Voronoi Diagrams
The Post Office Problem

Mohammad Amin Solhizadeh
amin.solhizadeh@gmail.com

Yazd University

Computer Science 2010
Outline

1. Definition and Basic Properties
2. Computing the Voronoi Diagram
3. Voronoi Diagrams of Line Segments
4. Farthest-Point Voronoi Diagrams
**Definition**

- Voronoi Assignment Model: The model where every point is assigned to the nearest site.
Introduction

**Definition**

- Voronoi Assignment Model: The model where every point is assigned to the nearest site
- Voronoi Diagram: The subdivision induced by Voronoi Assignment Model
**Introduction**

### Definition
- **Voronoi Assignment Model:** The model where every point is assigned to the nearest site.
- **Voronoi Diagram:** The subdivision induced by Voronoi Assignment Model.

### Usage
- Social Geography
- Physics
- Astronomy
- Robotics
- Delaunay Triangulation
Outline

1. **Definition and Basic Properties**

2. **Computing the Voronoi Diagram**

3. **Voronoi Diagrams of Line Segments**

4. **Farthest-Point Voronoi Diagrams**
Primary Definitions

Reminder

Denote the Euclidean distance between two points \( p \) and \( q \) by \( \text{dist}(p, q) \):

\[
\text{dist}(p, q) := \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}
\]
Primary Definitions

Reminder
Denote the Euclidean distance between two points \( p \) and \( q \) by \( \text{dist}(p, q) \):

\[
\text{dist}(p, q) := \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}
\]

Voronoi Diagram (\( \text{Vor}(P) \))
- Let \( P = p_1, p_2, \ldots, p_n \) be a set of \( n \) distinct points (sites)
- Dividing plane into \( n \) cells according to following rule:

\[
q \in p_i \quad \text{if and only if} \quad \text{dist}(q, p_i) < \text{dist}(q, p_j) \quad \forall p_j \in P, \ j \neq i
\]
Voronoi Cell and Bisector

**Voronoi Cell of** $p_i$

The cell of $Vor(P)$ that correspond to a site $p_i$ is denoted $V(p_i)$
Voronoi Cell and Bisector

Voronoi Cell of $p_i$

The cell of $Vor(P)$ that correspond to a site $p_i$ is denoted $V(p_i)$

Bisector of $p$ and $q$

- The perpendicular of the line segment $pq$
- $h(p, q)$: Open half-plane that contains $p$
- $h(q, p)$: Open half-plane that contains $q$
- $r \in h(p, q)$ if and only if $\text{dist}(r, p) < \text{dist}(r, q)$
Observation 7.1

$$V(p_i) = \bigcap_{1 \leq j \leq n, j \neq i} h(p_i, p_j)$$
Observation 7.1

$$V(p_i) = \cap_{1 \leq j \leq n, \ j \neq i} h(p_i, p_j)$$

Result

The Voronoi Diagram is a planar subdivision whose edges are **straight**.
**Theorem 7.2**

Let $P$ be a set of $n$ point sites in the plane. If all the sites are **collinear** then $\text{Vor}(P)$ consists of $n - 1$ parallel lines. Otherwise, $\text{Vor}(P)$ is **connected** and its edges are either **segments** or **half-lines**.
**Theorem 7.2**

Let $P$ be a set of $n$ point sites in the plane. If all the sites are collinear then $Vor(P)$ consists of $n - 1$ parallel lines. Otherwise, $Vor(P)$ is connected and its edges are either segments or half-lines.

**Proof**

- First part is easy.
Theorem 7.2

Let $P$ be a set of $n$ point sites in the plane. If all the sites are **collinear** then $Vor(P)$ consists of $n - 1$ parallel lines. Otherwise, $Vor(P)$ is **connected** and its edges are either **segments** or **half-lines**.

**Proof**

- First part is easy
- The edges of $Vor(P)$ are either segments or half-lines
**Theorem 7.2**

Let $P$ be a set of $n$ point sites in the plane. If all the sites are collinear then $\text{Vor}(P)$ consists of $n - 1$ parallel lines. Otherwise, $\text{Vor}(P)$ is connected and its edges are either segments or half-lines.

**Proof**

- First part is easy
- The edges of $\text{Vor}(P)$ are either segments or half-lines
- $\text{Vor}(P)$ is connected
No. of Edges and Vertices

**Theorem 7.3**

For $n \geq 3$, the number of vertices in the Voronoi diagram of a set of $n$ point sites in the plane is at most $2n - 5$ and the number of edges is at most $3n - 6$. 
**Theorem 7.3**

For $n \geq 3$, the number of vertices in the Voronoi diagram of a set of $n$ point sites in the plane is at most $2n - 5$ and the number of edges is at most $3n - 6$.

**Proof (Part I)**

Sites are all collinear.
Proof (Part II)

\[ m_v - m_e + m_f = 2 \]
No. of Edges and Vertices (Cont.)

Proof (Part II)

- \( m_v - m_e + m_f = 2 \)
- \( (n_v + 1) - n_e + n = 2 \)
No. of Edges and Vertices (Cont.)

Proof (Part II)

- $m_v - m_e + m_f = 2$
- $(n_v + 1) - n_e + n = 2$
- $2n_e \geq 3(n_v + 1)$
**Definition** ($C_P(q)$)

Largest empty circle of $q$ with respect to $P$:
The largest circle with $q$ as its center that does not contain any site of $P$ in its interior
Theorem 7.4

For the Voronoi diagram \( \text{Vor}(P) \) of a set of points \( P \) the following holds:

1. A point \( q \) is a vertex of \( \text{Vor}(P) \) if and only if its largest empty circle \( C_P(q) \) contains three or more sites on its boundary.
2. The bisector between sites \( p_i \) and \( p_j \) defines an edge of \( \text{Vor}(P) \) if and only if there is a point \( q \) on the bisector such that \( C_P(q) \) contains both \( p_i \) and \( p_j \) on its boundary but no other site.
Theorem 7.4

For the Voronoi diagram $Vor(P)$ of a set of points $P$ the following holds:

1. A point $q$ is a vertex of $Vor(P)$ if and only if its largest empty circle $C_P(q)$ contains three or more sites on its boundary.
Knowing Voronoi Diagram

Theorem 7.4

For the Voronoi diagram $\text{Vor}(P)$ of a set of points $P$ the following holds:

1. A point $q$ is a vertex of $\text{Vor}(P)$ if and only if its largest empty circle $C_P(q)$ contains three or more sites on its boundary.

2. The bisector between sites $p_i$ and $p_j$ defines an edge of $\text{Vor}(P)$ if and only if there is a point $q$ on the bisector such that $C_P(q)$ contains both $p_i$ and $p_j$ on its boundary but no other site.
Proof

Part I

If

Knowing Voronoi Diagram (Cont.)
Knowing Voronoi Diagram (Cont.)

Proof

Part 1
- If
- Only If
Knowing Voronoi Diagram (Cont.)

Proof

1. Part I
   - If
   - Only If
2. Part II
   - If
Knowing Voronoi Diagram (Cont.)

Proof

1. Part I
   - If
   - Only If

2. Part II
   - If
   - Only If
Outline

1. Definition and Basic Properties
2. Computing the Voronoi Diagram
3. Voronoi Diagrams of Line Segments
4. Farthest-Point Voronoi Diagrams
Different Ways

Ways of Computing Voroni Diagram
Different Ways

Ways of Computing Voronoi Diagram

- Simple way $\implies$ Complexity: $O(n^2 \lg n)$
Different Ways

Ways of Computing Voroni Diagram

1. Simple way $\implies$ Complexity: $O(n^2 \log n)$
2. Fortune’s Algorithm $\implies$ Complexity: $O(n \log n)$
**Different Ways**

### Ways of Computing Voronoi Diagram

1. **Simple way** ➞ Complexity: $O(n^2 \lg n)$
2. **Fortune’s Algorithm** ➞ Complexity: $O(n \lg n)$

### Why Fortune’s Algorithm is Optimal?

- Problem of sorting $n$ real numbers is reducible to the problem of computing Voronoi Diagram
Different Ways

Ways of Computing Voronoi Diagram

1. Simple way $\implies$ Complexity: $O(n^2 \log n)$
2. Fortune’s Algorithm $\implies$ Complexity: $O(n \log n)$

Why Fortune’s Algorithm is Optimal?

- Problem of sorting $n$ real numbers is reducible to the problem of computing Voronoi Diagram
- So Voronoi Diagram must take $\Omega(n \log n)$ time in the worst case
Overview

We use plane sweep strategy in Fortune’s Algorithm
Overview of Fortune’s Algorithm

Overview
We use plane sweep strategy in Fortune’s Algorithm

Problem
The part of $Vor(P)$ above $l$ depends not only on the sites that lie above $l$, but also on sites below $l$. 
Overview of Fortune’s Algorithm (Cont.)

Solution

- $l^+$: the closed half-plane above $l$
Overview of Fortune’s Algorithm (Cont.)

Solution

- $l^+$: the closed half-plane above $l$
- For which points $q \in l^+$ do we know for sure what their nearest site is?
Overview of Fortune’s Algorithm (Cont.)

Solution

- \( l^+ \): the closed half-plane above \( l \)
- For which points \( q \in l^+ \) do we know for sure what their nearest site is?
- The nearest site of \( q \) can not lie below \( l \) if \( q \) is at least as near to some site \( p_i \in l^+ \) as \( q \) is to \( l \)
The locus of points that are closer to any site above \( l \) than to \( l \) itself is bounded by Parabolic Arcs. We call this sequence of arcs, the beach line.
Observation 7.5

The beach line is x-monotone, that is, every vertical line intersects it in exactly one point.
Observation 7.5

The beach line is x-monotone, that is, every vertical line intersects it in exactly one point.

Note

- Breakpoints between the different parabolic arcs forming the beach line lie on edges of Voronoi Diagram.
**Observation 7.5**

The beach line is $x$-monotone, that is, every vertical line intersects it in exactly one point.

**Note**

- Breakpoints between the different parabolic arcs forming the beach line lie on edges of Voronoi Diagram.
- Breakpoints, exactly trace out the Voronoi Diagram.
New Arc

When a New Arc Appears on the Beach Line?

1. A degenerate parabola with zero width
2. Sweep line reaches a new site
3. Site event: where a new site is encountered
   - Sweep line moves downward
   - New parabola gets wider and wider

Diagrams:
1. Initial parabola
2. Parabola widens
3. New arc appears
New Arc

When a New Arc Appears on the Beach Line?

- One Occasion: When the sweep line \( l \) reaches a new site

\[ \begin{align*}
\text{Site Event} & \\
& \quad \text{Where a new site is encountered}
\end{align*} \]
**New Arc**

**When a New Arc Appears on the Beach Line?**

- One Occasion: When the sweep line $l$ reaches a new site
  - A degenerate parabola with zero width
New Arc

When a New Arc Appears on the Beach Line?
- One Occasion: When the sweep line $\ell$ reaches a new site
  1. A degenerate parabola with zero width
  2. Sweep Line moves downward $\Rightarrow$ New parabola gets wider and wider
New Arc

When a New Arc Appears on the Beach Line?

- One Occasion: When the sweep line \( l \) reaches a new site
  - A degenerate parabola with zero width
  - Sweep Line moves downward \( \Rightarrow \) New parabola gets wider and wider

Site Event

Where a new site is encountered
What Happens to the Voronoi Diagram at Site Event?

1. The two new breakpoints coincide at first.
2. Then move in opposite directions to trace out the same edge.

Note: Initially, this edge is not connected to the rest of the Voronoi Diagram above the sweep line.
What Happens to the Voronoi Diagram at Site Event?

1. The two new breakpoints coincide at first.
What Happens to the Voronoi Diagram at Site Event?

1. The two new breakpoints coincide at first
2. Then move in opposite directions to trace out the same edge
What Happens to the Voronoi Diagram at Site Event?

1. The two new breakpoints coincide at first
2. Then move in opposite directions to trace out the same edge

Note

Initially, this edge is not connected to the rest of Voronoi Diagram above the sweep line
New Arc Lemma

**Lemma 7.6**
The only way in which the new arc can appear on the beach line is through a site event.
**Lemma 7.6**

The only way in which the new arc can appear on the beach line is through a site event.

**Proof**

(Contradiction) An already existing parabola $\beta_j$ defined by a site $p_j$ breaks through the beach line.
There are two ways in which this could happen:
Proof (Part I)

- $\beta_j$ breaks through in the middle of an arc of a parabola $\beta_j$. 

\[
\beta_j : y = \frac{1}{2}(p_j, y - l_y)(x^2 - 2p_jx + p_j^2, x^2 + 2p_jx - 2l_y) 
\]

Result

$p_i, y$ and $p_j, y$ are larger than $l_y \Rightarrow$ It is impossible that $\beta_i, \beta_j$ have only one point of intersect.
Proof (Part I)

- $\beta_j$ breaks through in the **middle** of an arc of a parabola $\beta_j$
- $l_y$: y-coordinate of sweep line (at the moment of tangency)
- $p_j := (p_{j,x}, p_{j,y})$
**Proof (Part I)**

- $\beta_j$ breaks through in the **middle** of an arc of a parabola $\beta_j$
- $l_y$: $y$-coordinate of sweep line (at the moment of tangency)
- $p_j := (p_{j,x}, p_{j,y})$
- $\beta_j :=$
- \[
y = \frac{1}{2(p_{j,y}-l_y)} \left( x^2 - 2p_{j,x}x + p_{j,x}^2 + p_{j,y}^2 - l_y^2 \right)
\]
New Arc Lemma (Cont.)

Proof (Part I)

- \( \beta_j \) breaks through in the middle of an arc of a parabola \( \beta_j \)
- \( l_y \): y-coordinate of sweep line (at the moment of tangency)
- \( p_j := (p_{j,x}, p_{j,y}) \)
- \( \beta_j := y = \frac{1}{2(p_{j,y} - l_y)}(x^2 - 2p_{j,x}x + p_{j,x}^2 + p_{j,y}^2 - l_y^2) \)

Result

\( p_{i,y} \) and \( p_{j,y} \) are larger than \( l_y \) \( \implies \)
It is impossible that \( \beta_i, \beta_j \) have only one point of intersect
New Arc Lemma (Cont.)

Proof (Part II)

$\beta_j$ appears in between two arcs ($\beta_i$, $\beta_k$)
New Arc Lemma (Cont.)

Proof (Part II) (Cont.)

- Consider an infinitesimal motion of sweep line downward while keeping the circle $C$ tangent to $l$.
Proof (Part II) (Cont.)

- Consider an infinitesimal motion of sweep line downward while keeping the circle $C$ tangent to $l$
- Either $p_i$ or $p_k$ will penetrate the interior
**New Arc Lemma (Cont.)**

**Proof (Part II) (Cont.)**

- Consider an infinitesimal motion of sweep line downward while keeping the circle $C$ tangent to $l$.
- Either $p_i$ or $p_k$ will penetrate the interior.

**Result**

In a sufficiently small neighborhood of $q$ the parabola $\beta_j$ can not appear on the beach line when the sweep line moves downward.

![Diagram](image-url)
# Circle Event

**Assumption**

- $\alpha'$: Disappearing arc
- $\alpha, \alpha''$: Two neighboring arcs of $\alpha'$
- $q$: Is equidistant from $l$ and each of three sites

**Circle Event**

Where the sweep line reaches the lowest point of a circle through three sites defining consecutive arcs on the beach line.
Circle Event Lemma

Lemma 7.7

The only way in which an existing arc can disappear from the beach line is through a circle event.
**Events**

1. **Site event**
   - New arc appears
   - New edge starts to grow

2. **Circle event**
   - Existing arcs drops out
   - Two growing edges meet to form a vertex

**Question**

What else do we need?
Voronoi Diagram

- Data Structure: Doubly-connected edge list
- Is there any problem?
**Beach Line**
- Data Structure: Balanced binary search tree $T$; it is the status structure
- Leaves of $T$: Arcs
- Internal nodes of $T$: Breakpoints

**Event Queue**
Data Structure: Priority queue (Priority of an event is its y-coordinate)
Circle Event Detection

Event Cases

1. Consecutive triples whose two breakpoints do not cover
2. Circle event can disappear before the event has taken place

Question

What does Circle Event Algorithm do?
Circle Event Detection (Cont.)
Circle Event Detection (Cont.)

Voronoi Diagrams, The Post Office Problem
Circle Event Detection (Cont.)

When Circle Event Occurs?
A circle event occurs if the sweep line reaches the bottom of an empty circle defined by three sites that have consecutive parabolic arcs on the beach line.

What Should We Do Now!
We will make sure that any three sites that have consecutive arcs on the beach line and whose circle has its lowest point below the sweep line have this lowest point as circle event in the event list.
Circle Event Detection (Cont.)

Circle Event in Status Structure

In the status structure we can see all triples of consecutive parabolic arcs that can give circle events.

```
p1 p2 p3 p4
⟨p2,p3⟩
⟨p1,p2⟩
⟨p3,p4⟩
⟨p4,p5⟩
⟨p5,p6⟩

p5

p6
```

```
P1 P2 P3 P4
⟨P2,P3⟩
⟨P1,P2⟩
⟨P3,P4⟩
⟨P4,P5⟩
⟨P5,P6⟩
```

```
P1
P2
P3
P4
```
Are All Circle Events Correct?

We may have stored a circle event in the event list, but it may be that it never happens . . .
This is called a False Alarm

Reasons

There are two reasons for false alarms:

1. Site events
2. Other circle events
False Alarm Reason 1

Status (Before)

A site event that disrupts three consecutive parabolic arcs
False Alarm Reason 1 (Cont.)

The circle of a circle event may turn out not to be empty.
False Alarm Reason 1 (Cont.)

Status (After)

A site event that disrupts three consecutive parabolic arcs

\[
\langle p_1, p_2 \rangle \quad \langle p_2, p_3 \rangle \quad \langle p_3, p_4 \rangle \\
\langle p_4, p_5 \rangle \quad \langle p_7, p_3 \rangle \\
\langle p_5, p_6 \rangle
\]

\[ c(p_2, p_3, p_4) \text{ gone} \]
**False Alarm Reason 2**

**Status (Before)**

A circle event that disrupts three consecutive parabolic arcs

![Diagram of Voronoi diagrams]

- Circle event $C(p_2, p_3, p_4)$
- Circle event $C(p_1, p_2, p_3)$
- $\langle p_4, p_5 \rangle$
- $\langle p_2, p_3 \rangle$
- $\langle p_5, p_6 \rangle$
- $\langle p_1, p_2 \rangle$
- $\langle p_3, p_4 \rangle$
- $p_5$
- $p_6$
- $p_1$
- $p_2$
- $p_3$
- $p_4$
False Alarm Reason 2 (Cont.)

\[ C(p_1, p_2, p_3) \]
\[ \alpha_3 \text{ will disappear} \]

\[ C(p_2, p_3, p_4) \]
False Alarm Reason 2 (Cont.)

Status (After)

A circle event that disrupts three consecutive parabolic arcs

\[
\langle p_1, p_2 \rangle \ \langle p_2, p_4 \rangle \ \langle p_4, p_5 \rangle
\]

\[
\langle p_5, p_6 \rangle
\]

\[
\langle p_5, p_6 \rangle
\]

\[
\langle p_1, p_2 \rangle \ \langle p_2, p_4 \rangle \ \langle p_4, p_5 \rangle \ \langle p_5, p_6 \rangle
\]

\[
p_1 \ \ p_2 \ \ p_4
\]

\[
p_5 \ \ p_6
\]
**Lemma 7.8**

Every Voronoi vertex is detected by means of a circle event

**Proof**

We must show that just before the sweep line reaches the lowest point of \( C(p_i, p_j, p_k) \), there are three consecutive arcs \( \alpha, \alpha', \alpha'' \) on the beach line defined by \( p_i, p_j \) and \( p_k \)
Algorithm VORONOI\ DIAGRAM(P)

Input. A set \( P := \{p_1, \ldots, p_n\} \) of point sites in the plane.
Output. The Voronoi diagram \( \text{Vor}(P) \) given inside a bounding box in a doubly-connected edge list \( \mathcal{D} \).

1. Initialize the event queue \( \mathcal{Q} \) with all site events, initialize an empty status structure \( \mathcal{T} \) and an empty doubly-connected edge list \( \mathcal{D} \).
2. \textbf{while} \( \mathcal{Q} \) is not empty
3. \hspace{1em} \textbf{do} Remove the event with largest \( y \)-coordinate from \( \mathcal{Q} \).
4. \hspace{2em} \textbf{if} the event is a site event, occurring at site \( p_i \)
5. \hspace{3em} \textbf{then} HANDLE\ SITE\ EVENT(\( p_i \))
6. \hspace{2em} \textbf{else} HANDLE\ CIRCLE\ EVENT(\( \gamma \)), where \( \gamma \) is the leaf of \( \mathcal{T} \) representing the arc that will disappear
7. The internal nodes still present in \( \mathcal{T} \) correspond to the half-infinite edges of the Voronoi diagram. Compute a bounding box that contains all vertices of the Voronoi diagram in its interior, and attach the half-infinite edges to the bounding box by updating the doubly-connected edge list appropriately.
8. Traverse the half-edges of the doubly-connected edge list to add the cell records and the pointers to and from them.
**Voronoi Diagram Algorithm (Cont.)**

**HANDLE SITE EVENT**($p_i$)
1. If $T$ is empty, insert $p_i$ into it (so that $T$ consists of a single leaf storing $p_i$) and return. Otherwise, continue with steps 2–5.
2. Search in $T$ for the arc $\alpha$ vertically above $p_i$. If the leaf representing $\alpha$ has a pointer to a circle event in $Q$, then this circle event is a false alarm and it must be deleted from $Q$.
3. Replace the leaf of $T$ that represents $\alpha$ with a subtree having three leaves. The middle leaf stores the new site $p_i$ and the other two leaves store the site $p_j$ that was originally stored with $\alpha$. Store the tuples $\langle p_j, p_i \rangle$ and $\langle p_i, p_j \rangle$ representing the new breakpoints at the two new internal nodes. Perform rebalancing operations on $T$ if necessary.
4. Create new half-edge records in the Voronoi diagram structure for the edge separating $\mathcal{V}(p_i)$ and $\mathcal{V}(p_j)$, which will be traced out by the two new breakpoints.
5. Check the triple of consecutive arcs where the new arc for $p_i$ is the left arc to see if the breakpoints converge. If so, insert the circle event into $Q$ and add pointers between the node in $T$ and the node in $Q$. Do the same for the triple where the new arc is the right arc.
**Voronoi Diagram Algorithm (Cont.)**

**HANDLE CIRCLE EVENT (γ)**

1. Delete the leaf γ that represents the disappearing arc α from T. Update the tuples representing the breakpoints at the internal nodes. Perform rebalancing operations on T if necessary. Delete all circle events involving α from Q; these can be found using the pointers from the predecessor and the successor of γ in T. (The circle event where α is the middle arc is currently being handled, and has already been deleted from Q.)

2. Add the center of the circle causing the event as a vertex record to the doubly-connected edge list D storing the Voronoi diagram under construction. Create two half-edge records corresponding to the new breakpoint of the beach line. Set the pointers between them appropriately. Attach the three new records to the half-edge records that end at the vertex.

3. Check the new triple of consecutive arcs that has the former left neighbor of α as its middle arc to see if the two breakpoints of the triple converge. If so, insert the corresponding circle event into Q, and set pointers between the new circle event in Q and the corresponding leaf of T. Do the same for the triple where the former right neighbor is the middle arc.
Lemma 7.9

The algorithm runs in $O(n \lg n)$ time and it uses $O(n)$ storage.
Degenerate Cases

Case 1 (Collinear Events)

1. Two or more events lie on a common horizontal line
2. Several coincide circle events, when there are four or more co-circular sites

Case 2

When a site $p_i$ happens to be located exactly below the break point between two arcs on the beach line
Theorem 7.10

The Voronoi diagram of a set of \( n \) point sites in the plane can be computed with a sweep line algorithm in \( O(n \lg n) \) time using \( O(n) \) storage.
Outline

1. Definition and Basic Properties
2. Computing the Voronoi Diagram
3. Voronoi Diagrams of Line Segments
4. Farthest-Point Voronoi Diagrams
For a Voronoi diagram of other objects than point sites, we must decide to which point on each site we measure the distance . . .

This will be the closest point on the site.
Overview (Cont.)

Voronoi diagrams of line segments
Farthest-point Voronoi diagrams
Motion planning for a disc
Geometry
Plane sweep algorithm
Voronoi diagram of line segments
Computational Geometry Lecture 13: More on Voronoi diagrams

Voronoi Diagrams, The Post Office Problem
Overview (Cont.)

Voronoi diagrams of line segments
Farthest-point Voronoi diagrams
Motion planning for a disc
Geometry
Plane sweep algorithm
Voronoi diagram of line segments
Computational Geometry
Lecture 13: More on Voronoi diagrams
**Equal Distance (Sample)**

**Types**

1. The points of equal distance to two points lie on a line
2. The points of equal distance to two lines lie on a line (two lines)
3. The points of equal distance to a point and a line lie on a parabola
Bisector of Two Line Segments

Note

Two line segment sites have a bisector with up to 7 parts.
**Problem**

If two line segment sites share an endpoint, their bisector can have an area too.
Bisector of Two Line Segments (Cont.)

Solution

We assume that the line segment sites are fully disjoint, to avoid complications.

We could shorten each line segment from a set of non-crossing line segments a tiny amount.
Where Are Voronoi Vertices?

The Voronoi diagram has vertices at the centers of empty circles:

- Touching three different line segment sites (degree 3 vertex)
- Touching two line segment sites, one of which it touches in an endpoint of the line segment, and the segment is also part of the tangent line of the circle at that point (degree 2 vertex)

Note

At a degree 2 Voronoi vertex, one incident arc is a straight edge and the other one is a parabolic arc
Definition and Basic Properties
Computing the Voronoi Diagram
Voronoi Diagrams of Line Segments
Farthest-Point Voronoi Diagrams

Voronoi Vertices

Voronoi diagrams of line segments
Farthest-point Voronoi diagrams
Motion planning for a disc
Geometry Plane sweep algorithm
Empty circles

Computational Geometry Lecture 13: More on Voronoi diagrams
**Meaning**

Breakpoints trace arcs of equal distance to two different sites, or they trace segments perpendicular to a line segment starting at one of its endpoints, or they trace site interiors.
Breakpoints (Cont.)

Breakpoint Types

The algorithm uses 5 types of breakpoint:

1. If a point $p$ is closest to two site endpoints while being equidistant from them and $l$, then $p$ is a breakpoint that traces a line segment (as in the point site case).

2. If a point $p$ is closest to two site interiors while being equidistant from them and $l$, then $p$ is a breakpoint that traces a line segment.

3. If a point $p$ is closest to a site endpoint and a site interior of different sites while being equidistant from them and $l$, then $p$ is a breakpoint that traces a parabolic arc.
Breakpoints (Cont.)

**Breakpoint Types (Cont.)**

4. If a point $p$ is closest to a site endpoint, the shortest distance is realized by a segment that is perpendicular to the line segment site, and $p$ has the same distance from $l$, then $p$ is a breakpoint that traces a line segment.

5. If a site interior intersects the sweep line, then the intersection is a breakpoint that traces a line segment (the site interior).

**Note**

These two types of breakpoint do not trace Voronoi diagram edges but they do trace breaks in the beach line.
Breakpoints (Cont.)

Voronoi diagrams of line segments
Farthest-point Voronoi diagrams
Motion planning for a disc
Computational Geometry
Lecture 13: More on Voronoi diagrams
Events

Types

There are site events and circle events, but circle events come in different types.

Note

The types of circle events essentially correspond to the types of breakpoints that meet.

Not all types of breakpoint can meet.
Events (Cont.)
Result

**Theorem 7.11**

The Voronoi diagram of a set of $n$ disjoint line segment sites can be computed in $O(n \lg n)$ time using $O(n)$ storage.
Retraction
Retraction (Cont.)

Voronoi diagrams of line segments
Farthest-point Voronoi diagrams
Motion planning for a disc
Geometry Plane sweep algorithm
Computational Geometry Lecture 13: More on Voronoi diagrams
Retraction (Cont.)

Voronoi diagrams of line segments
Farthest-point Voronoi diagrams
Motion planning for a disc
Geometry Plane sweep algorithm
Computational Geometry Lecture 13: More on Voronoi diagrams
Algorithm RETRACTION(S, q\textsubscript{start}, q\textsubscript{end}, r)

Input. A set $S := \{s_1, \ldots, s_n\}$ of disjoint line segments in the plane, and two discs $D_{\text{start}}$ and $D_{\text{end}}$ centered at $q_{\text{start}}$ and $q_{\text{end}}$ with radius $r$. The two disc positions do not intersect any line segment of $S$.

Output. A path that connects $q_{\text{start}}$ to $q_{\text{end}}$ such that no disc of radius $r$ with its center on the path intersects any line segment of $S$. If no such path exists, this is reported.

1. Compute the Voronoi diagram $\text{Vor}(S)$ of $S$ inside a sufficiently large bounding box.
2. Locate the cells of $\text{Vor}(P)$ that contain $q_{\text{start}}$ and $q_{\text{end}}$.
3. Determine the point $p_{\text{start}}$ on $\text{Vor}(S)$ by moving $q_{\text{start}}$ away from the nearest line segment in $S$. Similarly, determine the point $p_{\text{end}}$ on $\text{Vor}(S)$ by moving $q_{\text{end}}$ away from the nearest line segment in $S$. Add $p_{\text{start}}$ and $p_{\text{end}}$ as vertices to $\text{Vor}(S)$, splitting the arcs on which they lie into two.
4. Let $G$ be the graph corresponding to the vertices and edges of the Voronoi diagram. Remove all edges from $G$ for which the smallest distance to the nearest sites is smaller than or equal to $r$.
5. Determine with depth-first search whether a path exists from $p_{\text{start}}$ to $p_{\text{end}}$ in $G$. If so, report the line segment from $q_{\text{start}}$ to $p_{\text{start}}$, the path in $G$ from $p_{\text{start}}$ to $p_{\text{end}}$, and the line segment from $p_{\text{end}}$ to $q_{\text{end}}$ as the path. Otherwise, report that no path exists.
Theorem 7.12

Given $n$ disjoint line segment obstacles and a disc-shaped robot, the existence of a collision-free path between two positions of the robot can be determined in $O(n \lg n)$ time using $O(n)$ storage.
Outline

1. Definition and Basic Properties
2. Computing the Voronoi Diagram
3. Voronoi Diagrams of Line Segments
4. Farthest-Point Voronoi Diagrams
Roundness Criterion

The roundness of a set of points is the width of the smallest annulus that contains the points.

An annulus is the region between two co-centric circles. Its width is the difference in radius.

Computational Geometry Lecture 13: More on Voronoi diagrams
Smallest-Width Annulus

Cases

1. $C_{outer}$ contains at least three points of $P$, and $C_{inner}$ contains at least one point of $P$

2. $C_{outer}$ contains at least one point of $P$, and $C_{inner}$ contains at least three points of $P$

3. $C_{outer}$ and $C_{inner}$ both contain two points of $P$
**Definition**

The **farthest-point Voronoi diagram** is the partition of the plane into regions where the same point is farthest.

**Note**

The region of a site $p_i$ is the common intersection of $n_1$ half-planes, so regions are convex, and boundaries are parts of bisectors.
FPVD Example

$\text{cell of } p_j$

$\text{cell of } p_i$

$p_i$

$p_j$
**Note**

Only points of the convex hull of $P$ can have cells in the farthest-point Voronoi diagram.

**Observation 7.13**

Given a set of $P$ of points in the plane, a point of $P$ has a cell in the farthest-point Voronoi diagram if and only if it is a vertex of the convex hull of $P$. 

---

$P_i$ $q$
**Farthest-Point Voronoi Diagrams**

**FPVD Construction**

**Step 0**
Compute the convex hull of $P$
Let $p_1, p_2, p_3, \ldots, p_n$ be the points on the convex hull
FPVD Construction: Phase 1

Remove and Remember

- Let the $p_1, \ldots, p_h$ be the random order of convex hull vertices.
- We remove the points $p_h, \ldots, p_4$ one by one from the cyclic order, and when removing $p_i$, store its clockwise neighbor $cw(p_i)$ and the counterclockwise neighbor $ccw(p_i)$ at the time of removal.
FPVD Construction: Phase 1 (Cont.)

For \( i \) ← \( m \) downto 4 do

Remove \( p_i \) from the convex hull; remember its 2 neighbors \( \text{cw}(p_i) \) and \( \text{ccw}(p_i) \) (at removal!)

\( p_8 \)

\( \text{cw}(p_8) \)

\( p_8 \)

\( \text{ccw}(p_8) \)
FPVD Construction: Phase 1 (Cont.)

For $i \leftarrow m$ downto 4 do
- Remove $p_i$ from the convex hull;
- Remember its 2 neighbors $cw(p_i)$ and $ccw(p_i)$ (at removal!)

$ccw(p_7)$

$cw(p_7)$

$p_8$

$p_7$

$p_7$

$p_8$
FPVD Construction: Phase 1 (Cont.)

For $i \leftarrow m$ downto 4 do
- Remove $p_i$ from the convex hull;
- Remember its 2 neighbors $\text{cw}(p_i)$ and $\text{ccw}(p_i)$ (at removal!)

$p_8$
$p_7$
$p_6$
$\text{cw}(p_6)$
$\text{ccw}(p_6)$
$p_8$
$p_7$
$p_6$
$\text{cw}(p_6)$
$\text{ccw}(p_6)$
FPVD Construction: Phase 2

**Put Back and Construct**

- Construct the farthest-point Voronoi diagram $F_3$ of $p_3, p_2, p_1$
- Add $p_i$ to the farthest-point Voronoi diagram $F_{i-1}$ to make $F_i$
FPVD Construction: Phase 2 (Cont.)

\[ F_{i-1} \]
FPVD Construction: Phase 2 (Cont.)

Voronoi diagrams of line segments
Farthest-point Voronoi diagrams
Roundness
Higher-order Voronoi diagrams
Computing the farthest-point Voronoi diagram
Roundness
FPVD Construction: Phase 2 (Cont.)

\[ F_{i-1} \]

Voronoi diagrams of line segments
Farthest-point Voronoi diagrams
Roundness
Higher-order Voronoi diagrams
Computing the farthest-point Voronoi diagram
Computational Geometry Lecture 13: More on Voronoi diagrams

\[ p_i \]

\[ ccw(p_i) \]

\[ cw(p_i) \]

\[ \text{bisector of } p_i \text{ and } ccw(p_i) \]

\[ \text{cell of } ccw(p_i) \]
FPVD Construction: Phase 2 (Cont.)

\[ F_{i-1} \]

\[ ccw(p_i) \]

\[ cw(p_i) \]

\[ cell \ of \ ccw(p_i) \]
FPVD Construction: Phase 2 (Cont.)

- Voronoi diagrams of line segments
- Farthest-point Voronoi diagrams
- Roundness
- Higher-order Voronoi diagrams
- Computing the farthest-point Voronoi diagram

\[ F_{i-1} \]

\[ p_i \]

\[ ccw(p_i) \]

\[ cw(p_i) \]

cell of \[ ccw(p_i) \]
FPVD Construction: Phase 2 (Cont.)

\[ p_i \]

\[ F_{i-1} \]

\[ ccw(p_i) \]

\[ cw(p_i) \]

\[ cell \ of \ cw(p_i) \]

\[ cell \ of \ ccw(p_i) \]
FPVD Construction: Phase 2 (Cont.)

$F_{i-1}$

$ccw(p_i)$

$cw(p_i)$

cell of $cw(p_i)$

cell of $ccw(p_i)$

$p_i$
FPVD Construction: Phase 2 (Cont.)

Voronoi diagrams of line segments
Farthest-point Voronoi diagrams
Roundness
Higher-order Voronoi diagrams
Computing the farthest-point Voronoi diagram
Roundness
Computational Geometry Lecture 13: More on Voronoi diagrams
FPVD Construction: Phase 2 (Cont.)

\[ F_{i-1} \]

\[ ccw(p_i) \]

\[ cw(p_i) \]

\[ cell \ of \ ccw(p_i) \]

\[ Pi \]
**Theorem 7.14**

Given a set of $n$ points in the plane, its farthest-point Voronoi diagram can be computed in $O(n \lg n)$ expected time using $O(n)$ storage.
Theorem 7.15

Given a set of $P$ of $n$ points in the plane, the smallest-width annulus (and the roundness) can be determined in $O(n^2)$ time using $O(n)$ storage.