane, find three distinct points $p_i, p_j, p_k \in P$ such that the area of the triangle $\triangle p_ip_jp_k$ is minimal among all other triangles defined by three distinct points in $P$. 

A naive algorithm requires $O(n^3)$ time. It is possible to compute in $O(n^2)$ time. The analysis of the algorithm time-complexity uses the zone complexity theorem.
**Application: Minimum-Area Triangle**

**Application (Minimum-Area Triangle)**

Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of $n$ points in the plane, find three distinct points $p_i, p_j, p_k \in P$ such that the area of the triangle $\triangle p_ip_jp_k$ is minimal among all other triangles defined by three distinct points in $P$.

- A naive algorithm requires $O(n^3)$ time.
Application: Minimum-Area Triangle

Application (Minimum-Area Triangle)

Given a set \( P = \{ p_1, p_2, \ldots, p_n \} \) of \( n \) points in the plane, find three distinct points \( p_i, p_j, p_k \in P \) such that the area of the triangle \( \triangle p_ipjpk \) is minimal among all other triangles defined by three distinct points in \( P \).

- A naive algorithm requires \( O(n^3) \) time.
- It is possible to compute in \( O(n^2) \) time.
  - The analysis of the algorithm time-complexity uses the zone complexity theorem.
Minimum Area Triangle: Duality

- $P^*$—the set of lines dual to the input points.
  - $P^*$ does not contain vertical lines.

- $p_i^*, p_j^* \in P^*$

- $\ell_{ij}^*$—the point of intersection between $p_i^*$ and $p_j^*$.

- $\ell_{ij}$—the line that contains $p_i$ and $p_j$.

- $p_k$—the point that defines the minimum-area triangle with $p_i$ and $p_j$.

- $p_k$ is the closest point to $\ell_{ij}$ \Rightarrow $p_k$ is the closest point to $\ell_{ij}$ in the vertical distance.

- $p_k^*$—the line immediately above or below the point $\ell_{ij}^*$.
Outline

1. **CGAL**
   - Introduction
   - Content
   - Literature
   - Geometry Factory
   - Details

2. **Arrangement**
   - Minimum Area Triangle
   - Spherical Gaussian Map
Gaussian Map (Normal Diagram) in 2D

Definition (Gasusian map or normal diagram)

The Gaussian map of a convex polygon $P$ is the decomposition of $S$ into maximal connected arcs so that the extremal point of $P$ is the same for all directions within one arc.

An arc of directions between $\vec{d}_1$ and $\vec{d}_2$.

Generalizes to higher dimensions.
Gasusian Map (Normal Diagram) in 3D

Definition (Gasusian map or normal diagram in 3D)

The Gaussian map of a convex polytope $P$ in $\mathbb{R}^3$, denoted as $\mathcal{G}\mathcal{M}(\partial P)$, is the decomposition of $S^2$ into maximal connected regions so that the extremal point of $P$ is the same for all directions within one region:

$\mathcal{G}\mathcal{M}$ is a set-valued function from $\partial P$ to $S^2$. $\mathcal{G}\mathcal{M}(p \in \partial P) =$ the set of outward unit normals to support planes of $P$ at $p$.

- $v, e, f$—a vertex, an edge, a facet of $P$
- $\mathcal{G}\mathcal{M}(f) =$ outward unit normal to $f$
- $\mathcal{G}\mathcal{M}(e) =$ geodesic segment
- $\mathcal{G}\mathcal{M}(v) =$ spherical polygon
- $\mathcal{G}\mathcal{M}(P)$ is an arrangement on $S^2$
- $\mathcal{G}\mathcal{M}(P)$ is unique.

- If each face $\mathcal{G}\mathcal{M}(v)$ is extended with $v$, $\Rightarrow \mathcal{G}\mathcal{M}^{-1}(\mathcal{G}\mathcal{M}(P)) = P$