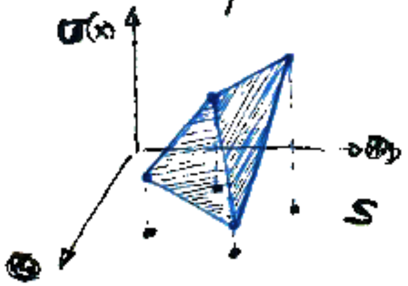
Triangulation of Point Sets

The following problem is of practical concern:

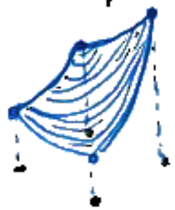
> Given a point set $S \subseteq \mathbb{R}^2$ where each point $x \in S$ is associated with a weight value $\sigma(x)$. Find a way to extend the definition of σ onto $\mathbb{R}^2$.

The problem is typically named "interpolation problem". There are uncountbly many ways to interpolate without doubt. But we hope that σ is as simple as possible. A very appealing choice would be piecewise linear function. The illustration on the left gives an example when $|S| = 4$. The four blue points correspond to the original definition



of σ, and we extended them "linearly" to cover the entire region of $\mathrm{Conv}(S)$. If we want, we could also extend further to $\mathbb{R}^2$, but for clarity we didn't show it in the illustration.

Here's where the notion of triangulation comes into play. Since in general three points in the space determines a plane, we somehow have to "triangulate" the point set S if we really want to have a piecewise linear interpolation.



[It's in general impossible to interpolate linearly between 4 points in the space. So "triangulation" is the absolute way to go.]

def. triangulation of point set.

Let $S \subseteq \mathbb{R}^2$ be a finite point set. A Collection $\mathcal{T}$ of triangles is a triangulation of S if

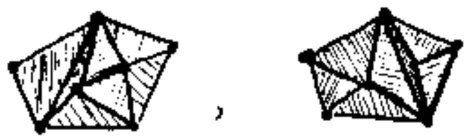
(1) $\bigcup\limits_{T \in \mathcal{T}} T = \mathrm{Conv}(S)$

(2) $\forall T \neq T' \in \mathcal{T}$, $T \cap T' = \emptyset$, a vertex, or an edge shared by both.

(3) $\bigcup\limits_{T \in \mathcal{T}} V(T) = S$.

The definition gurantees that no $T \in \mathcal{T}$ would contain a point from S in ~~the~~ its interior. (Exercise).

e.g.



are good;

are bad.

If we could find a triangulation of S, then the interpolation by piecewise linear function easily follows.

Actually, it's rather painless to prove the existence of triangulations for any S.

Theorem 18.
For any finite point set $S \subseteq \mathbb{R}^2$, except the very extreme~~s~~ case where all points in S are collinear, we could find a triangulation $\mathcal{T}$ of S in $O(n \log n)$ time.

Proof. The algorithm incrementally builds a triangulation for S, scanning from left to right. That is, assume the points in S are ordered $x_1, \dots, x_n$ from left to right; ~~we gradually~~ in step i it builds triangulations for the point set $\{x_1, \dots, x_i\}$ by inserting a ~~new~~ point x_i to the previously built triangulation and connect it to all points that it "sees".

Algorithm ScanTriangulate

Sort the points in S from left to right as $x_1, \dots, x_n$

let i_0 be the smallest index s.t. $\{x_1, \dots, x_{i_0}\}$ are non-collinear.

$\mathcal{T} :=$

for $i = i_0+1 \dots n$ do

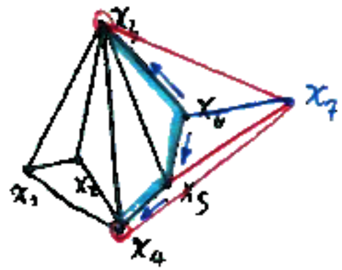
$\Gamma :=$ all points in $\{x_1, \dots, x_{i-1}\}$ that x_i could see.

$=: \{y_1, \dots, y_r\}$ in circular order

$\mathcal{T} := \mathcal{T} \cup \{\triangle y_j\, y_{j+1}\, x_i : 1 \le j \le r-1\}$

We could implement the for-loop efficiently.
In the round i:

- Observe that x_i could always see x_{i-1}. So $x_{i-1} \in \Gamma$.
- In order to obtain other elements of Γ, we start walking clockwise/counterclockwise from x_{i-1} until we reach the right/left tangents. Tangency test could be done in constant time for each point we encountered. (why?)



- Note that the edges that we walked through are immediately trapped after we insert x_i. So during the entire course of the for-loop, each each is traversed at most once.
- Hence the for-loop runs in $O(n)$ time. ■

But as you could see, the triangulation constructed by the Scan algorithm contains many skinny triangles. This is not only an aesthetic deficit but also a practical one: In the context of interpolation, a long and skinny triangle means a long "crease" in the piecewise linear surface, which makes the interpolation "more edgy" and "less smooth".

There should be better ways of triangulation, and at this moment, Delaunay has a say...