Delaunay Triangulation
Delaunay Triangulation
Delaunay Triangulation
Delaunay Triangulation

- Pencils of circles
- Definition, empty circle property
- Motivations: reconstruction, meshing
- Properties: EMST, max-min angle
- Lower bound
- In-sphere predicate
- Diagonal flipping
- Incremental algorithm
- Sweep line algorithm
- Miscellaneous
Delaunay Triangulation: pencils of circles

Imagine moving circles
Delaunay Triangulation: pencils of circles

Imagine moving circles

fixed center

increasing radius
Delaunay Triangulation: pencils of circles

Imagine moving circles

fixed center

increasing radius
Delaunay Triangulation: pencils of circles

Imagine moving circles

fixed center

increasing radius
Delaunay Triangulation: pencils of circles

Imagine moving circles

fixed center

increasing radius
Delaunay Triangulation: pencils of circles

Imagine moving circles

fixed center

increasing radius

Cocentric pencil
Delaunay Triangulation: pencils of circles

Imagine moving circles
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points

radical axis

Pencil with base points
Delaunay Triangulation: pencils of circles

Imagine moving circles
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points
Delaunay Triangulation: pencils of circles

Imagine moving circles
two fixed points
Pencil with limit points
radical axis
4 - 9
Delaunay Triangulation: pencils of circles

Imagine moving circles

two fixed points

Pencil with limit points

orthogonal

radical axis

4 - 10
Delaunay Triangulation: pencils of circles

Imagine moving circles
Delaunay Triangulation: pencils of circles

Imagine moving circles

a point on a line
Delaunay Triangulation: pencils of circles

Imagine moving circles

a point on a line
Delaunay Triangulation: pencils of circles

Imagine moving circles

a point on a line
Delaunay Triangulation: pencils of circles

Imagine moving circles

a point on a line
Delaunay Triangulation: pencils of circles

Imagine moving circles

a point on a line
Delaunay Triangulation: pencils of circles

Imagine moving circles

a point on a line
Delaunay Triangulation: pencils of circles

Imagine moving circles

a point on a line
Delaunay Triangulation: pencils of circles

Imagine moving circles

a point on a line
Delaunay Triangulation: pencils of circles

Imagine moving circles

radical axis

5 - 10

a point on a line

Pencil with tangent point
Delaunay Triangulation: pencils of circles

Imagine moving circles

a point on a line

orthogonal

radical axis

Pencil with tangent point

5 - 11
Delaunay Triangulation: pencils of circles

Circle equation

\[ x^2 + y^2 - 2ax - 2by + c = 0 \]
Delaunay Triangulation: pencils of circles

Circle equation

\[ x^2 + y^2 - 2ax - 2by + c = 0 \]

Another circle equation

\[ x^2 + y^2 - 2a'x - 2b'y + c' = 0 \]
Delaunay Triangulation: pencils of circles

Circle equation

\[ x^2 + y^2 - 2ax - 2by + c = 0 \]

Another circle equation

\[ x^2 + y^2 - 2a'x - 2b'y + c' = 0 \]

Pencil of circles

\[ \lambda \cdot (x^2 + y^2 - 2ax - 2by + c) + (1 - \lambda) \cdot (x^2 + y^2 - 2a'x - 2b'y + c') = 0 \]
Delaunay Triangulation: pencils of circles

Circle equation

\[ x^2 + y^2 - 2ax - 2by + c = 0 \]

Another circle equation

\[ x^2 + y^2 - 2a'x - 2b'y + c' = 0 \]

Pencil of circles

\[ \lambda \cdot (x^2 + y^2 - 2ax - 2by + c) + (1 - \lambda) \cdot (x^2 + y^2 - 2a'x - 2b'y + c') = 0 \]

A special "circle: the radical axis

\[ \lambda = \infty \]
Delaunay Triangulation: pencils of circles

Power of a point w.r.t a circle

\[ x^2 + y^2 - 2ax - 2by + c \]
Delaunay Triangulation: pencils of circles

Power of a point w.r.t a circle

\[ x^2 + y^2 - 2ax - 2by + c \]

\[ \begin{align*} 
= 0 & \quad \text{on the circle} \\
< 0 & \quad \text{inside the circle} \\
> 0 & \quad \text{outside the circle}
\end{align*} \]
Delaunay Triangulation: pencils of circles

Power of a point w.r.t a circle

\[ \lambda \ (x^2 + y^2 - 2a'x - 2b'y + c') + (1 - \lambda) \ (x^2 + y^2 - 2ax - 2by + c) = 0 \]

blue yields smaller power

black yields smaller power

equal power
Delaunay Triangulation: definition, empty circle property
Delaunay Triangulation: definition, empty circle property

Point set

8 - 2
Delaunay Triangulation: definition, empty circle property

Point set

Query
Delaunay Triangulation: definition, empty circle property

Point set

Query
Delaunay Triangulation: definition, empty circle property
Delaunay Triangulation: definition, empty circle property

Point set
Query
Nearest neighbor
Delaunay Triangulation: definition, empty circle property

Point set

Voronoi diagram
Delaunay Triangulation: definition, empty circle property
Delaunay Triangulation: definition, empty circle property
Delaunay Triangulation: definition, empty circle property

Point set

Voronoi diagram
Delaunay Triangulation: definition, empty circle property

- Point set
- Voronoi diagram
- Delaunay triangulation
Delaunay Triangulation: definition, empty circle property

Point set

Delaunay triangulation

Empty circle property
Delaunay Triangulation: definition, empty circle property
Delaunay Triangulation:

Input: a set of points on an unknown curve
Delaunay Triangulation:

Input: a set of points on an unknown curve

Output: the curve (the points in order along the curve)
Delaunay Triangulation:

Input: a set of points on an unknown curve

Output: the curve (the points in order along the curve)
Delaunay Triangulation:

Input: a set of points on an unknown curve

Output: the curve (the points in order along the curve)
Delaunay Triangulation:

Input: a set of points on an unknown curve

Output: the curve (the points in order along the curve)

If good sampling, output $\in$ Delaunay
Delaunay Triangulation:

Teaser reconstruction lecture
Delaunay Triangulation:

Input: a set of points on an unknown surface

Output: the surface (a triangulation of the points approximating the surface)
Delaunay Triangulation:

Input: a set of points on an unknown surface

Output: the surface (a triangulation of the points approximating the surface)
Delaunay Triangulation:

Input: a set of points on an unknown surface

Output: the surface (a triangulation of the points approximating the surface)

If good sampling, output ∈ Delaunay
Delaunay Triangulation:

Teaser reconstruction lecture
Delaunay Triangulation:

Teaser meshing lecture
Delaunay Triangulation:

Shape
Delaunay Triangulation:

Shape
Cut in simple pieces
triangles
Delaunay Triangulation:

Shape
Cut in simple pieces
triangles
Delaunay Triangulation:

Shape

Cut in simple pieces

Quads
Delaunay Triangulation:

Shape

Cut in simple pieces

triangles

For unstructured mesh: add points and compute Delaunay
Delaunay Triangulation:

Teaser meshing lecture
Delaunay Triangulation: Teaser meshing lecture

sharp features
Delaunay Triangulation: EMST
Delaunay Triangulation: EMST

A spanning tree
Delaunay Triangulation: EMST

Another spanning tree
Delaunay Triangulation: EMST

The Euclidean Minimum-length Spanning Tree
Delaunay Triangulation: EMST

The Euclidean Minimum-length Spanning Tree is included in Delaunay
Delaunay Triangulation: EMST

The Euclidean Minimum-length Spanning Tree is included in Delaunay

Proof:
Choose an edge of EMST
Delaunay Triangulation: \text{EMST}

The Euclidean Minimum-length Spanning Tree

Proof:

Split points
Delaunay Triangulation: EMST

Proof:

Split points
Delaunay Triangulation: EMST

Is diametral circle empty?

Proof:

Split points
Delaunay Triangulation: \textit{EMST}

Is diametral circle empty?  

assume $\exists$ blue point inside

Proof:

Split points

12 - 10
Delaunay Triangulation: \textit{EMST}

**Is diametral circle empty?**

**Proof:**

Assume \( \exists \) blue point inside better spanning tree.
Delaunay Triangulation: EMST

Is diametral circle empty?

Proof:

Empty circle $\implies$ The edge is in Delaunay triangulation

Split points

assume $\exists$ blue point inside better spanning tree
Delaunay Triangulation: \textit{EMST}

Is diametral circle empty?

Proof:

Adding a red blue-edge create a cycle $\Rightarrow$

The edge is the shortest red-blue edge

Split points
Delaunay Triangulation: EMST

Algorithm
Delaunay Triangulation: EMST

Algorithm
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge
Delaunay Triangulation: EMST

Algorithm

choose shorter purple edge

$O(n \log n)$ after Delaunay
Delaunay Triangulation: size
Delaunay Triangulation: size

Convex hull

Euler relation

Polytope boundary

Three dimensions

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Edges</th>
<th>Faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$e$</td>
<td>$f$</td>
</tr>
</tbody>
</table>

$\begin{align*}
  n - e + f &= 2 \\
  3f &= 2e \\
  f &= 2n - 4 \\
  e &= 3n - 6
\end{align*}$
Delaunay Triangulation: size

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Edges</th>
<th>Faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$e$</td>
<td>$f$</td>
</tr>
</tbody>
</table>

$$n - e + f = 2$$
Delaunay Triangulation: size

Convex hull

\[ k \]

\[ n - k \]

Vertices: \( n - e + f = 2 \)

Edges

\[ e \]

Faces

\[ f \]

Triangles

\[ \infty \]

1

\[ t \]
Delaunay Triangulation: size

Convex hull

- $k$
- $n - k$

$n - k$

Vertices: $n - e + f = 2$

Edges: $\infty$

Faces: $1$

Triangles: $t$

$n - e + t + 1 = 2$
Delaunay Triangulation: size

Vertex: $n - k$

Edges: $n - e + f = 2$

Faces: $3t + k = 2e$

Convex hull: $k$

Triangles: $1$
Delaunay Triangulation: size

Convex hull

$k$

$n - k$

$n - e + t + 1 = 2$

$3t + k = 2e$

Triangles

$\infty$

$1$

$t$

Vertices

$n - e + f = 2$

Edges

Faces

$2n - 3t - k + 2t = 2$
Delaunay Triangulation: size

Convex hull

$k$

$n - k$

Vertices

$n - e + f = 2$

Edges

$\infty$

Faces

1

Triangles

$t$

$n - e + t + 1 = 2$

$3t + k = 2e$

$2n - 3t - k + 2t = 2$

$t = 2n - k - 2 < 2n$

$e = 3n - k - 3 < 3n$
Delaunay Triangulation: size

\[ \sum_{p \in S} d^\circ(p) = 2e = 6n - 2k - 6 \]

\[ \mathbb{E}(d^\circ(p)) = \frac{1}{n} \sum_{p \in S} d^\circ(p) < 6 \]

average on the choice of point \( p \) in set of points \( S \)

\[ n - e + t + 1 = 2 \]

\[ 3t + k = 2e \]

\[ 2n - 3t - k + 2t = 2 \]

\[ t = 2n - k - 2 < 2n \]

\[ e = 3n - k - 3 < 3n \]
Delaunay Triangulation: max-min angle
Delaunay Triangulation: max-min angle
Delaunay Triangulation: max-min angle

Triangulation

Delaunay

smallest angle
Delaunay Triangulation: max-min angle

Triangulation

Delaunay

smallest angle
Delaunay Triangulation: max-min angle

Triangulation

Delaunay

smallest angle
Delaunay Triangulation: max-min angle

Triangulation

Delaunay

smallest angle

second smallest angle
Delaunay Triangulation: max-min angle

Proof
Delaunay Triangulation: max-min angle

Definition

Delaunay edge
Delaunay Triangulation: max-min angle

Definition

Delaunay edge

∃ empty circle
Delaunay Triangulation: max-min angle

Definition

locally Delaunay edge w.r.t. a triangulation
Delaunay Triangulation: \text{max-min angle}

Definition

\text{locally} \quad \text{Delaunay edge} \quad \text{w.r.t. a triangulation}

\exists \text{ circle}

\text{not enclosing the two neighbors}

\text{neighbor} = \text{visible from the edge}
Delaunay Triangulation: max-min angle

Lemma \( (\forall \text{ edge: locally Delaunay}) \iff \text{Delaunay} \)
Delaunay Triangulation: max-min angle

Lemma \((\forall \text{ edge: locally Delaunay}) \iff \text{Delaunay}\)

Proof:

choose an edge
Delaunay Triangulation: max-min angle

Lemma \( (\forall \text{ edge: locally Delaunay}) \iff \text{Delaunay} \)

Proof:

- choose an edge
- edges of the quadrilateral are locally Delaunay
Delaunay Triangulation: max-min angle

Lemma \( (\forall \text{ edge: locally Delaunay}) \iff \text{Delaunay} \)

Proof:

- choose an edge
- edges of the quadrilateral are locally Delaunay
Delaunay Triangulation: max-min angle

Lemma    \((\forall \text{ edge: locally Delaunay}) \iff \text{Delaunay}\)  

Proof:

- choose an edge
- edges of the quadrilateral are locally Delaunay
- Vertices visible through one edge are outside circle
Delaunay Triangulation: max-min angle

Lemma \((\forall \text{ edge}: \text{ locally Delaunay}) \iff \text{ Delaunay}\)

Proof:

- choose an edge
- edges of the quadrilateral are locally Delaunay
- Vertices visible through one edge are outside circle
- Induction \(\rightarrow\) all vertices outside circle
Delaunay Triangulation: max-min angle

Lemma
For four points in convex position

Delaunay $\iff$ maximize the smallest angle

Two possible triangulation
Delaunay Triangulation: max-min angle

Lemma
For four points in convex position

Delaunay $\iff$ maximize the smallest angle

Case 1: smallest angle in corner
Delaunay Triangulation: max-min angle

Lemma For four points in convex position

Delaunay $\iff$ maximize the smallest angle

Case 1: smallest angle in corner

$\exists$ a smaller angle $\in$ other triangulation
Delaunay Triangulation: max-min angle

Lemma  For four points in convex position

Delaunay $\iff$ maximize the smallest angle

Case 2: smallest angle along diagonal
Delaunay Triangulation: max-min angle

Lemma

For four points in convex position

\[ \text{Delaunay} \iff \text{maximize the smallest angle} \]

Case 2: smallest angle along diagonal
Delaunay Triangulation: max-min angle

Lemma

For four points in convex position

Delaunay $\iff$ maximize the smallest angle

Case 2: smallest angle along diagonal

$\exists$ a smaller angle $\in$ other triangulation
Delaunay Triangulation: \( \text{max-min angle} \)

Map: Triangulations \( \rightarrow \mathbb{R}^{6n-3k-4} \) smallest angle \( \alpha_1 \)
Delaunay Triangulation: \( \text{max-min angle} \)

Map: Triangulations \( \rightarrow \mathbb{R}^{6n-3k-4} \)

- smallest angle \( \alpha_1 \)
- second smallest angle \( \alpha_2 \)
Delaunay Triangulation: max-min angle

Map: Triangulations $\rightarrow \mathbb{R}^{6n-3k-4}$

smallest angle $\alpha_1$
second smallest angle $\alpha_2$
third smallest angle $\alpha_3$
Delaunay Triangulation: max-min angle

Map: Triangulations $\rightarrow \mathbb{R}^{6n-3k-4}$

$\left( \alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{6n-3k-4} \right)$

smallest angle $\alpha_1$

second smallest angle $\alpha_2$

third smallest angle $\alpha_3$
Delaunay Triangulation: \( \text{max-min angle} \)

Map: Triangulations \( \rightarrow \mathbb{R}^{6n-3k-4} \)

\( (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{6n-3k-4}) \)

smallest angle \( \alpha_1 \)
second smallest angle \( \alpha_2 \)
third smallest angle \( \alpha_3 \)

sort triangulations in lexicographic order
Delaunay Triangulation: \ max-min \ angle

Theorem:

Delaunay maximizes minimum angles (in lexicographic order)
Delaunay Triangulation: max-min angle

Theorem:

Delaunay maximizes minimum angles (in lexicographic order)

Proof:

Let $T$ be the triangulation maximizing angles
Theorem:

Delaunay triangulation maximizes minimum angles (in lexicographic order).

Proof:

Let $T$ be the triangulation maximizing angles.

$$\iff \forall \text{ convex quadrilateral (from 2 triangles } \in T) \text{ the diagonal maximizes smallest angle (in quad)}$$
Delaunay Triangulation: max-min angle

Theorem:

Delaunay maximizes minimum angles (in lexicographic order)

Proof:

Let $T$ be the triangulation maximizing angles

$\implies \forall$ convex quadrilateral (from 2 triangles $\in T$)

the diagonal maximizes smallest angle (in quad)

$\implies \forall$ edge, it is locally Delaunay
**Delaunay Triangulation:** max-min angle

**Theorem:**

Delaunay maximizes minimum angles (in lexicographic order)

**Proof:**

Let $T$ be the triangulation maximizing angles

$\implies \forall$ convex quadrilateral (from 2 triangles $\in T$)

the diagonal maximizes smallest angle (in quad)

$\implies \forall$ edge, it is locally Delaunay

$\implies T = $ Delaunay
Delaunay Triangulation: lower bound

Convex hull

A stupid algorithm for sorting numbers

\[ f(n) + O(n) \geq \Omega(n \log n) \]
Delaunay Triangulation: lower bound

A stupid algorithm for sorting numbers

\[ \begin{align*} O(n) & \quad \text{project on parabola} \\
\quad & \\
\quad & \\
f(n) & \quad \text{compute convex hull} \\
\quad & \\
\quad & \\
O(n) & \quad \text{find lowest point} \\
\quad & \\
\quad & \\
O(n) & \quad \text{enumerate x coordinates in ccw CH order} \\
\quad & \\
\quad & \\
\end{align*} \]

Lower bound on sorting

\[ f(n) + O(n) \geq \Omega(n \log n) \]
Convex hull

\[ vwn + ? \]

\[
\begin{vmatrix}
  x_w - x_v & x_n - x_v \\
  y_w - y_v & y_n - y_v
\end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\
  x_v & x_w & x_n \\
  y_v & y_w & y_n
\end{vmatrix} > 0
\]

\[ vwn - ? \]

\[
\begin{vmatrix} 1 & 1 & 1 \\
  x_v & x_w & x_n \\
  y_v & y_w & y_n
\end{vmatrix} < 0
\]

\[ vwn 0 ? \]

\[
\begin{vmatrix} 1 & 1 & 1 \\
  x_v & x_w & x_n \\
  y_v & y_w & y_n
\end{vmatrix} = 0
\]

Delaunay Triangulation: incircle predicate
Delaunay Triangulation: incircle predicate

$pqr$ ccw triangle

query $s$
Delaunay Triangulation: incircle predicate

$pqr$ ccw triangle

query $s$ inside circumcircle
Delaunay Triangulation: incircle predicate

$pqr$ ccw triangle

query $s$ cocircular
Delaunay Triangulation: incircle predicate

$pqr$ ccw triangle

query $s$ outside circumcircle
Delaunay Triangulation: \texttt{incircle predicate}

\[pqr \text{ ccw triangle}

\text{query } s

22 - 6
Delaunay Triangulation: incircle predicate

Space of circles

\[ p = (x, y) \leadsto p^* = (x, y, x^2 + y^2) \]
Delaunay Triangulation: incircle predicate

Space of circles

\[ p = (x, y) \mapsto p^\star = (x, y, x^2 + y^2) \]

\[ C : x^2 + y^2 - 2ax - 2by + a^2 + b^2 - r^2 = 0 \]

\[ \mapsto C^\star = (a, b, a^2 + b^2 - r^2) \]
Delaunay Triangulation: incircle predicate

Space of circles

\[ p = (x, y) \mapsto p^* = (x, y, x^2 + y^2) \]

\[ C : x^2 + y^2 - 2ax - 2by + a^2 + b^2 - r^2 = 0 \]

\[ \mapsto C^* = (a, b, a^2 + b^2 - r^2) \]

\[ \mapsto C^\dagger : z - 2ax - 2by + a^2 + b^2 - r^2 = 0 \]
Delaunay Triangulation: incircle predicate

Space of circles

\[ p \in C \iff p^* \in C^\dagger \]

\[ p = (x, y) \leadsto p^* = (x, y, x^2 + y^2) \]

\[ C : x^2 + y^2 - 2ax - 2by + a^2 + b^2 - r^2 = 0 \]

\[ \leadsto C^* = (a, b, a^2 + b^2 - r^2) \]

\[ \leadsto C^\dagger : z - 2ax - 2by + a^2 + b^2 - r^2 = 0 \]
Delaunay Triangulation: incircle predicate

Space of circles

\[ p \in C \iff p^* \in C^\dagger \]

circle through \( pqr \)
\[ \leadsto \text{plane through } p^*q^*r^* \]

\[ p = (x, y) \leadsto p^* = (x, y, x^2 + y^2) \]

\[ C : x^2 + y^2 - 2ax - 2by + a^2 + b^2 - r^2 = 0 \]
\[ \leadsto C^* = (a, b, a^2 + b^2 - r^2) \]

\[ \leadsto C^\dagger : z - 2ax - 2by + a^2 + b^2 - r^2 = 0 \]
Delaunay Triangulation: incircle predicate

Space of circles

\[ p \in C \iff p^* \in C^\dagger \]

\[ \text{s inside/outside of circle through } pqr \]
\[ \leadsto \text{plane through } p^*q^*r^* \]
\[ \text{above/below } s^* \]

\[ p = (x, y) \leadsto p^* = (x, y, x^2 + y^2) \]

\[ C : x^2 + y^2 - 2ax - 2by + a^2 + b^2 - r^2 = 0 \]

\[ \leadsto C^* = (a, b, a^2 + b^2 - r^2) \]

\[ \leadsto C^\dagger : z - 2ax - 2by + a^2 + b^2 - r^2 = 0 \]
Delaunay Triangulation: incircle predicate

Space of circles

- $s$ inside/outside of circle through $pqr$
- $\sim\Rightarrow$ plane through $p^*q^*r^*$
- above/below $s^*$

incircle predicate

$\sim\Rightarrow$ 3D orientation predicate
Delaunay Triangulation: incircle predicate

Space of circles

$s$ inside/outside of circle through $pqr$

$\leadsto$ plane through $p^*q^*r^*$

above/below $s^*$

incircle predicate

$\leadsto$ 3D orientation predicate

\[
\text{sign} = \begin{vmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{vmatrix}
\]

\[
\begin{vmatrix}
x_p & x_q & x_r & x_s \\
y_p & y_q & y_r & y_s \\
x_p^2 + y_p^2 & x_q^2 + y_q^2 & x_r^2 + y_r^2 & x_s^2 + y_s^2 \\
\end{vmatrix}
\]
Delaunay Triangulation: incircle predicate

Degeneracies
Delaunay Triangulation: incircle predicate

Degeneracies

Degree 4 vertex in Voronoi diagram
Delaunay Triangulation: incircle predicate

Degeneracies

Degree 4 vertex in Voronoi diagram
Delaunay Triangulation: incircle predicate

Degeneracies

Degree 4 vertex in Voronoi diagram

Delaunay quad ? random diagonal ?
Delaunay Triangulation: incircle predicate

Degeneracies

Degree 4 vertex in Voronoi diagram

Delaunay quad? random diagonal?

Teaser robustness lecture

assume no degeneracies for a while
Delaunay Triangulation:

Data structure for (Delaunay) triangulation

Representing incidences

Representing hull boundary

Representing user’s data

...
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping

non locally Delaunay
Delaunay Triangulation: Diagonal flipping

non locally Delaunay

locally Delaunay
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping

check edges of quadrilateral
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping
Delaunay Triangulation: Diagonal flipping

Delaunay is obtained
Delaunay Triangulation: Diagonal flipping

Complexity?
Delaunay Triangulation: Diagonal flipping

Complexity ?
Delaunay Triangulation: Diagonal flipping

Complexity?

locally convex

Locally Delaunay
Delaunay Triangulation: Diagonal flipping

Complexity?

Convex

Delaunay
Delaunay Triangulation: Diagonal flipping

Complexity?

Non convex

Non Delaunay
Delaunay Triangulation: Diagonal flipping

Complexity?

Non convex edge

Non Delaunay
Delaunay Triangulation: Diagonal flipping

- Complexity ?
- Non convex
- Flip

Non Delaunay
Delaunay Triangulation: Diagonal flipping

Complexity ?

Non convex

Flip

Non Delaunay
Delaunay Triangulation: Diagonal flipping

- Complexity
- Non convex
- Flip

- Non Delaunay
Delaunay Triangulation: Diagonal flipping

Complexity?

Non convex

Flip

Non Delaunay
Delaunay Triangulation: Diagonal flipping

Complexity?

Non convex

Flip

An hidden edge cannot be visible again

Non Delaunay
Delaunay Triangulation: Diagonal flipping

Complexity?

Non convex

Flip

An hidden edge cannot be visible again

At most $\frac{n(n-1)}{2}$ edges
Delaunay Triangulation: Diagonal flipping

Complexity?

Non convex

Flip

An hidden edge cannot be visible again

At most $\frac{n(n-1)}{2}$ edges

Complexity of diagonal flipping is $O(n^2)$
Delaunay Triangulation: Diagonal flipping

Complexity ?
Delaunay Triangulation: Diagonal flipping

Complexity ?
Delaunay Triangulation: Diagonal flipping

Complexity?
Delaunay Triangulation: Diagonal flipping

Complexity?
Delaunay Triangulation: Diagonal flipping

Complexity?
Delaunay Triangulation:  Diagonal flipping

Complexity ?

Delaunay
Delaunay Triangulation: Diagonal flipping

Complexity ?

Delaunay

Do not care
Delaunay Triangulation: Diagonal flipping

Complexity?

Encoding a triangulation

0011101010
Delaunay Triangulation: Diagonal flipping

Complexity?

Delaunay

11111000000

27 - 9
Delaunay Triangulation: Diagonal flipping

Complexity?

Encoding a triangulation

Flip

001110101010

swap
Delaunay Triangulation: Diagonal flipping

Complexity ?

00000011111

27 - 11
Delaunay Triangulation: Diagonal flipping

Complexity?

at least \( \left( \frac{n}{2} \right)^2 \) flips
Delaunay Triangulation: incremental algorithm
Delaunay Triangulation: incremental algorithm

New point
Delaunay Triangulation: incremental algorithm

New point

Locate
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: straight walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: straight walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: straight walk
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: straight walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: straight walk
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: straight walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: visibility walk

not unique
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: visibility walk

not unique
Delaunay Triangulation: incremental algorithm

Visibility walk terminates?
Delaunay Triangulation: incremental algorithm

Visibility walk terminates
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

May loop
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

May loop

Not Delaunay
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

Delaunay Triangulation: pencils of circles

Power of a point w.r.t a circle

$$\lambda \left( x^2 + y^2 - 2a' x - 2b'y + c' \right)$$
$$+ (1 - \lambda) \left( x^2 + y^2 - 2ax - 2by + c \right) = 0$$

blue yields smaller power
black yields smaller power
equal power
Delaunay Triangulation: incremental algorithm

Visibility walk terminates
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

Green power < Red power
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

Green power < Red power

Power decreases
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

Green power < Red power

Power decreases

Visibility walk terminates
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
**Delaunay Triangulation:** incremental algorithm

1. New point
2. Locate
3. Search conflicts
Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Delaunay Triangulation: incremental algorithm

New point
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts

\# triangles in conflict

\# triangles neighboring triangles in conflict
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts

\# triangles in conflict

\# triangles neighboring triangles in conflict

Degree of new point in new triangulation

< n
Delaunay Triangulation: incremental algorithm

Complexity

Locate
Walk may visit all triangles
< 2n

Search conflicts
degree of new point in new triangulation
< n
Delaunay Triangulation: incremental algorithm

Complexity

Locate

$O(n)$ per insertion

Search conflicts
Delaunay Triangulation: incremental algorithm

Complexity

Locate \( O(n) \) per insertion

Search conflicts \( O(n^2) \) for the whole construction
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts

half-parabola and circle
Delaunay Triangulation: incremental algorithm

Complexity
Locate
Search conflicts

half-parabola and circle
Delaunay triangle
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts

Insertion: $\Omega(n)$

Whole construction: $\Omega(n^2)$
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts

In practice

Many possibilities (walk, Delaunay hierarchy)

Randomized

Teaser randomization lecture
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Certified Delaunay triangles
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Certified Delaunay triangles
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Certified Delaunay triangles
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Certified Delaunay triangles
Certified Delaunay edges
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Boundary edges
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Boundary edges
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Boundary edges

Empty circles
tangent to sweep line
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Boundary edges

Empty circles tangent to sweep line

in order
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

New point

Empty circles tangent to sweep line
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

New point
Locate vertically
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

New point
Locate vertically
Create edge
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

New point
Locate vertically
Create edge
Modify boundary edges

α = β
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

New point
Locate vertically
Create edge
Modify boundary edges
Modify circle events
to be defined now

\[ \alpha = \beta \]
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Closing a triangle?
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Closing a triangle?
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Closing a triangle?

Circle events
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Closing a triangle?

Circle events

Next circle event
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Closing a triangle?

Next circle event
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Next circle event
Close triangle
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Next circle event
Close triangle
Modify boundary edges
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Next circle event
Close triangle
Modify boundary edges
Modify circle events
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Summary:

Process circle events and point events in $x$ order

Three data structures

- Triangulation
- List of events ($x$ sorted)
- List of boundary edges (ccw sorted)
## Delaunay Triangulation: sweep-line algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triangulation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List of events ($x$ sorted)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List of boundary edges</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ccw sorted)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

32 - 1
### Delaunay Triangulation: sweep-line algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events processed</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triangulation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List of events ($x$ sorted)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List of boundary edges (ccw sorted)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 - 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Delaunay Triangulation: sweep-line algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events processed</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>2n</td>
<td>n</td>
</tr>
</tbody>
</table>

### Triangulation

- List of events ($x$ sorted)
- List of boundary edges (ccw sorted)

32 - 3
**Delaunay Triangulation**: sweep-line algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events processed</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>$2n$</td>
<td>$n$</td>
</tr>
<tr>
<td>Triangulation</td>
<td>create 2 triangles per event</td>
<td>create one edge per event</td>
</tr>
<tr>
<td>List of events ($x$ sorted)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List of boundary edges</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ccw sorted)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$32 - 4$
## Delaunay Triangulation: sweep-line algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events processed</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td><strong>$2n$</strong></td>
<td><strong>$n$</strong></td>
</tr>
<tr>
<td>Triangulation</td>
<td>create 2 triangles per event</td>
<td>create one edge per event</td>
</tr>
<tr>
<td>List of events ($x$ sorted)</td>
<td>≤ 3 deletions ≤ 2 insertions per event</td>
<td>≤ 2 deletions ≤ 2 insertions per event</td>
</tr>
<tr>
<td>List of boundary edges (ccw sorted)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

32 - 5
**Delaunay Triangulation**: sweep-line algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events processed</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>2n</td>
<td>n</td>
</tr>
<tr>
<td>Triangulation</td>
<td>create 2 triangles per event</td>
<td>create one edge per event</td>
</tr>
<tr>
<td>List of events (x sorted)</td>
<td>≤ 3 deletions, ≤ 2 insertions per event</td>
<td>≤ 2 deletions, ≤ 2 insertions per event</td>
</tr>
<tr>
<td>List of boundary edges (ccw sorted)</td>
<td>replace 2 edges by 1 per event</td>
<td>locate, then insert 2 edges per event</td>
</tr>
</tbody>
</table>

32 - 6
**Delaunay Triangulation:** sweep-line algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events processed</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>2n</td>
<td>n</td>
</tr>
<tr>
<td>O(1) per operation</td>
<td>create 2 triangles per event</td>
<td>create one edge per event</td>
</tr>
<tr>
<td>O(log n) per operation</td>
<td>≤ 3 deletions, ≤ 2 insertions per event</td>
<td>≤ 2 deletions, ≤ 2 insertions per event</td>
</tr>
<tr>
<td>List of boundary edges</td>
<td>replace 2 edges by 1 per event</td>
<td>locate, then insert 2 edges per event</td>
</tr>
</tbody>
</table>

- List of events ($x$ sorted)
- List of boundary edges (ccw sorted)
### Delaunay Triangulation: sweep-line algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events processed</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>2n</td>
<td>n</td>
</tr>
<tr>
<td>Number</td>
<td>create 2 triangles</td>
<td>create one edge</td>
</tr>
<tr>
<td>Number</td>
<td>replace 2 edges by 1</td>
<td>locate, then insert 2 edges per event</td>
</tr>
<tr>
<td>Number</td>
<td>O(1) per operation</td>
<td>O(n log n) per event</td>
</tr>
<tr>
<td>List of events ((x) sorted)</td>
<td>per event</td>
<td></td>
</tr>
<tr>
<td>List of boundary edges (ccw sorted)</td>
<td>O(n log n) per event</td>
<td></td>
</tr>
</tbody>
</table>

#### Details:
- **Complexity**
  - Number of operations per event:
    - Circle events: \(O(1)\) per operation
    - Point events: \(O(n \log n)\) per operation
  - List of events (\(x\) sorted): \(O(\log n)\) per operation
  - List of boundary edges (ccw sorted): \(O(\log n)\) per operation

- **Number of Circle Events**:
  - \(2n\)

- **Number of Point Events**:
  - \(n\)

- **Operations per Event**:
  - Circle events:
    - Create 2 triangles
    - Replace 2 edges by 1
  - Point events:
    - Create one edge
    - Locate, then insert 2 edges per event
Delaunay Triangulation: divide & conquer (sketch)
Delaunay Triangulation: divide & conquer (sketch)
Delaunay Triangulation: divide & conquer (sketch)

Divide
Recurse
Delaunay Triangulation: divide & conquer (sketch)
Delaunay Triangulation: divide & conquer (sketch)

Divide
Recurse
Conquer

$O(n \log n)$
Delaunay Triangulation: divide & conquer (sketch)

Divide
Recurse
Conquer

$O(n \log n)$

balanced
linear time
easier conquer
Delaunay Triangulation: deletion algorithm (sketch)
Delaunay Triangulation: deletion algorithm (sketch)

Delaunay Triangulation: incremental algorithm

New point
Delaunay Triangulation: deletion algorithm (sketch)
Delaunay Triangulation: deletion algorithm (sketch)
Delaunay Triangulation: deletion algorithm (sketch)
Delaunay Triangulation: deletion algorithm (sketch)

Extract hole
Delaunay Triangulation: deletion algorithm (sketch)

Extract hole
Triangulate
Delaunay Triangulation: deletion algorithm (sketch)

Extract hole

Triangulate
Delaunay Triangulation: deletion algorithm (sketch)

Extract hole
Triangulate
Delaunay Triangulation: deletion algorithm (sketch)

Extract hole
Triangulate
and sew
Delaunay Triangulation: deletion algorithm (sketch)

- Extract hole
- Triangulate
- and sew

Be careful
Hole may be not convex
Delaunay Triangulation: deletion algorithm (sketch)

Extract hole
Triangulate
and sew

Be careful
Hole may be not convex
Delaunay Triangulation: deletion algorithm (sketch)

Ear queue
Delaunay Triangulation: deletion algorithm (sketch)

Ear queue

Ear with largest power is added
Delaunay Triangulation: deletion algorithm (sketch)

Ear queue
Ear with largest power is added
Delaunay Triangulation: deletion algorithm (sketch)

Ear queue

Ear with largest power is added

Iterate
Delaunay Triangulation: deletion algorithm (sketch)

Ear queue

Ear with largest power is added

Iterate
Delaunay Triangulation: deletion algorithm (sketch)

Triangulate and flip
Delaunay Triangulation: deletion algorithm (sketch)

Triangulate and flip
Triangulate and flip

for degree $\geq 8$
Delaunay Triangulation: deletion algorithm (sketch)

Decision tree for small holes

for degree $\leq 7$
Delaunay Triangulation: deletion algorithm (sketch)

Decision tree for small holes

degree 3
nothing to do

CGAL
for degree $\leq 7$

34 - 22
Delaunay Triangulation: deletion algorithm (sketch)

Decision tree for small holes

degree 4

for degree ≤ 7

34 - 23
Delaunay Triangulation: deletion algorithm (sketch)

Decision tree for small holes

degree 4

only one predicate

for degree $\leq 7$

CGAL

34 - 24
Delaunay Triangulation: deletion algorithm (sketch)

Decision tree for small holes

degree 5

for degree $\leq 7$
Delaunay Triangulation: deletion algorithm (sketch)

Decision tree for small holes

degree 6

"manual" d&conquer

for degree \( \leq 7 \)
Delaunay Triangulation: deletion algorithm (sketch)

Decision tree for small holes

degree 7

"manual" divide & conquer

symmetric tree

for degree \( \leq 7 \)
Delaunay Triangulation: 3D

Same as 2D

Dual Voronoi diagram

Empty sphere property

Triangle $\rightarrow$ Tetrahedron

Duality with 4D convex hull

Incremental algorithm (find the hole and star)
Delaunay Triangulation: 3D

Same as 2D

Duality with 4D convex hull

Incremental algorithm (find the hole and star)

Dehn Sommerville relations
\[ f_i = \#(\text{faces of dim } i) \]

Euler:
\[ f_0 - f_1 + f_2 - \ldots f_{d-1} = (-1)^{d-1} + 1 \]

Empty sphere property

Triangle

Tetrahedron

Convex hull

Higher dimensions

Dehn Sommerville relations
\[ f_i = \#(\text{faces of dim } i) \]

Euler:
\[ f_0 - f_1 + f_2 - \ldots f_{d-1} = (-1)^{d-1} + 1 \]

\[ \sum_j = k^{d-1} - 1 \begin{pmatrix} j + 1 \\ k + 1 \end{pmatrix} f_j = (-1)^{d-1} f_k \]

\[ -1 \leq k \leq d - 2 \]

\[ f_{-1} = f_d = 1 \]

\[ \left\lfloor \frac{d + 1}{2} \right\rfloor \text{ independent equations} \]
Delaunay Triangulation: 3D

Same as 2D

Dual Voronoi diagram

Empty sphere property

Triangle

Tetrahedron

Duality with 4D convex hull

Incremental algorithm (find the hole and star)

Convex hull

Higher dimensions

Dehn Sommerville relations $f_i = \#(\text{faces of dim } i)$

Euler: $f_0 - f_1 + f_2 - \ldots - f_{d-1} = (-1)^{d-1} + 1$

$$\sum_{j} = k^{d-1} - 1^j \binom{j + 1}{k + 1} f_j = (-1)^{d-1} f_k$$

$-1 \leq k \leq d - 2$

$f_{-1} = f_{d} = 1$

$$f_{-1} = f_{d} = 1$$

$$\left\lfloor \frac{d + 1}{2} \right\rfloor \text{ independent equations}$$

quadratic
Delaunay Triangulation: 3D

Quadratic examples
Delaunay Triangulation: 3D

Quadratic examples
Delaunay Triangulation: 3D

Quadratic examples
Delaunay Triangulation: 3D

Quadratic examples
Delaunay Triangulation: 3D

Quadratic examples
Delaunay Triangulation: 3D

Quadratic examples

Teaser probability lecture

Better results for random points
Delaunay Triangulation: 3D

Algorithms

4D convex hull duality

Flip

Incremental
Delaunay Triangulation: 3D

Algorithms

- 4D convex hull duality
- Flip
- Incremental

$O(f \log n + n^{\frac{4}{3}})$ or $\Theta(n^2)$

$\Theta(n^3)$

practical

Teaser randomization lecture
Delaunay Triangulation: higher dimensions

\[ d + 1 \text{ convex hull duality} \quad O \left( n^{-\left\lfloor \frac{d+1}{2} \right\rfloor} \right) \]

Incremental practical \( O(n) \) for random points

coeff exponential in \( d \)
The end