VORONOI DIAGRAM
PART II

PETR FELKEL
FEL CTU PRAGUE
felkel@fel.cvut.cz
https://cw.felk.cvut.cz/doku.php/courses/a4m39vg/start

Based on [Berg], [Reiberg] and [Nandy]

Version from 16.11.2017
Talk overview

- Incremental construction
- Voronoi diagram of line segments
- VD of order k
- Farthest-point VD
Summary of the VD terms

- **Site** = input point, line segment, ...
- **Cell** = area around the site, in $VD_1$ the nearest to site
- **Edge, arc** = part of Voronoi diagram (border between cells)
- **Vertex** = intersection of VD edges
Incremental construction – bounded cell
Incremental construction – unbounded cell
Incremental construction algorithm

**InsertPoint(S, Vor(S), y) ... y = a new site**

*Input:* Point set $S$, its Voronoi diagram, and inserted point $y \notin S$

*Output:* VD after insertion of $y$

1. Find the site $x$ in which cell point $y$ falls, ... $O(\log n)$
2. Detect the intersections $\{a, b\}$ of bisector $L(x, y)$ with cell $x$ boundary => create the first edge $e = ab$ on the border of site $x$ ... $O(n)$
3. Set start intersection point $p = b$, set new intersection $c = \text{undef}$
4. site $z$ = neighbor site across the border with intersection $b$ ... $O(1)$
5. **while** (exists($p$) and $c \neq a$) // trace the bisectors from $b$ in one direction
   a. Detect intersection $c$ of $L(y, z)$ with border of cell $z$
   b. Report Voronoi edge $pc$
   c. $p = c$, $z$=neighbor site across border with intersec. $c$ 
   
   **5. if** ($c \neq a$) **then** // trace the bisectors from $a$ in other direction
   a. $p = a$
   b. *Similarly as in steps 3, 4, 5 with $a* 

   $O(n^2)$ worst-case, $O(n)$ expected time for some distributions
**Voronoi diagram of line segments**

Input: \( S = \{s_1, \ldots, s_n\} = \text{set of } n \text{ disjoint line segments (sites)} \)

VD: line segments, parabolic arcs

Distance measured perpendicularly to the object (line segment)

- Type 1
- Type 2
- Type 3

[Berg]

Felkel: Computational geometry

(8 / 45)
VD of line segments with bounding box

BBOX => standard DCEL
Bisector of 2 line-segments in detail

- Consists of line segments and parabolic arcs
  
  Distance from point-to-object is measured to the closest point on the object (perpendicularly to the object silhouette)

  - Line segment – bisector of end-points\(^{(1)}\) or of interiors\(^{(2)}\)
  - Parabolic arc – of point and interior\(^{(3)}\) of a line segment

![Diagram of bisector of two disjoint line segments with ≤7 parts]
Bisector in greater details

Type 2

Bisector of two line segment interiors  
(in intersection of perpendicular slabs only)

Type 3

Bisector of (end-)point and line segment interior

Felkel: Computational geometry  
(11 / 45)
VD of points and line segments examples

2 points       Point & segment       2 line segments
Voronoi diagram of line segments

- More complex bisectors of line segments
  - VD contains line segments and parabolic arcs
- Still combinatorial complexity of $O(n)$
- Assumptions on the input line segments:
  - non-crossing
  - strictly disjoint end-points (slightly shorten the segm.)

if (we allow touching segments)

Shared endpoints cause complication:
The whole region is equally close to two line segments
Shape of Beach line for line segments

= Points with distance to the closest site above sweep line $l$ equal to the distance to $l$

- Beach line contains
  - *parabolic arcs* when closest to a site end-point
  - *straight line segments* when closest to a site interior
  (or just the part of the site interior above $l$ if the site $s$ intersects $l$)

(This is the shape of the beach line)
### Beach line breakpoints types

Breakpoint $p$ is equidistant from $l$ and equidistant and closest to:

<table>
<thead>
<tr>
<th>Points</th>
<th>Segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. two site end-points</td>
<td>=&gt; $p$ traces a VD line segment</td>
</tr>
<tr>
<td>2. two site interiors</td>
<td>=&gt; $p$ traces a VD line segment</td>
</tr>
<tr>
<td>3. end-point and interior</td>
<td>=&gt; $p$ traces a VD parabolic arc</td>
</tr>
<tr>
<td>4. one site end-point</td>
<td>=&gt; $p$ traces a line segment (border of the slab perpendicular to the site)</td>
</tr>
<tr>
<td>5. site interior intersects the scan line $l$</td>
<td>=&gt; $p =$ intersection, traces the input line segment</td>
</tr>
</tbody>
</table>

Cases 4 and 5 involve only one site and therefore do not form a Voronoi diagram edge (are used by alg.only)
Breakpoints types and what they trace

- 1,2 trace a Voronoi line segment (part of VD edge) **DRAW**
- 3 traces a Voronoi parabolic arc (part of VD edge) **DRAW**
- 4,5 trace a line segment (used only by the algorithm) **MOVE**
  - 4 limits the slab perpendicular to the line segment
  - 5 traces the intersection of input segment with a sweep line

(This is the shape of the traced VD arcs)
Site event – sweep line reaches an endpoint

I. At upper endpoint of
   – Arc above is split into two
   – four new arcs are created
     (2 segments + 2 parabolas)
   – Breakpoints for two segments
     are of type 4-5-4
   – Breakpoints for parabolas
     depend on the surrounding
     sites
     • Type 1 for two end-points
     • Type 3 for endpoint and interior
     • etc…
Site event – sweep line reaches an endpoint

II. At lower endpoint of
   − Intersection with interior
     (breakpoint of type 5)
     - is replaced by two breakpoints
       (of type 4)
       with parabolic arc between them
Circle event – lower point of circle of 3 sites

- Two breakpoints meet (on the beach-line)
- Solution depends on their type
  - Any of first three types (1, 2, or 3) meet
    - 3 sites involved – Voronoi vertex created
  - Type 4 with something else
    - two sites involved – breakpoint changes its type
    - Voronoi vertex not created
      (Voronoi edge may change its shape)
  - Type 5 with something else
    - never happens for disjoint segments
      (meet with type 4 happens before)
Motion planning example - retraction

Find path for a circular robot of radius $r$ from $Q_{\text{start}}$ to $Q_{\text{end}}$
Motion planning example - retraction

Find path for a circular robot of radius $r$ from $Q_{start}$ to $Q_{end}$

- Create Voronoi diagram of line segments, take it as a graph
- Project $Q_{start}$ to $P_{start}$ on $VD$ and $Q_{end}$ to $P_{end}$
- Remove segments with distance to sites smaller than radius $r$ of a robot
- Depth first search if path from $P_{start}$ to $P_{end}$ exists
- Report path $Q_{start} P_{start} ... path ... P_{end}$ to $Q_{end}$

- $O(n \log n)$ time using $O(n)$ storage
Order-2 Voronoi diagram

\[ V(p_i, p_j) : \text{the set of points of the plane closer to each of } p_i \text{ and } p_j \text{ than to any other site} \]

Property
The order-2 Voronoi regions are convex
Construction of $V(3,5) = V(5,3)$

Intersection of all halfplanes except $h(3,5)$ and $h(5,3)$

$$\bigcap_{x \neq 5} h(3, x) \cap \bigcap_{x \neq 3} h(5, x)$$
Order-2 Voronoi edges

edge: set of centers of circles passing through 2 sites s and t and containing one site p
=> c_p(s,t)

Question
Which are the regions on both sides of c_p(s,t)?

=> V(p,s) and V(p,t)
Order-2 Voronoi vertices

vertex: center of a circle passing through at least 3 sites and containing either site p or nothing

⇒ $u_p(Q)$ or $u_\emptyset(Q \cup p)$

$u_5(2,3,7), u_\emptyset(3,6,7)$

(circle circumscribed to Q)
Order-2 Voronoi vertex $u_p(Q)$

- Center of a circle passing through at least 3 sites and containing either site $p$ or nothing.

**Case** $u_p(Q)$ $u_5(2,3,7)$

- 5 is inside for all incident edges: $C_5(2,3)$ $C_5(2,7)$ $C_5(3,7)$

- => is inside for circle with center in vertex.
Order-2 Voronoi vertex $\mathbf{u}_\emptyset(Q \cup p)$

**vertex** : center of a circle passing through at least 3 sites and containing either site $p$ or nothing

**Case** $\mathbf{u}_\emptyset(Q \cup p)$
$\mathbf{u}_\emptyset(3,6,7,5)$

Felkel: Computational geometry
(Nandy)
Order-k Voronoi Diagram

The size of the order-k diagrams is $O(k(n-k))$

The order-k diagrams can be constructed from the order-(k-1) diagrams in $O(k(n-k))$ time

The order-k diagrams can be iteratively constructed in $O(n \log n + k^2(n-k))$ time
Order $n-1 = \text{Farthest-point Voronoi diagram}$

Cell $V_{-1}(7) = V_{n-1}(\{1,2,3,4,5,6\})$
- set of points in the plane farther from $p_i=7$
- than from any other site

$\text{Vor}_{-1}(P) = \text{Vor}_{n-1}(P)$
- partition of the plane formed by the farthest point Voronoi regions, their edges, and vertices
Farthest-point Voronoi diagrams example

Roundness of manufactured objects

- Input: set of measured points in 2D
- Output: width of the smallest-width annulus (region between two concentric circles $C_{inner}$ and $C_{outer}$)

Three cases to test – one will win:

a) 3 in – 1 out
b) 1 point in – 3 out
c) 2 in – 2 out
Smallest width annulus – cases with 3 pts

a) $C_{\text{inner}}$ contains at least 3 points
- Center is the *vertex of normal Voronoi diagram* (1\textsuperscript{st} order VD)
- The remaining point on $C_{\text{outer}}$ in $O(n)$ for each vertex

b) $C_{\text{outer}}$ contains at least 3 points
- Center is the *vertex of the farthest Voronoi diagram*
- The remaining point on $C_{\text{inner}}$ in $O(n)$
Smallest width annulus – case with 2+2 pts

c) $C_{\text{inner}}$ and $C_{\text{outer}}$ contain 2 points each

- Generate vertices of overlay of Voronoi (___) and farthest-point Voronoi (- - -) diagrams
  => $O(n^2)$ candidates for centers
  (we need only vertices, not the complete overlay)

- annulus computed in $O(1)$ from center and 4 points
  (same for all 3 cases)

- $O(n^2)$
Smallest width annulus

Smallest-Width-Annulus
Input: Set $P$ of $n$ points in the plane
Output: Smallest width annulus center and radii $r$ and $R$ (roundness)

1. Compute Voronoi diagram $\text{Vor}(P)$
   and farthest-point Voronoi diagram $\text{Vor}_1(P)$ of $P$
2. For each vertex of $\text{Vor}(P)$ ($r$) determine the farthest point ($R$) from $P$
   $\Rightarrow O(n)$ sets of four points defining candidate annuli – case a)
3. For each vertex of $\text{Vor}_1(P)$ ($R$) determine the closest point ($r$) from $P$
   $\Rightarrow O(n)$ sets of four points defining candidate annuli – case b)
4. For every pair of edges $\text{Vor}(P)$ and $\text{Vor}_1(P)$ test if they intersect
   $\Rightarrow$ another set of four points defining candidate annulus – c)
5. For all candidates of all three types chose the smallest-width annulus

$O(n^2)$ time using $O(n)$ storage
Farthest-point Voronoi diagram

$V_{-1}(p_i)$ cell
= set of points in the plane farther from $p_i$ than from any other site

$\text{Vor}_{-1}(P)$ diagram
= partition of the plane formed by the farthest point Voronoi regions, their edges, and vertices
Farthest-point Voronoi region (cell)

Computed as intersection of halfplanes, but we take “other sides” of bisectors

Construction of $V_{-1}(7)$

$$V_{-1} = \bigcap_{x=1}^{n} h(y, x), \ y \neq x$$

Property

The farthest point Voronoi regions are convex and unbounded
Farthest-point Voronoi region

Properties:

- Only vertices of the convex hull have their cells in farthest Voronoi diagram
- The farthest point Voronoi regions are unbounded
- The farthest point Voronoi edges and vertices form a tree (in the graph sense)
Farthest point Voronoi edges and vertices

**edge**: set of points equidistant from 2 sites and closer to all the other sites

**vertex**: point equidistant from at least 3 sites and closer to all the other sites

[Source: [Nandy](https://www.computational-geometry.org)]
Application of Vor_{-1}(P) : Smallest enclosing circle

- Construct Vor_{-1}(P) and find minimal circle with center in Vor_{-1}(P) vertices or on edges
Modified DCEL for farthest-point Voronoi diagram

- Half-infinite edges -> we adapt DCEL
- Half-edges with origin in infinity
  - Special vertex-like record for origin in infinity
  - Store direction instead of coordinates
  - Next(e) or Prev(e) pointers undefined
- For each inserted site $p_j$
  - store a pointer to the most CCW half-infinite half-edge of its cell in DCEL
Idea of the algorithm

1. Create the convex hull and number the CH points randomly
2. Remove the points starting in the last of this random order and store \(cw(p_i)\) and \(ccw(p_i)\) points at the time of removal.
3. Include the points back and compute \(V_{-1}\)

\[
\begin{array}{|c|c|c|}
\hline
p_i & ccw(p_i) & cw(p_i) \\
\hline
p_6 & p_3 & p_5 \\
\hline
p_5 & p_3 & p_2 \\
\hline
\ldots & & \\
\hline
\end{array}
\]
Farthest-point Voronoi d. construction

Farthest-pointVoronoi O(nlog n) time in O(n) storage

Input: Set of points P in plane
Output: Farthest-point VD Vor-1(P)

1. Compute convex hull of P
2. Put points in CH(P) of P in random order p_1,...,p_h
3. Remove p_h, ...,p_4 from the cyclic order (around the CH).
   When removing p_i, store the neighbors: cw(p_i) and ccw(p_i) at the time of removal. (This is done to know the neighbors needed in step 6.)
4. Compute Vor-1( { p_1, p_2, p_3 } ) as init
5. for i = 4 to h do
6.   Add site p_i to Vor-1( { p_1, p_2, ..., p_{i-1} } ) between site cw(p_i) and ccw(p_i)
7.     - start at most CCW edge of the cell ccw(p_i)
8.     - continue CW to find intersection with bisector( ccw(p_i), p_i )
9.     - trace borders of Voronoi cell p_i in CCW order, add edges
10.    - remove invalid edges inside of Voronoi cell p_i
Farthest-point Voronoi d. construction

Insertion of site $p_i$
Start with site ccw($p_i$) and ccw edge of its cell
Farthest-point Voronoi d. construction

After insertion of site $p_i$
References


