Algorithmic Robotics and Motion Planning

Efi Fogel

Tel Aviv University

CGAL 2D Arrangements
Apr. 9th, 2018
Outline

1 CGAL 2D Arrangements
   • Representation
   • Queries
     • Vertical Decomposition
     • Point Location Queries
   • The Zone Computation Algorithmic Framework
   • The Plane Sweep Algorithmic Framework
   • Arrangement of Unbounded Curves
   • Arrangement-Traits Classes
   • Extending the Arrangement
   • Map Overlay
   • Adapting to Boost Graphs
   • Arrangement on Surfaces
   • Literature
Two Dimensional Arrangements

Definition (Arrangement)

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

An arrangement of circles in the plane.

An arrangement of lines in the plane.

An arrangement of great-circle arcs on a sphere.
Outline

1. **Cgal 2D Arrangements**
   - Representation
   - Queries
     - Vertical Decomposition
     - Point Location Queries
   - The Zone Computation Algorithmic Framework
   - The Plane Sweep Algorithmic Framework
   - Arrangement of Unbounded Curves
   - Arrangement-Traits Classes
   - Extending the Arrangement
   - Map Overlay
   - Adapting to **Boost** Graphs
   - Arrangement on Surfaces
   - Literature
The CGAL Arrangement_on_surface_2 Package

- Constructs, maintains, modifies, traverses, queries, and presents arrangements on two-dimensional parametric surfaces.
- Complete and Robust
  - All inputs are handled correctly (including degenerate input).
  - Exact number types are used to achieve robustness.
- Generic—easy to interface, extend, and adapt
- Modular—geometric and topological aspects are separated
- Supports among the others:
  - various point location strategies
  - zone-construction paradigm
  - sweep-line paradigm
    - vertical decomposition
    - overlay computation
    - batched point location
- Part of the CGAL basic library
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-traits 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>`.
The Doubly-Connected Edge List

- One of a family of combinatorial data-structures called the *halfedge data-structures*.
- Represents each edge using a pair of directed *halfedges*.
- Maintains incidence relations among cells of 0 (vertex), 1 (edge), and 2 (face) dimensions.

- The target vertex of a halfedge and the halfedge are *incident* to each other.
- The source and target vertices of a halfedge are *adjacent*. 

The Doubly-Connected Edge List Components

- **Vertex**
  - An incident halfedge pointing at the vertex.

- **Halfedge**
  - The opposite halfedge.
  - The previous halfedge in the component boundary.
  - The next halfedge in the component boundary.
  - The target vertex of the halfedge.
  - The incident face.

- **Face**
  - An incident halfedge on the outer $C_{cb}$.
  - An incident halfedge on each inner $C_{cb}$.

- **Connected component of the boundary ($C_{CB}$)**
  - The circular chains of halfedges around faces.
Arrangement Representation

- The halfedges incident to a vertex form a circular list.
- The halfedges are clockwise oriented around the vertex.
- The halfedges around faces form circular chains.
- All halfedges of a chain are incident to the same face.
- The halfedges are counterclockwise oriented along the boundary.
- Geometric interpretation is added by classes built on top of the halfedge data-structure.
Modifying the Arrangement

Inserting a curve that induces a new hole inside the face $f$, $\text{arr} \cdot \text{insert in face interior} \left(c, f\right)$.

Inserting a curve from an existing vertex $u$ that corresponds to one of its endpoints, $\text{insert from left vertex} \left(c, v\right)$, $\text{insert from right vertex} \left(c, v\right)$.

Inserting an $x$-monotone curve, the endpoints of which correspond to existing vertices $v_1$ and $v_2$, $\text{insert at vertices} \left(c, v_1, v_2\right)$.

- The new pair of halfedges close a new face $f'$.
- The hole $h_1$, which belonged to $f$ before the insertion, becomes a hole in this new face.
Outline

1 CGAL 2D Arrangements
   ● Representation
   ● Queries
      ● Vertical Decomposition
      ● Point Location Queries
   ● The Zone Computation Algorithmic Framework
   ● The Plane Sweep Algorithmic Framework
   ● Arrangement of Unbounded Curves
   ● Arrangement-Traits Classes
   ● Extending the Arrangement
   ● Map Overlay
   ● Adapting to BOOST Graphs
   ● Arrangement on Surfaces
   ● Literature
Vertical Decomposition

- Is a refinement of the original subdivision $\mathcal{A}$ of $n$ edges.

- In the plane
  - Contains $O(n)$ pseudo trapezoids (triangles and trapezoids).
  - A pseudo trapezoid is determined by
    - 2 vertices $\text{left}(\Delta)$ and $\text{right}(\Delta)$, and
    - 2 segments $\text{top}(\Delta)$ and $\text{bottom}(\Delta)$.

- Generalizes to higher dimensions and arrangements induces by well behaved objects.
Arrangement Point Location

Given a subdivision $A$ of the space into cells and a query point $q$, find the cell of $A$ containing $q$. 

![Diagram of a subdivision with a query point $q$.]
Arrangement Point Location

Given a subdivision $A$ of the space into cells and a query point $q$, find the cell of $A$ containing $q$. 

![Diagram of a cell containing a query point $q$.]
Arrangement Point Location

Given a subdivision $A$ of the space into cells and a query point $q$, find the cell of $A$ containing $q$.

- In degenerate situations the query point can...
Arrangement Point Location

Given a subdivision $A$ of the space into cells and a query point $q$, find the cell of $A$ containing $q$.

- In degenerate situations the query point can
  - lie on an edge, or
Arrangement Point Location

Given a subdivision $A$ of the space into cells and a query point $q$, find the cell of $A$ containing $q$.

- In degenerate situations the query point can
  - lie on an edge, or
Arrangement Point Location

Given a subdivision $A$ of the space into cells and a query point $q$, find the cell of $A$ containing $q$.

- In degenerate situations the query point can
  - lie on an edge, or
  - coincide with a vertex.
Arrangement Point Location

Given a subdivision $A$ of the space into cells and a query point $q$, find the cell of $A$ containing $q$.

- In degenerate situations the query point can
  - lie on an edge, or
  - coincide with a vertex.
Outline

1. **CGAL 2D Arrangements**
   - Representation
   - Queries
     - Vertical Decomposition
     - Point Location Queries
   - The Zone Computation Algorithmic Framework
   - The Plane Sweep Algorithmic Framework
   - Arrangement of Unbounded Curves
   - Arrangement-Traits Classes
   - Extending the Arrangement
   - Map Overlay
   - Adapting to **BOOST** Graphs
   - Arrangement on Surfaces
   - Literature
Outline

1. **CGAL 2D Arrangements**
   - Representation
   - Queries
     - Vertical Decomposition
     - Point Location Queries
   - The Zone Computation Algorithmic Framework
   - The Plane Sweep Algorithmic Framework
   - Arrangement of Unbounded Curves
   - Arrangement-Traits Classes
   - Extending the Arrangement
   - Map Overlay
   - Adapting to Boost Graphs
   - Arrangement on Surfaces
   - Literature
Outline

1 Cgal 2D Arrangements
   • Representation
   • Queries
     • Vertical Decomposition
     • Point Location Queries
   • The Zone Computation Algorithmic Framework
   • The Plane Sweep Algorithmic Framework
   • Arrangement of Unbounded Curves
   • Arrangement-Traits Classes
   • Extending the Arrangement
   • Map Overlay
   • Adapting to Boost Graphs
   • Arrangement on Surfaces
   • Literature
Outline

1 Cgal 2D Arrangements
   - Representation
   - Queries
     - Vertical Decomposition
     - Point Location Queries
   - The Zone Computation Algorithmic Framework
   - The Plane Sweep Algorithmic Framework
   - Arrangement of Unbounded Curves
   - Arrangement-Traits Classes
     - Extending the Arrangement
     - Map Overlay
     - Adapting to Boost Graphs
     - Arrangement on Surfaces
     - Literature
Arrangement Geometry Traits

- Separates geometric aspects from topological aspects
  - Arrangement\_2<\textbf{Traits}, Dcel>—main component.
- Is a parameter of the data structures and algorithms.
  - Defines the family of curves that induce the arrangement.
  - A parameterized data structure or algorithm can be used with any family of curves for which a traits class is supplied.
- Aggregates
  - Geometric types (point, curve).
  - Operations over types (accessors, predicates, constructions).
Arrangement Geometry Traits

- Separates geometric aspects from topological aspects
  - `Arrangement_2<Traits, Dcel>`—main component.
- Is a parameter of the data structures and algorithms.
  - Defines the family of curves that induce the arrangement.
  - A parameterized data structure or algorithm can be used with any family of curves for which a traits class is supplied.
- Aggregates
  - Geometric types (point, curve).
  - Operations over types (accessors, predicates, constructions).
- Each input curve is subdivided into x-monotone subcurves.
  - Most operations involve points and x-monotone curves.
Arrangement Traits Hierarchy

The traits-concept hierarchy for arrangements induced by bounded curves.

```
ArrangementBasicTraits_2

ArrangementXMonotoneTraits_2

ArrangementTraits_2
```

Algorithmic Robotics and Motion Planning 26
ArrangementBasicTraits_2 Concept

- Types:
  - Point_2
  - X_monotone_curve_2

- Methods:
  1. Compare_x_2—compares the $x$-coordinates of 2 points.
  2. Compare_xy_2—lexicographically compares the $x$-coordinates of 2 points.
  3. Equal_2
     - Are two points represent the same geometric entity?
     - Are two $x$-monotone curves represent the same geometric entity?
  4. Construct_min_vertex—returns the lexicographically smallest endpoint of an $x$-monotone curve.
  5. Construct_max_vertex—returns the lexicographically largest endpoint of an $x$-monotone curve.
  6. Is_vertical—determines whether an $x$-monotone curve is vertical.
ArrangementBasicTraits_2 Concept (Cont.)

- **Methods:**
  7. `Compare_y_at_x_2`—determines the relative position of an \( x \)-monotone curve and a point.
  8. `Compare_y_at_x_right_2`—determines the relative position of 2 \( x \)-monotone curves to the right of a point.
  9. `Compare_y_at_x_left_2`—determines the relative position of 2 \( x \)-monotone curves to the left of a point (optional).

- **Categories:**
  - `Has_left_category`—determines whether the predicate `Compare_y_at_x_left_2` is supported.
  - Determines whether \( x \)-monotone curves may reach the corresponding boundary.
    - `Arr_left_side_category`
    - `Arr_right_side_category`
    - `Arr_bottom_side_category`
    - `Arr_top_side_category`
ArrangementXMonotoneTraits_2 Concept

Supporting intersecting bounded curves.

- **Methods:**
  1. `Split_2`—splits an $x$-monotone curve at a point into two interior disjoint subcurves.
  2. `Are_mergeable_2`—determines whether two curves can be merged into a single curve.
  3. `Merge_2`—merges two mergeable curves into a single curve.
  4. `Intersection_2`—finds all intersections of 2 $x$-monotone curves.
**ArrangementTraits_2 Concept**

Supporting arbitrary bounded curves.

- **Types:** Curve_2
- **Methods:**
  - 1. Make_x_monotone_2—subdivides a curve into x-monotone curves and isolated points.

- \((x^2 + y^2)(x^2 + y^2 - 1) = 0\)—the defining polynomial of a curve \(c\).
- \(c\) comprizes of
  - the unit circle (the locus of all points for which \(x^2 + y^2 = 1\)) and
  - the origin (the singular point \((0,0)\)).
- \(c\) is subdivided into two circular arcs and an isolated point.
Arrangement Traits Hierarchy for the Landmark Point Location

The traits-concept hierarchy for arrangements that support the Landmark Point Location strategy.

```
ArrangementBasicTraits_2

Arr...ConstructXMonotoneCurveTraits_2    Arr...ApproximateTraits_2

ArrangementLandmarkTraits_2
```
Supporting Lanmark point-location strategy.

- **Arrangement**\_\textit{ConstructXMonotoneCurveTraits\_2} Concept
  - Methods:
    - \texttt{Construct\_x\_monotone\_curve\_2}—constructs an $x$-monotone curve connecting two given points.

- **Arrangement**\_\textit{ApproximateTraits\_2} Concept
  - Methods:
    - \texttt{Approximate\_2}—approximates the $x$- and $y$-coordinates of a given point using the fixed precision number type.
The traits-concept hierarchy for arrangements induced by unbounded curves.

```
ArrangementBasicTraits_2

OpenLeftTraits  OpenRightTraits  OpenBottomTraits  OpenTopTraits

ArrangementOpenBoundaryTraits_2
```
Supporting unbounded curves.

- Methods:
  - 1. `Parameter_space_in_x_2`—determines the location of the curve end along the x-dimension.
  - 2. `Compare_y_near_boundary_2`—compares the y-coordinate of 2 curve ends near their limits.
Supporting unbounded curves.

- **Methods:**
  1. `Parameter_space_in_y_2` — Determines the location of the curve end along the y-dimension.
  2. `Compare_x_on_boundary_2` — Compare the x-coordinate of 2 curve ends at their limits.
  3. `Compare_x_near_boundary_2` — Compare the x-coordinate of 2 curve ends near their limits.
     - Precondition: the x-coordinate of the curves at their limit is equal.
Arrangement Traits Models

- Line segments:
  1. Uses the kernel point and segment types.
  2. Caches the underlying line.
- Linear curves, i.e., line segments, rays, and lines.
- Circular arcs and line segments.
- Conic curves
- Arcs of rational functions.
- Bézier curves.
- Algebraic curves of arbitrary degrees.
Traits Model: Non Caching Line Segments

- \texttt{Arr\_non\_caching\_segment\_traits\_2<Kernel>}
- Nested point type is \texttt{Kernel::Point\_2}.
- Nested curve type is \texttt{Kernel::Segment\_2}.
- Most of the defined operations are delegations of the corresponding operations of the \texttt{Kernel} type.
The Effect of Cascading of Intersections

- An arrangement induced by 4 line segments.
- The segments are inserted in the order indicated in brackets.
- The insertion creates a cascading effect of segment intersection.
- The bit-lengths of the intersection-point coords grow exponentially.
- Indiscriminate normalization considerably slows down the arrangement construction.
Traits Model: Caching Line Segments

- **Arr_segment_traits_2<Kernel>**
- Nested point type is `Kernel::Point_2` (like `Arr_segment_non_caching_traits_2`)
- A segment is represented by:
  - its two endpoints,
  - its supporting line,
  - a flag indicating whether the segment is vertical, and
  - a flag indicating whether the segment target-point is lexicographically larger than its source.
- Superior when the number of intersections is large.
Traits Model: Polycurves

- \texttt{Arr\_polycurve\_traits\_2<SubcurveTraits>}
- A \textit{polycurve} is a continuous non necessarily linear piecewise curve.
- \texttt{SubcurveTraits} must be a model of
  - \textit{ArrangementTraits\_2} and
  - \textit{ArrangementDirectionalXMonotoneTraits\_2}, and
Traits Model: Arcs of Rational Functions

- `Arr_rational_arc_traits_2<AlgKernel, NtTraits>`

**Definition (polynomial)**

A polynomial is an expression of finite length constructed from variables and constants, using only the operations of addition, subtraction, multiplication, and non-negative integer exponents.

For example: \( x^2 - 4x + 7 \)

- \( P(x), Q(x) \) — Univariate polynomials of arbitrary degrees.
- \( y = \frac{P(x)}{Q(x)} \) — A rational function.
- The coefficient are rational numbers.
- \( [x_{\min}, x_{\max}] \) is an interval over which an arc is defined \( \Rightarrow x_{\min} \) and \( x_{\max} \) can be arbitrary algebraic numbers.
- Supports arcs of rational functions.
A rational arc is always $x$-monotone.

A rational arc is not necessarily continuous.

$$y = \frac{1}{(x-1)(2-x)}$$ defined over the interval $[0, 3]$.

- Has two singularities at $x = 1$ and at $x = 2$.
- Is subdivided by Make_x_monotone_2 into 3 continuous portions defined over the intervals $[0, 1)$, $(1, 2)$, and $(2, 3]$, respectively.
An arrangement of 4 bounded rational arcs (ex_rational_functions.cpp).

An arrangement of 6 unbounded arcs of rational functions (ex_unbounded_rational_functions.cpp).
Traits Model: Algebraic Curves

\textbf{Arr\_algebraic\_segment\_traits\_2\(<\text{Coefficient}\>\)}

\textbf{Definition (Algebraic curve)}

An \textbf{algebraic curve} is the (real) zero set of a bivariate polynomial $f(x, y)$.

\begin{itemize}
  \item Supports
    \begin{itemize}
      \item algebraic curves and
      \item continuous $x$-monotone segments of algebraic curves, which are not necessarily maximal.
      \item Non $x$-monotone segments are not supported.
      \item $x$-monotone segments are not necessarily maximal.
    \end{itemize}
  \item An Oval of Cassini, $(y^2 + x^2 + 1)^2 - 4y^2 = 4/3$.
    \begin{itemize}
      \item The induced arrangement consists of 2 faces, 10 edges, and 10 vertices.
    \end{itemize}
\end{itemize}
### Arrangement Geometry Traits Models

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Curve Family</th>
<th>Degree</th>
<th>Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arr_non_caching_segment_basic_traits_2</td>
<td>line segments</td>
<td>1</td>
<td>①</td>
</tr>
<tr>
<td>Arr_non_caching_segment_traits_2</td>
<td>line segments</td>
<td>1</td>
<td>①,②,③</td>
</tr>
<tr>
<td>Arr_segment_traits_2</td>
<td>line segments</td>
<td>1</td>
<td>①,②,③</td>
</tr>
<tr>
<td>Arr_linear_traits_2</td>
<td>line segments, rays,</td>
<td>1</td>
<td>①,②,③,④</td>
</tr>
<tr>
<td></td>
<td>and lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arr_circle_segment_traits_2</td>
<td>line segments and</td>
<td>≤ 2</td>
<td>②,③</td>
</tr>
<tr>
<td></td>
<td>circular arcs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arr_circular_line_arc_traits_2</td>
<td>line segments and</td>
<td>≤ 2</td>
<td>②</td>
</tr>
<tr>
<td></td>
<td>circular arcs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arr_conic_traits_2</td>
<td>circles, ellipses, and</td>
<td>≤ 2</td>
<td>①,②,③</td>
</tr>
<tr>
<td></td>
<td>conic arcs,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arr_rational_function_traits_2</td>
<td>curves of rational</td>
<td>≤ 2</td>
<td>①,②,③,④</td>
</tr>
<tr>
<td></td>
<td>functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arr_Bezier_curve_traits_2</td>
<td>Bézier curves</td>
<td>≤ n</td>
<td>②,③</td>
</tr>
<tr>
<td>Arr_algebraic_segment_traits_2</td>
<td>algebraic curves</td>
<td>≤ n</td>
<td>②,③,④</td>
</tr>
<tr>
<td>Arr_polyline_traits_2</td>
<td>polylines</td>
<td>∞</td>
<td>①,②,③</td>
</tr>
<tr>
<td>Arr_polycurve_traits_2</td>
<td>polycurves</td>
<td>∞</td>
<td>①,②,③</td>
</tr>
<tr>
<td>Arr_geodesic_arcs_traits_2</td>
<td>geodesic arcs on</td>
<td>1</td>
<td>①,②,③</td>
</tr>
<tr>
<td></td>
<td>sphere</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

1 **CGAL 2D Arrangements**
   - Representation
   - Queries
     - Vertical Decomposition
     - Point Location Queries
   - The Zone Computation Algorithmic Framework
   - The Plane Sweep Algorithmic Framework
   - Arrangement of Unbounded Curves
   - Arrangement-Traits Classes
   - Extending the Arrangement
     - Map Overlay
     - Adapting to **Boost** Graphs
     - Arrangement on Surfaces
     - Literature
The Notification Mechanism

Definition (Observer)

An observer defines a one-to-many dependency between objects, so that when one object changes state, all its dependents are notified and updated automatically.

- The 2D Arrangements package offers a mechanism that uses observers.
- The observed type is derived from an instance of `Arr_observer<Arrangement>`.
- The observed object does not know anything about the observers.
- Each arrangement object stores a list of pointers to `Arr_observer` objects.
- The trapezoidal-RIC and the landmark point-location strategies use observers to keep their auxiliary data-structures up-to-date.
Observer Notification Functions

The set of functions can be divided into 3 categories:

1. Notifiers on changes that affect the topological structure of the arrangement. There are 2 pairs *(before and after)* that notify when
   - the arrangement is cleared or
   - the arrangement is assigned with the contents of another one.

2. Pairs of notifiers before and after of a local change that occurs in the topological structure.
   - A new vertex is constructed or deleted.
   - An new edge is constructed or deleted.
   - 1 edge is split into 2 edges, or 2 are merged into 1.
   - 1 face is split into 2 faces, or 2 are merged into 1.
   - 1 hole is created in the interior of a face or removed from it.
   - 2 holes are merged into 1, or 1 is split into 2.
   - A hole is moved from one face to another.

3. Notifiers on a structural change caused by a free function. A single pair *(before_global_change() and after_global_change()*.
Extending all the **DCEL** Records

- An instance of

  \[
  \text{Arr\_extended\_dcel}\langle \text{Traits}\!, \text{VertexData}\!, \text{HalfedgeData}\!, \text{FaceData}\rangle
  \]

  is a **DCEL** that extends the vertex, halfedge, and face records with the corresponding types.

```cpp
enum Color {BLUE, RED, WHITE};

typedef CGAL::Arr_extended_dcel<Traits_2, Color, bool, unsigned int> Dcel;
typedef CGAL::Arrangement_2<Traits_2, Dcel> Ex_arrangement_2;
```

```cpp
Ex_arrangement_2::Vertex_iterator vit;
for (vit = arr.vertices_begin(); vit != arr.vertices_end(); ++vit) {
    unsigned int degree = vit->degree();
    vit->set_data((degree == 0) ? BLUE : ((degree <= 2) ? RED : WHITE));
}
```

```cpp
std::cout << "The arrangement vertices:
for (vit = arr.vertices_begin(); vit != arr.vertices_end(); ++vit) {
    std::cout << '(' << vit->point() << ")\nswitch (vit->data()) {
    case BLUE : std::cout << "BLUE.\n    case RED : std::cout << "RED.\n    case WHITE : std::cout << "WHITE.\n
```
#include <string>
#include <CGAL/basic.h>
#include <CGAL/Arr_dcel_base.h>

/* The map extended dcel vertex */
template <typename Point_2>
class Arr_map_vertex : public CGAL::Arr_vertex_base<Point_2> {
  public:
    std::string name, type;
};

/* The map extended dcel halfedge */
template <typename X_monotone_curve_2>
class Arr_map_halfedge : public CGAL::Arr_halfedge_base<X_monotone_curve_2> {
  public:
    std::string name, type;
};

/* The map extended dcel face */
class Arr_map_face : public CGAL::Arr_face_base {
  public:
    std::string name, type;
};

/* The map extended dcel */
template <typename Traits>
class Arr_map_dcel : public
  CGAL::Arr_dcel_base<Arr_map_vertex<typename Traits::Point_2>,
  Arr_map_halfedge<typename Traits::X_monotone_curve_2>,
  Arr_map_face>
{ };
Outline

1 Cgal 2D Arrangements
   ● Representation
   ● Queries
     ● Vertical Decomposition
     ● Point Location Queries
   ● The Zone Computation Algorithmic Framework
   ● The Plane Sweep Algorithmic Framework
   ● Arrangement of Unbounded Curves
   ● Arrangement-Traits Classes
   ● Extending the Arrangement
   ● Map Overlay
     ● Adapting to Boost Graphs
     ● Arrangement on Surfaces
     ● Literature
Outline

1. **Cgal 2D Arrangements**
   - Representation
   - Queries
     - Vertical Decomposition
     - Point Location Queries
   - The Zone Computation Algorithmic Framework
   - The Plane Sweep Algorithmic Framework
   - Arrangement of Unbounded Curves
   - Arrangement-Traits Classes
   - Extending the Arrangement
   - Map Overlay
   - **Adapting to Boost Graphs**
   - Arrangement on Surfaces
   - Literature
Outline

1 Cgal 2D Arrangements
   • Representation
   • Queries
     ● Vertical Decomposition
     ● Point Location Queries
   • The Zone Computation Algorithmic Framework
   • The Plane Sweep Algorithmic Framework
   • Arrangement of Unbounded Curves
   • Arrangement-Traits Classes
   • Extending the Arrangement
   • Map Overlay
   • Adapting to Boost Graphs
   • Arrangement on Surfaces
   • Literature
Outline

1 Cgal 2D Arrangements

- Representation
- Queries
  - Vertical Decomposition
  - Point Location Queries
- The Zone Computation Algorithmic Framework
- The Plane Sweep Algorithmic Framework
- Arrangement of Unbounded Curves
- Arrangement-Traits Classes
- Extending the Arrangement
- Map Overlay
- Adapting to Boost Graphs
- Arrangement on Surfaces
- Literature
Boris Aronov and Dmitriy Drusvyatskiy
Complexity of a Single Face in an Arrangement of s-Intersecting Curves

Jon Louis Bentley and Thomas Ottmann.
Algorithms for Reporting and Counting Geometric Intersections.

Eric Berberich, Efi Fogel, Dan Halperin, Michael Kerber, and Ophir Setter.
Arrangements on parametric surfaces ii: Concretizations and applications, 2009.

Ulrich Finke and Klaus H. Hinrichs.
Overlaying simply connected planar subdivisions in linear time.

Ron Wein, Efi Fogel, Baruch Zukerman, Dan Halperin, and Eric Berberich.
2D Arrangements.

David G. Kirkpatrick.
Optimal search in planar subdivisions.

N. Sarnak and Robert E. Tarjan.
Planar point location using persistent search trees.

Kentan Mulmuley.
A fast planar partition algorithm, I.
Arrangement Bibliography II

- Raimund Seidel
  A Simple and Fast Incremental Randomized Algorithm for Computing Trapezoidal Decompositions and for Triangulating Polygons.

- Olivier Devillers, Sylvain Pion, and Monique Teillaud.
  Walking in a triangulation.

- Luc Devroye Christophe, Christophe Lemaire, and Jean-Michel Moreau.
  Fast Delaunay Point-Location with Search Structures.

- Luc Devroye, Ernst Peter Mücke, and Binhai Zhu.
  A Note on Point Location in Delaunay Triangulations of Random Points.

- Olivier Devillers.
  The Delaunay hierarchy.

- Sunil Arya
  A Simple Entropy-Based Algorithm for Planar Point Location.
  *ACM Transactions on Graphics*. 3(2), 2007

- Masato Edahiro, Iwao Kokubo, And Takao Asano
  A new Point-Location Algorithm and its Practical Efficiency: comparison with existing algorithms

- Micha Sharir and Pankaj Kumar Agarwal
  *Davenport-Schinzel Sequences and Their Geometric Applications*.
Arrangement Bibliography III

Bernard Chazelle, Leonidas J. Guibas, and Der-Tsai Le.
The Power of Geometric Duality.

Herbert Edelsbrunner,
*Algorithms in Combinatorial Geometry*,

Mark de Berg, Mark van Kreveld, Mark H. Overmars, and Otfried Cheong.
*Computational Geometry: Algorithms and Applications*.

Herbert Edelsbrunner, Raimund Seidel, and Micha Sharir.
On the Zone Theorem for Hyperplane Arrangements.

Silvio Micali and Vijay V. Vazirani.
An $O(\sqrt{|V||E|})$ Algorithm for Finding Maximum Matching in General Graphs.

Jack Edmonds.
Paths, Trees, and Flowers.

Robert Endre Tarjan.

Marcin Mucha and Piotr Sankowski.
Maximum Matchings via Gaussian Elimination
Arrangement Bibliography IV

Jeremy G. Siek, Lie-Quan Lee, and Andrew Lumsdaine.  
*The Boost Graph Library.*  
Addison-Wesley, 2002

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