# Chapter 1

# Delaunay Triangulation and Voronoi Diagrams

(with material from [1], [3], and [4], pictures are missing)

In this lecture we partition the convex hull induced by a set of points. We *triangulate* the area. A *triangulation* $\mathcal{T}$ of a set of points $P \subseteq \mathbb{R}$ is a decomposition of the convex hull $\mathrm{CH}(P)$ into triangles, so that the vertices of each triangle are points in $P$, and every $p \in P$ is the vertex of some triangle (see Figure **??**) for an example. The property of a triangulation is that any two triangles are either disjoint or share an edge.

How to compute a triangulation? A simple way to obtain one, is to extend the sweep-like algorithm for computing the convex hull: When a point $p$ (in lexicographic order) is *processed*, it "sees" a set of edges of the current convex hull. For each such edge, add a triangle with a third vertex being equal to $p$.

The number of triangles (and edges) is always the same in any triangulation of $P$. It depends on the number of points belonging to $\partial\mathrm{CH}(P)$, the boundary of the convex hull (not needed to be identical to the number of vertices of $\mathrm{CH}(P)$).

**Theorem 1.** *For a set of points $P$ (not all collinear), with $k$ points on $\partial\mathrm{CH}(P)$, $\mathcal{T}(\mathcal{P})$ consists of $2n-2-k$ triangles and $3n-3-k$ edges.*

*Proof.* Let $m$ be the number of triangles. Following there are $n_f := m+1$ two-dimensional faces in $\mathbb{R}^2$ subdivision induced by $\mathcal{T}(\mathcal{P})$. Each of the $m$ triangles has 3 edges, the unbounded face (containing $\mathrm{CH}(P)$ as a hole) has $k$ edges. Every edge is incident to two faces. In total we have $n_e := (3m+k)/2$ edges in $\mathcal{T}$. Euler's formula states: $n - n_e + n_f = 2$ (where $n$ is the number of points and $n_f$ is the number of faces). Putting together: $m = 2n-2-k$, which implies $n_e = 3n-3-k$. □

## 1.1 Representation

It is convenient to only consider the triangles in a data structure that represents a triangulation at once. The problem is the unbounded face that contains $\mathrm{CH}(P)$ as a hole. But there is a nice trick: It is very common to introduce an "infinite" vertex $v_{\inf}$ and "infinite" edges each connecting a vertex of $\mathrm{CH}(P)$ with $v_{\inf}$. Following we also have "infinite" faces, namely those that are incident to $v_{\inf}$ and to end edge of $\mathrm{CH}(P)$. This has implications as now each edge is incident to two faces (triangles) and we decomposed the plane into finitely many triangles, where non-infinite ones are actual triangles of the triangulation.[1] Note: The infinite vertex has no geometric coordinates and no geometric operation can be applied to it, or to any of its incident edges and faces.

A triangulation is usually stored with a vertex-based representation (preferred over edge-based representations, that we see in Lecture **??**). This saves spaces and also results in faster algorithms [2]. Besides vertices, faces are central. Given a triangular face $\Delta$, it provides access to its three vertices $v_1, v_2, v_3$ (in counter-clockwise order) and to the three neighboring faces (triangles) $\Delta_1, \Delta_2, \Delta_3$ (with the convention that $\Delta(v_i)$ is opposite to the vertex $v_i$, and also in counter-clockwise order). Note that for proper triangle all $v_i$ are non-infinite and there is at most one infinite triangle. For an infinite triangle, exactly one vertex is equal to $v_{\inf}$ and two neighboring faces are infinite, too. There are functions to access the vertices' neighbors in clockwise and counterclockwise order. Similar the neighboring faces "rotate" in clockwise- and counterclockwise fashion. Edges are not explicitly stored. The implicit storage relies on the adjacency information of triangles: Each edge has exactly two vertices (in two different triangles) to which it is opposite.

## 1.2 Adding points

The sweep algorithm is not directly able to add points to a triangulation. What to do when a new point should be inserted? We first have to locate the triangle whose interior contains the new point and then split the triangle into three, by adding edges from each vertex to the new point (vertex). In case the new point hits a vertex, we are already done, and if it is contained in the interior of edge, then we split the edge at the point and split the two incident triangles into two by adding for each an edge from its third vertex to the new points (vertex).

The open question is how to locate the triangle (edge/vertex) in (expected) sub-linear time. This can be done, for example, by a line traversal starting in some (given = hint) vertex. The expected search time (if points are uniformly distributed) is $O(\sqrt{n})$.

[1] Actually, the "surface" we obtain is topologically equivalent to the unit-sphere.

More efficient queries can be achieved with a hierarchical triangulation (which usually work best for Delaunay triangulations, see below). At the bottom level, the actual triangulation is stored (to where actual operations are applied). In a higher level, only a random subset of the points in the level below is triangulated. Point location is then done by a top-down nearest-neighbor query starting in the top-most triangulation. For each following level, the next nearest neighbor in that level can be found by start walking from the nearest neighbor found at the preceding level.

## 1.3 Terrains

We want to model the earth's surface (restricted to a piece where the earth's spherical shape is not "critical"). Even then, a purely planar triangulation only suffices for very few countries. Usually we have to model mountains and valleys. For that we consider *terrains*. It is a two-dimensional surface in a three-dimensional space, that is, a function $f : A \subset \mathbb{R}^2 \to \mathbb{R}$. The function assigns a height to every point in the domain $A$. Naturally we do not know all heights of the earth, but only for a finite subset (e.g., by satellite measurements). That is, we want to approximate the height for points we do not have a value. Naively, we can simply assign the function value of the closest point. However, this terrain looks rather ugly, in particular, it is non-continuous (and so very unnatural).

Instead, a triangulation is often used for the interpolation of functions. Assume that for given point set $P \subset \mathbb{R}$ we know function values $f(p)$ for $p \in P$. Out goal is to get "some" value for each point $\tilde{p} \in \mathrm{CH}(P)$. This can be done by first triangulating $P$ and obtain $\mathcal{T}$. In a second step we construct a special surface in $\mathbb{R}^3$ consisting of as many (spatial) triangles as in $\mathcal{T}$, that is, for a planar triangle $\Delta(p, q, r)$ we will have $\Delta'(f(p), f(q), f(r))$ in $\mathbb{R}^3$. Aiming for $\tilde{f}(\tilde{p})$ with $\tilde{p} \in \mathrm{CH}(P)$ (where $\tilde{f}$ is an interpolated value of $f$ at $\tilde{p}$), we first obtain the triangle $\Delta$ in $\mathcal{T}$ that contains $\tilde{p}$ (by a point-location query) and then return the height of $\Delta'$ over $\tilde{p}$. This can be computed using *barycentric coordinates*. That is, if $\tilde{p} \in \Delta = (p, q, r)$, there exists $c_p, c_q, c_r$ with $c_p + c_q + c_r = 1$, such that $\tilde{p} = c_p p + c_q q + c_r r$. Then, $\tilde{f}(\tilde{p})$ is set to be equal to $c_p f(p) + c_q f(q) + c_r f(r)$. Such a terrain is polyhedral, a piecewise linear function.

An open question is: How to triangulate $P$? The problem is that we do not know the original terrain, so a definite answer is hard to obtain. Theoretically, all possible triangulations seem to be equally good. However, some of the final terrains look more "natural". Luckily there is only a finite set of triangulations, so we can aim for a "nice" one. But what does it mean for a triangulation to be nice.

PICTURE

Skinny triangles, that is, triangles with (very) small angles lead to inter-

polated function values whose interpolation points are very far away. That is, we aim for a more balanced situation.[2] That is we aim for "fat" triangle over "skinny" ones. More formally, it would be nice to maximize minimal angles.

## 1.4 Delaunay

**Definition 2.** *A Delaunay triangulation (DT) is a triangulation of a point set $P$, where the circumcircle of each triangle contains no other point of $P$.*

We first consider the $3m$ angles in a given triangulation $\mathcal{T}$. Let sort them in a vector $A(\mathcal{T})$ in increasing order $(\alpha_1, \ldots, \alpha_{3m})$, the *angle-vector* of $\mathcal{T}$. For two triangulations $\mathcal{T}, \mathcal{T}'$ we can compare the angle vectors (of same size) lexicographically, that is, $A(\mathcal{T}) > \mathcal{A}(\mathcal{T}')$ if $\exists i$ with $1 \leq i \leq 3m$ such that $\alpha_i = \alpha'_i$ for all $j < i$ and $\alpha_i > \alpha'_i$. $\mathcal{T}$ is angle-optimal if $A(\mathcal{T}) \geq \mathcal{A}(\mathcal{T}')$ for all $\mathcal{T}'$. We often aim for such a triangulation.

Consider two adjacent (proper) triangles that form a convex quadrilateral, for example for two triangles $(p_i, p_j, p_k)$, $(p_i, p_l, p_l)$ adjacent to an edge $e = \overline{p_i p_j}$ of some $\mathcal{T}(\mathcal{P})$ that is not part of $\partial\mathrm{CH}(P)$. The angle vector has six entries $A = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$. The quadrilateral can be re-triangulated (here we make use of its convexity) by exchanging the diagonal $\overline{p_i p_j}$ by the diagonal $\overline{p_k p_l}$ — a so-called *flip*. Let $A' = (\alpha'_1, \alpha'_2, \alpha'_3, \alpha'_4, \alpha'_5, \alpha'_6)$ denote the new angle-vector. If $A' > A$, then the minimum angle in a triangle is increased.

### 1.4.1 Properties

**Theorem 3.** *Every finite point set in $\mathbb{R}^2$ has a Delaunay triangulation.*

The proof is actually the basis for the flipping algorithm.

*Proof.* The proof is three-step.

- We start with a convex quadrilateral. There are only two triangulations. Using Thales' theorem, we can show that exactly one of the diagonals is an illegal.

- In any non-Delaunay triangulation $\mathcal{T}$ of $P$ search for an edge that is an illegal diagonal in a convex quadrilateral. These two triangles are locally non-Delaunay. Assuming the (1), we flip the diagonals, which increases the angle-vector of the quadrilateral and so of $\mathcal{T}$. As there are only finitely many triangulations, the process stops.

[2] Assuming that the original terrain is continuous and the local feature size (to be defined) is not too small.

- Consider a triangle $\Delta(p, q, r)$ and another point $s$ in one of the chords defined by the triangle and its circumcircle - say the one defined by $\overline{pq}$. There is an adjacent triangle $\Delta(p, q, t)$. If $t$ is in that chord, we found a pair of triangles with an illegal edge. Otherwise, $s$ must also be contained in the circumcircle of $\Delta(p, q, t)$, but it is closer to a point of $\Delta(p, q, t)$ than to any point of $\Delta(p, q, r)$. Now we can repeat the argument with $\Delta(p, q, t)$ and $s$ - but only a finite number of times, where we have to arrive at the first case.

□

If no four points in $P$ lie on a common circle, the Delaunay triangulation is a proper triangulation. Otherwise, there may be faces with more than three sides (needs further triangulation). A DT also optimizes other criteria. The maximization of the minimal-angle is direct (or more precisely: a DT lexicographically maximizes the angles from smallest to largest). Beyond, one can show that the maximum radius of a circumcircle is minimized (and more).

### 1.4.2 The flipping algorithm

Several algorithms exists to compute a DT: Sweep, incremental insertion, divide-and-conquer. A very simple one is based on the observation made above, namely to make a triangulation locally more and more "Delaunay" by flipping edges until no further improvements are possible. That is also why we call it the *flipping algorithm*. It has a worst-case running time of $O(n^2)$ (which is inferior to others), but we study the usage of the side-of-circle predicate which is central to all algorithms. So we have chosen this simple algorithm.

We initially compute any triangulation $\mathcal{T}(\mathcal{P})$, for example, using a sweep approach. Let $e = \overline{p_i p_j}$ be an non-convex-hull edge. With $Q_e = p_i p_l p_j p_l$ we denote the quadrilateral defined by the two adjacent triangles that share $e$. If $Q_e$ is convex, the edge $e$ is said to be *illegal* if each triangle circumcircle contains the missing fourth point. Note that here we utilize the side-of-circle predicate discussed in the tutorials. If $e$ is illegal, we *flip diagonals in* $Q_e$, that is, we replace $e$ by the other diagonal $\overline{p_k p_l}$.

**Algorithm 1** Flipping algorithm

```
Obtain an initial triangulation T
while T is not Delaunay do
  find quadrilateral Q_e with illegal edge e
  flip diagonals in Q_e
end while
```

The algorithm terminates for two reasons: There are only finitely many different triangulations and, in each step we make the angle-vector larger. There is also another proof which relates a DT to a convex hull in $\mathbb{R}^3$. We present the idea and the theorem below.

In an actual implementation we would maintain a queue of potentially illegal edges. An edge not in the queue is either an edge of $\mathrm{CH}(P)$ or the two adjacent triangles form a non-convex quadrilateral, or the convex quadrilateral of them is already locally Delaunay. We initialize the queue with all interior edges of an initial triangulation. In each step, we remove one edge of the queue, and if we flip diagonals in a convex quadrilateral, its outer non-convexhull edges are added to the queue.

**Algorithm 2** Flipping algorithm with queue

```
Obtain an initial triangulation T
Initialize a queue with all non-convexhull edges
while queue is not empty do
  remove front edge e
  if Q_e is reversed then
    flip diagonals in Q_e
    append non-convexhull outside edges p_ip_l, p_lp_j, p_jp_k, p_kp_i to queue
  end if
end while
```

### 1.4.3 More properties

Two other properties are also useful: Given two points $p_1, p_2 \in P$. There distance along edges of $DT(P)$ is at most $2.42 \cdot d(p_1, p_2)$, where $d(\cdot, \cdot)$ denotes Euclidean distance in $\mathbb{R}^2$.

We also mention that the minimum spanning tree of $P$ ($\mathrm{MST}(P)$) is a subgraph of $\mathrm{DT}(P)$. Following, in order to compute the MST one can start with computing a DT and compute the tree only on DT's edges, instead of starting with a full set.

### 1.4.4 Delaunay triangulation via convex hull

Consider the map:

$$\ell : \mathbb{R}^2 \to \mathbb{R}^3 \text{ with } \ell(p_x, p_y) = (p_x, p_y, p_x^2, p_y^2)$$

that is, we map any point in the plane onto the paraboloid centered at the origin with equation $z = x^2 + y^2$. Then $\mathrm{DT}(P)$ equals the projection of the lower convex hull of the lifted points $(p_{ix}, p_{iy}, p_{ix}^2 + p_{iy}^2)$, $1 \leq i \leq n$.

*Sketch of proof:* Consider three points $p, q, r \in P$ sorted counter-clockwisely on a circle. The triangle $\Delta(p, q, r)$ is part of $\mathrm{DT}(P)$ if there is no $s \in P' :=$

$P \setminus \{p, q, r\}$ lying in the circumcircle of $\Delta(p, q, r)$. This is equivalent to say that side-of-circle$(p, q, r, s) > 0$ for all $s \in P'$. We next use the three-dimensional orientation predicate (without proof): Given for four points $p', q', r', s' \in \mathbb{R}^3$ and $p', q', r'$ define a plane $E'$:

$$\text{orientation}(p', q', r', s') = \text{sign}\det\begin{vmatrix} 1 & p'_x & p'_y & p'_z \\ 1 & q'_x & q'_y & q'_z \\ 1 & r'_x & r'_y & r'_z \\ 1 & s'_x & s'_y & s'_z \end{vmatrix} = \begin{cases} -1, & s \text{ lies below } E' \\ 0, & s \text{ is contained in } E' \\ 1, & s \text{ lies above } E' \end{cases}$$

Note that the side-of-circle-predicate is identical to this orientation predicate for the chosen paraboloid. If we write orientation$(p, q, r, s) > 0$ for all $s \in P'$, we know that $\ell(s)$ lies above the plane defined by the three points $\ell(p), \ell(q), \ell(r)$. But this is just expressing that $\Delta(\ell(p), \ell(q), \ell(r))$ is a facet of the lower hull of $\ell(P)$. □

We can exploit this "dualism" a little bit further: If a convex hull is unique, then the Delaunay triangulation must also be unique, as every convex hull can be mapped onto a Delaunay triangulation. If we consider again a convex quadrilateral, its Delaunay triangulation corresponds to the lower hull of the lifted points, while the flipped (illegal) triangulation corresponds to the upper hull (play with the predicates to see it). An arbitrary triangulation corresponds to a surface spanned between the points lifted onto the paraboloid. Flipping illegal diagonals transforms this surface towards the lower hull of the lifted point set.

## 1.5 Voronoi Diagrams

There is also a relation of Delaunay triangulation and the Voronoi diagram. What is the Voronoi diagram? Think of being a fast-food company and you want to open a new store at a certain location. Obviously you want to be successful, so it would be nice to estimate the number of customers it can reach. That is, to answer a question like: To which store does a customer travel? Or if you are a cell-phone company, and you want to install a new antenna. Then the question is, in which area around the antenna do cellular phones connect to this new antenna? To answer such question, there are Voronoi diagrams in computational geometry. A Voronoi diagram is a partitioning of a space (often $\mathbb{R}^2$) induced by a set of sites $P$. This is often a rather simplified assumptions as costs to reach a site (e.g., a store) may vary or prices are in different sites are not really identical. But assuming so, gives a at least a good estimate.

Let us first give some notation. Let $d(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$ the Euclidean distance between two points in the plane. Furthermore, the

set of sites $P$ consists of $n$ points $p_1, \ldots, p_n$. The *Voronoi diagram Vor*$(P)$ is the subdivision of $\mathbb{R}^2$ into $n$ cells, namely one for each site $\mathcal{V}(p_i)$. Each cell is having the property that for any point $q \in \mathcal{V}(p_i)$, we have $d(q, p_i) < d(q, p_j)$ for any $j \neq i$. The *bisector* $B(p_i, p_j) := \{x \in \mathbb{R}^2 | d(x, p_i) = d(x, p_j)\}$ is the set of points that have equal distance to two sites. In case of sites being points the bisector is formed by a line perpendicular to the segment $\overline{p_i p_j}$. It splits the plane into two half-planes. The plane that contains $p_i$ is denoted by $h(p_i, p_j)$, the other that contains $p_j$ is called $h(p_j, p_i)$. It is clear that $q \in h(p_i, p_j)$ if $d(q, p_i) < d(q, p_j)$. This observation allows us to define the set of a Voronoi cell: $\mathcal{V}(p_i) = \bigcap_{1 \leq j \leq n, j \neq i} h(p_i, p_j)$. One can immediately see that the boundary of a (possibly unbounded) Voronoi cell consists of up $n - 1$ vertices and $n - 1$ edges. Such a cell is also convex.

But what about the entire Voronoi diagram? Let us first introduce an opposite definition: $\mathcal{C}(x) := \{p \in P | d(x, p) < d(x, q) \text{ with } q \in P\}$. This set comprises all points of $P$ which minimize the distance to a given $x \in \mathbb{R}^2$. In general, the cardinality is one, except for points $x$ contained in a bisector (or intersections of bisectors). The Voronoi diagram is exactly the set of points, for which the cardinality of $\mathcal{C}$ is larger than one: $\text{Vor}(P) := \{x \in \mathbb{R}^2 : |\mathcal{C}(x)| \geq 2\}$. A vertex $v$ of $\text{Vor}(P)$ is a point with $|\mathcal{C}(x)| \geq 3$, edges have cardinality 2 and cells only 1.

**Theorem 4.** *For Vor*$(P)$ *it holds:*

1. *$q$ is a vertex of Vor$(P)$ iff largest empty circle $C_P(q)$ contains at least three points of $P$.*
2. *$B(p_i, p_j)$ defines an edge $e$ of Vor$(P)$ iff there is $q \in e$ such that $C_P(q)$ contains $p_i$ and $p_j$ but no other site.*

*Proof.* 1. Assume there is $q$ with $C_P(q)$ containing three sites $p_i, p_j, p_k$. Since the circle's interior is empty, $q$ must be on the boundary of each $\mathcal{V}(p_i)$, $\mathcal{V}(p_j)$, and $\mathcal{V}(p_k)$ and so $q$ must be a vertex of $\text{Vor}(P)$.

In the other direction, each vertex $q$ is incident to at least three edges and so to at least three proper Voronoi cells. So, $q$ is equidistant to at least the three sites of the three cell, and no other site can be closer to $q$, as so $q$ would not be incident. Following, the interior of a circle with three sites on its boundary does not contain any other site.

2. Assume such a $q$. We obviously have $d(q, p_i) = d(q, p_j) \leq d(q, p_k)$ for all $k$. Following $q$ is on an edge or a vertex of $\text{Vor}(P)$. But we just have seen that it cannot be a vertex, as three sites must exists on the circle. Hence, only the stated case remains.

   Conversely, let the edge $e$ be defined by $B(p_i, p_j)$. It is clear that the largest empty circle of centered at any point on the edge's interior must contain $p_i$ and $p_j$ on its boundary, and no other site.

□

### 1.5.1 Voronoi and Delaunay

There is a one-to-one relation between the Voronoi cells and the cells of the Delaunay triangulation. A vertex of the Voronoi diagram is a point where at least three sites (points of $P$) are equally distant. They are centrepoints for circumcenters in the Delaunay triangulation. Or said differently: One can draw a circle through these three points, such that no other point of $P$ is contained it the circle. That is, a Voronoi vertex corresponds to an open polygonal region in the Delaunay triangulation, an edge of the Voronoi diagram corresponds to a Delaunay edge, and a open polygonal region of the Voronoi diagram corresponds to a vertex in the Delaunay triangulation (i.e., a site itself). The correspondence is also valid for $d$-dimensional spaces, with the property that dimensions of related Delaunay and Voronoi regions always sum up to $d$.

### 1.5.2 Computing a Voronoi Diagram

Using the connection between Delaunay triangulations and convex hulls, any convex-hull algorithm for $\mathbb{R}^3$ can be used to obtain DT in $\mathbb{R}^2$, such as incremental insertion, divide-and-conquer, or gift-wrapping (we have not seen those in the lecture). The Voronoi diagram itself can be computed in linear time from DT using the one-to-one correspondence (requires to compute rational coordinates).

Beyond those, there is Fortune's sweep line algorithm ($O(n \log n)$) to compute the Voronoi diagram directly. It is already optimal, as sorting $n$ numbers can be reduced to the Voronoi problem and following the tight $\Omega(n \log n)$ lower bound holds as well. We omit details on the algorithm, as we first introduce the sweep for another problem in the next lecture. Anyhow, the interested reader is referred to the nice introduction given in [1, Chapter 7.2].

### 1.5.3 Misc

There are non-trivial subtleties to consider, if $P$ is not in general position.

Some more theorems:

**Theorem 5.** *If all $p \in P$ are collinear, $Vor(P)$ consists of $n-1$ parallel lines and $n$ cells. Otherwise, it is connected and its edges are either segments of half-lines.*

**Theorem 6.** *$Vor(P)$ (with $|P| \geq 3$) has at most $2n-5$ vertices and not more than $3n-6$ edges.*

Voronoi diagrams exists in many spaces, with many site-types, and many different distance functions. All are beyond the scope of this overview. More can be found in [1, Chapter 7] and [3, Chapter 23].

# Bibliography

[1] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark H. Overmars. *Computational geometry*. Springer, Berlin [u.a.], 3., ed. edition, 2008.

[2] Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, and Mariette Yvinec. Triangulations in cgal. *Comput. Geom. Theory Appl.*, 22:5–19, 2002.

[3] Jacob E. Goodman and Joseph O'Rourke. *Handbook of discrete and computational geometry*, volume - of *Discrete mathematics and its applications*. Chapman & Hall/CRC, Boca Raton, 2nd. ed. edition, 2004.

[4] Mariette Yvinec. 2d triangulations. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 edition, 2009.