

Triangulations & Voronoi diagrams

Lesson 03*


1	The Voronoi Diagram	1
2	The Delaunay Triangulation	3
2.1	Convex Hull	3
2.2	Local Optimality	3
2.3	Angle Optimality	4
2.4	Lifting on the paraboloid	4
2.5	Degeneracies	5
3	Duality between the DT and the VD	6
4	Incremental construction of the DT	6
5	Data structures for storing a DT	9
6	Constrained and Conforming Delaunay Triangulations	10
7	Triangulation of a polygon	12
7.1	The trivial case of a convex polygon	12
7.2	Greedy algorithm for polygons with holes	13
7.3	Ear clipping for polygons without holes	13
8	Notes and comments	14

1 The Voronoi Diagram

Let S be a set of points in \mathbb{R}^2 (the two-dimensional Euclidean space). The Voronoi cell of a point $p \in S$, defined \mathcal{V}_p , is the set of points $x \in \mathbb{R}^2$ that are closer to p than to any other point in S ; that is:

$$\mathcal{V}_p = \{x \in \mathbb{R}^2 \mid \|x - p\| \leq \|x - q\|, \forall q \in S\}. \quad (1)$$

The union of the Voronoi cells of all generating points $p \in S$ form the Voronoi diagram of S , defined $\text{VD}(S)$. If S contains only two points p and q , then $\text{VD}(S)$ is formed by a single line defined by all the points $x \in \mathbb{R}^2$ that are equidistant from p and q . This line is the perpendicular

* Hugo Ledoux, Ravi Peters, Ken Arroyo Ohori. This work is licensed under a Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>) (last update: November 2, 2018)

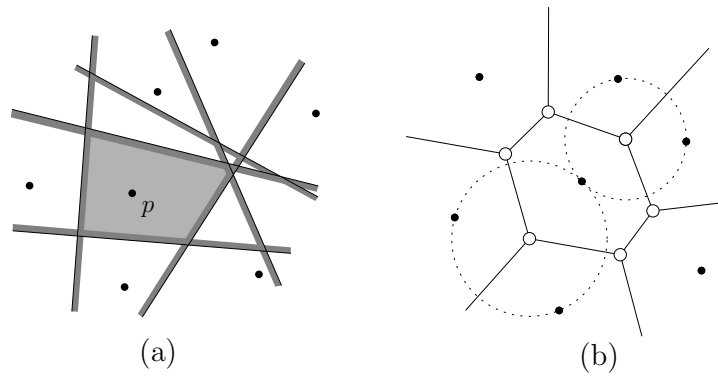


Figure 1: **(a)** The Voronoi cell \mathcal{V}_p is formed by the intersection of all the half-planes between p and the other points. **(b)** The VD for a set S of points in the plane (the black points). The Voronoi vertices (white points) are located at the centre of the circle passing through three points in S , provided that this circle contains no other points in S in its interior.

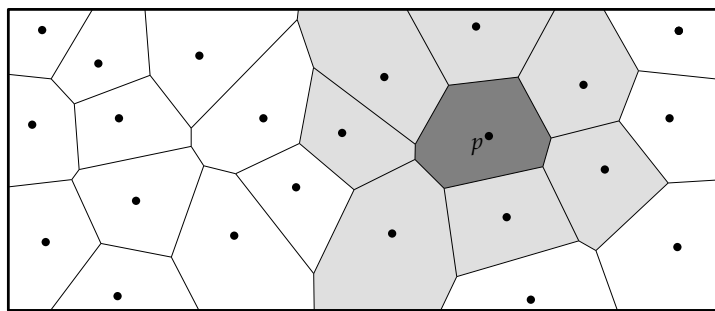


Figure 2: VD of a set of points in the plane (clipped by a box). The point p (whose Voronoi cell is dark grey) has seven neighbouring cells (light grey).

bisector of the line segment from p to q , and splits the plane into two half-planes. \mathcal{V}_p is formed by the half-plane containing p , and \mathcal{V}_q by the one containing q . As shown in Figure 1a, when S contains more than two points (let us say it contains n points), the Voronoi cell of a given point $p \in S$ is obtained by the intersection of $n - 1$ half-planes defined by p and the other points $q \in S$. That means that \mathcal{V}_p is always convex. Notice also that every point $x \in \mathbb{R}^2$ has at least one nearest point in S , which means that $\text{VD}(S)$ covers the entire space.

As shown in Figure 1b, the VD of a set S of points in \mathbb{R}^2 is a planar graph. Its edges are the perpendicular bisectors of the line segments of pairs of points in S , and its vertices are located at the centres of the circles passing through three points in S . The VD in \mathbb{R}^2 can also be seen as a two-dimensional cell complex where each 2-cell is a (convex) polygon (see Figure 2). Two Voronoi cells, \mathcal{V}_p and \mathcal{V}_q , lie on the opposite sides of the perpendicular bisector separating the points p and q .

The VD has many interesting properties, what follows is a list of the most relevant properties in the context of this course.

Size: if S has n points, then $\text{VD}(S)$ has exactly n Voronoi cells since there is a one-to-one mapping between the points and the cells.

Voronoi vertices: a Voronoi vertex is equidistant from 3 data points. Observe for instance in Figure 1b that the Voronoi vertices are at the centre of circles.

Voronoi edges: a Voronoi edge is equidistant from 2 points.

Convex hull: let S be a set of points in \mathbb{R}^2 , and p one of its points. \mathcal{V}_p is unbounded if p

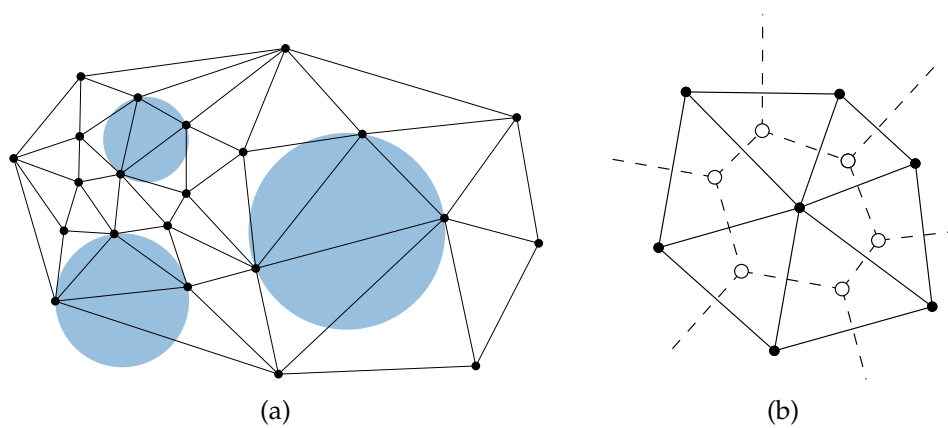


Figure 3: **(a)** The DT of a set of points in the plane (same point set as Figure 2). **(b)** Both the DT (black lines) and the VD (dashed lines) of a set of points in the plane.

bounds $\text{conv}(S)$. Otherwise, \mathcal{V}_p is the convex hull of its Voronoi vertices. Observe that in Figure 1b, only the point in the middle has a bounded Voronoi cell.

2 The Delaunay Triangulation

Let \mathcal{D} be the VD of a set S of points in \mathbb{R}^2 . Since $\text{VD}(S)$ is a planar graph, it has a dual graph, and let \mathcal{T} be this dual graph obtained by drawing straight edges between two points $p, q \in S$ if and only if \mathcal{V}_p and \mathcal{V}_q are adjacent in \mathcal{D} . Because the vertices in \mathcal{D} are of degree 3 (3 edges connected to it), the graph \mathcal{T} is a triangulation. \mathcal{T} is actually called the Delaunay triangulation (DT) of S , and, as shown in Figure 3a, partitions the plane into triangles—where the vertices of the triangles are the points in S generating each Voronoi cell—that satisfy the *empty circumcircle* test (a circle is said to be *empty* when no points are in its interior). If S is in general position, then $\text{DT}(S)$ is unique.

2.1 Convex Hull

The DT of a set S of points subdivides completely $\text{conv}(S)$, ie the union of all the triangles in $\text{DT}(S)$ is $\text{conv}(S)$.

Let S be a set of points in \mathbb{R}^2 , the *convex hull* of S , denoted $\text{conv}(S)$, is the minimal convex set containing S . It is best understood with the elastic band analogy: imagine each point in \mathbb{R}^2 being a nail sticking out of the plane, and a rubber band stretched to contain all the nails, as shown in Figure 4. When released, the rubber band will assume the shape of the convex hull of the nails. Notice that $\text{conv}(S)$ is not only formed by the edges connecting the points (the rubber band), but all the points of \mathbb{R}^2 that are contained within these edges (thus the whole polygon).

2.2 Local Optimality

Let \mathcal{T} be a triangulation of S in \mathbb{R}^2 . An edge σ is said to be *locally* Delaunay if it either:

- (i) belongs to only one triangle, and thus bounds $\text{conv}(S)$, or
- (ii) belongs to two triangles τ_a and τ_b , formed by the vertices of σ and respectively the vertices p and q , and q is outside of the circumcircle of τ_a .

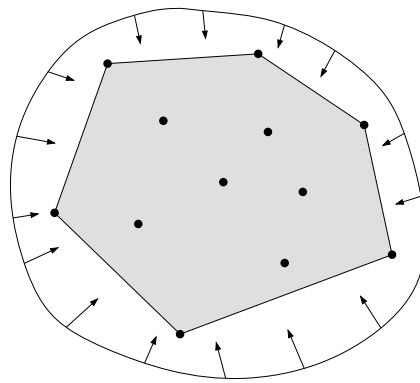


Figure 4: The convex hull of a set of points in \mathbb{R}^2 .

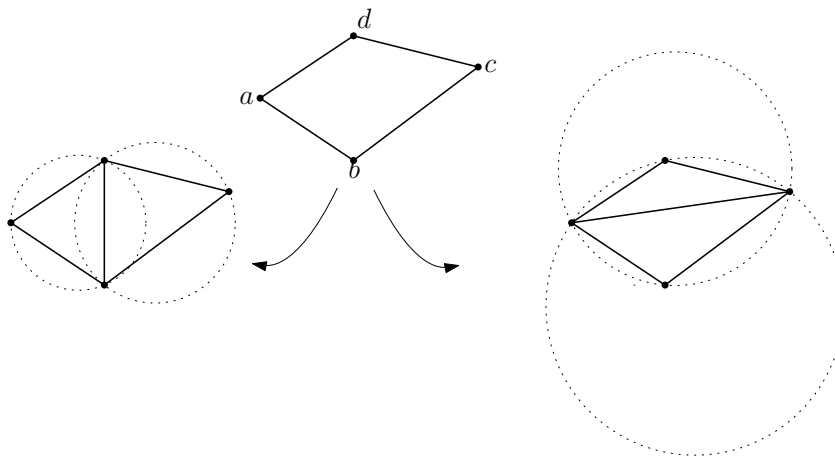


Figure 5: Only one configuration is Delaunay (the left one).

The second case is illustrated in Figure 5: the triangles abd and bcd are Delaunay, and thus is the edge bd ; the edge ac is not Delaunay. In an arbitrary triangulation, not every edge that is locally Delaunay is necessarily a edge of $DT(S)$, but local optimality implies globally optimality in the case of the DT:

Let \mathcal{T} be a triangulation of a point set S in \mathbb{R}^2 . If every edge of \mathcal{T} is locally Delaunay, then \mathcal{T} is the Delaunay triangulation of S .

This has serious implications as the DT—and its dual—are locally modifiable, ie we can theoretically insert, delete or move a points in S without recomputing $DT(S)$ from scratch.

2.3 Angle Optimality

The DT in two dimensions has a very important property that is useful in applications such as finite element meshing or interpolation: the *max-min angle optimality*. Among all the possible triangulations of a set S of points in \mathbb{R}^2 , $DT(S)$ maximises the minimum angle (max-min property), and also minimises the maximum circumradii. In other words, it creates triangles that are as equilateral as possible. Notice here that maximising the minimum angle is not the same as minimising the maximum, and the DT only guarantees the former.

2.4 Lifting on the paraboloid

There exists a close relationship between DTs in \mathbb{R}^d and convex polytopes in \mathbb{R}^{d+1} .

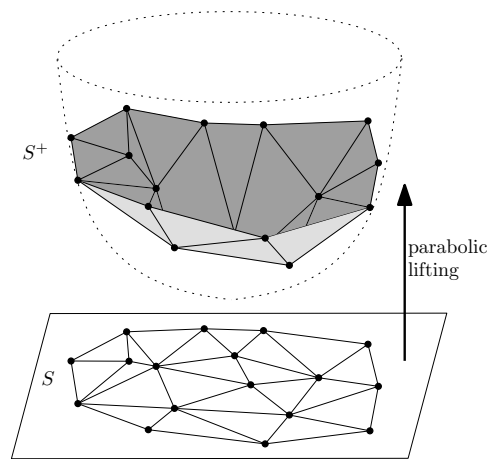


Figure 6: The parabolic lifting map for a set S of points \mathbb{R}^2 .

Let S be a set of points in \mathbb{R}^d , and let x_1, x_2, \dots, x_d be the coordinates axes. The parabolic lifting map projects each vertex $v(v_{x1}, v_{x2}, \dots, v_{xd})$ to a vertex $v^+(v_{x1}, v_{x2}, \dots, v_{xd}, v_{x1}^2 + v_{x2}^2 + \dots + v_{xd}^2)$ on the paraboloid of revolution in \mathbb{R}^{d+1} . The set of points thus obtained is denoted S^+ . Observe that, for the two-dimensional case, the paraboloid in three dimensions defines a surface whose vertical cross sections are parabolas, and whose horizontal cross sections are circles; the same ideas are valid in higher dimensions.

The relationship is the following: every facet (a d -dimensional simplex) of the lower envelope of $\text{conv}(S^+)$ projects to a d -simplex of the Delaunay triangulation of S . This is illustrated in Figure 6 for the construction of the DT in \mathbb{R}^2 .

In short, the construction of the d -dimensional DT can be transformed into the construction of the convex hull of the lifted set of points in $(d + 1)$ dimensions. In practice, since it is easier to construct convex hulls (especially in higher dimensions, ie 4+), the DT is often constructed with this method.

2.5 Degeneracies

The previous definitions of the VD and the DT assumed that the set S of points is in general position, ie the distribution of points does not create any ambiguity in the two structures. For the VD/DT in \mathbb{R}^2 , the degeneracies, or special cases, occur when 3 points lie on the same line and/or when 4 points are cocircular. For example, in two dimensions, when four or more points in S are cocircular there is an ambiguity in the definition of $\text{DT}(S)$. As shown in Figure 7, the quadrilateral can be triangulated with two different diagonals, and an arbitrary choice must be made since both respect the Delaunay criterion (points should not be on the interior of a circumcircle, but more than three can lie directly on the circumcircle).

This implies that in the presence of four or more cocircular points, $\text{DT}(S)$ is not unique. Notice that even in the presence of cocircular points, $\text{VD}(S)$ is still unique, but it has different properties. For example, in Figure 7, the Voronoi vertex in the middle has degree 4 (remember that when S is in general position, every vertex in $\text{VD}(S)$ has degree 3). When three or more points are collinear, $\text{DT}(S)$ and $\text{VD}(S)$ are unique, but problems with the implementation of the structures can arise.

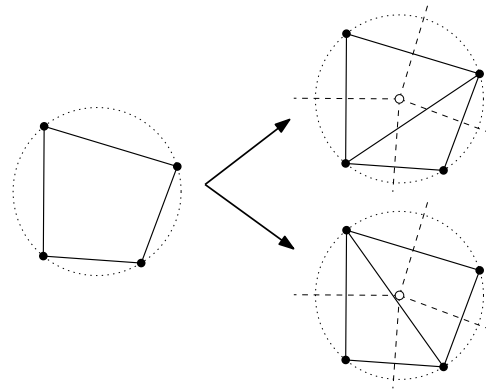


Figure 7: The DT for four cocircular points in two dimensions is not unique (but the VD is).

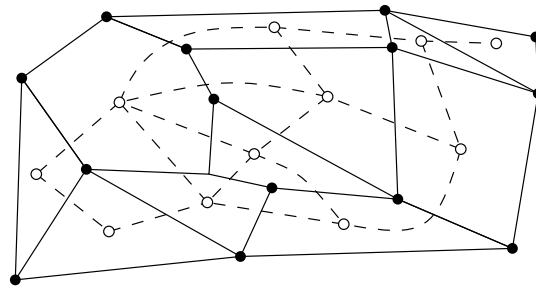


Figure 8: A graph G (black lines), and its dual graph G^* (dashed lines).

3 Duality between the DT and the VD

Duality can have many different meanings in mathematics, but it always refers to the translation or mapping in a one-to-one fashion of concepts or structures. We use it in this course in the sense of the dual graph of a given graph. Let G be a planar graph, as illustrated in Figure 8 (black edges). Observe that G can also be seen as a cell complex in \mathbb{R}^2 . The duality mapping is as follows (also shown in details in Figure 9) The dual graph G^* has a vertex for each face (polygon) in G , and the vertices in G^* are linked by an edge if and only if the two corresponding dual faces in G are adjacent (in Figure 8, G^* is represented with dashed lines). Notice also that each polygon in G^* corresponds to a vertex in G , and that each edge of G is actually dual to one edge (an arc in Figure 8) of G^* (for the sake of simplicity the dual edges to the edges on the boundary of G are not drawn).

The VD and the DT are the best example of the duality between plane graphs.

4 Incremental construction of the DT

Since the VD and the DT are dual structures, the knowledge of one implies the knowledge of the other one. In other words, if one has only one structure, she can always extract the other one. Because it is easier, from an algorithmic and data structure point of view, to manage triangles over arbitrary polygons (they have a constant number of vertices and neighbours), constructing and manipulating a VD by working only on its dual structure is simpler and usually preferred. When the VD is needed, it is extracted from the DT. This has the additional advantage of speeding up algorithms because when the VD is used directly intermediate Voronoi vertices—that will not necessarily exist in the final diagram—need to be computed and stored.

While there exists different strategies to construct a DT, we focus here on the *incremental*

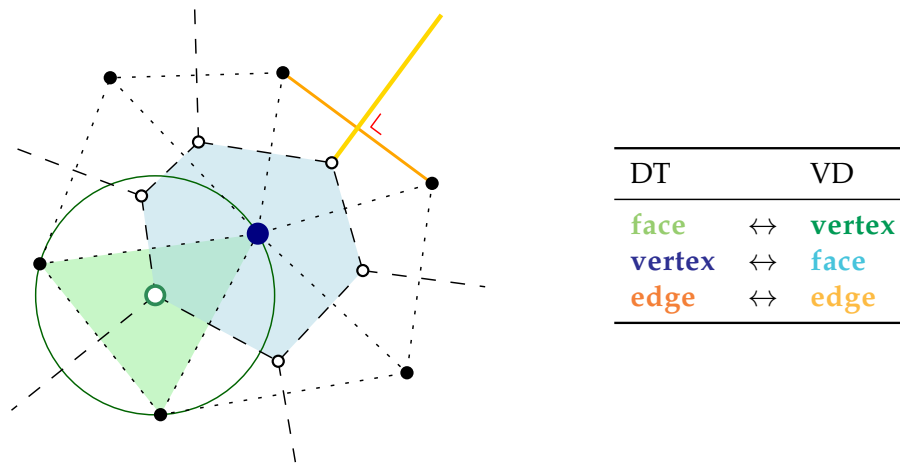


Figure 9: Duality between the DT (dotted) and the VD (dashed).

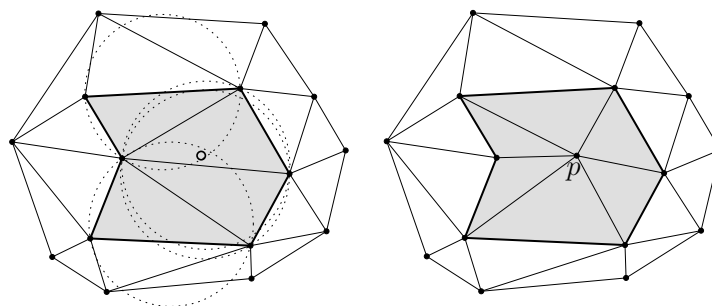


Figure 10: The DT before and after a point p has been inserted. Notice that the DT is updated only locally (only the shaded part of the triangulation is affected).

method (since it is easier to understand and implement). An incremental algorithm is one where the structure is built incrementally; in our case this means that each point is inserted one at a time in a valid DT and the triangulation is updated, with respect to the Delaunay criterion (empty circumcircle), after each insertion. Observe that the insertion of a single point p in a DT modifies only locally the DT, ie only the triangles whose circumcircle contains p need to be deleted and replaced by new ones respecting the Delaunay criterion (see Figure 10 for an example). In sharp contrast to this, other strategies to construct a DT (eg divide-and-conquer and plane sweep algorithms, see Section 8), build a DT in *one* operation (this is a batch operation), and if another point needs to be inserted after this, the whole construction operation must be done again from scratch. That hinders their use for some applications where new data coming from a sensor would have to be added.

Figure 11 describes the algorithm, and Algorithm 1 its pseudo-code. In a nutshell, for the insertion of a new point p in a $DT(S)$, the triangle τ containing p is identified and then split into three new triangles by joining p to every vertex of τ . Second, each new triangle is tested—according to the Delaunay criterion—against its opposite neighbour (with respect to p); if it is not a Delaunay triangle then the edge shared by the two triangles is *flipped* (see below) and the two new triangles will also have to be tested later. This process stops when every triangle having p as one of its vertices respects the Delaunay criterion.

Walk/Point location. To find the triangle containing the new inserted point p , we can use the point-in-polygon test for every triangle (as explained in class), but that would be too slow. An better alternative, is to use the adjacency relationships between the triangles, and use the

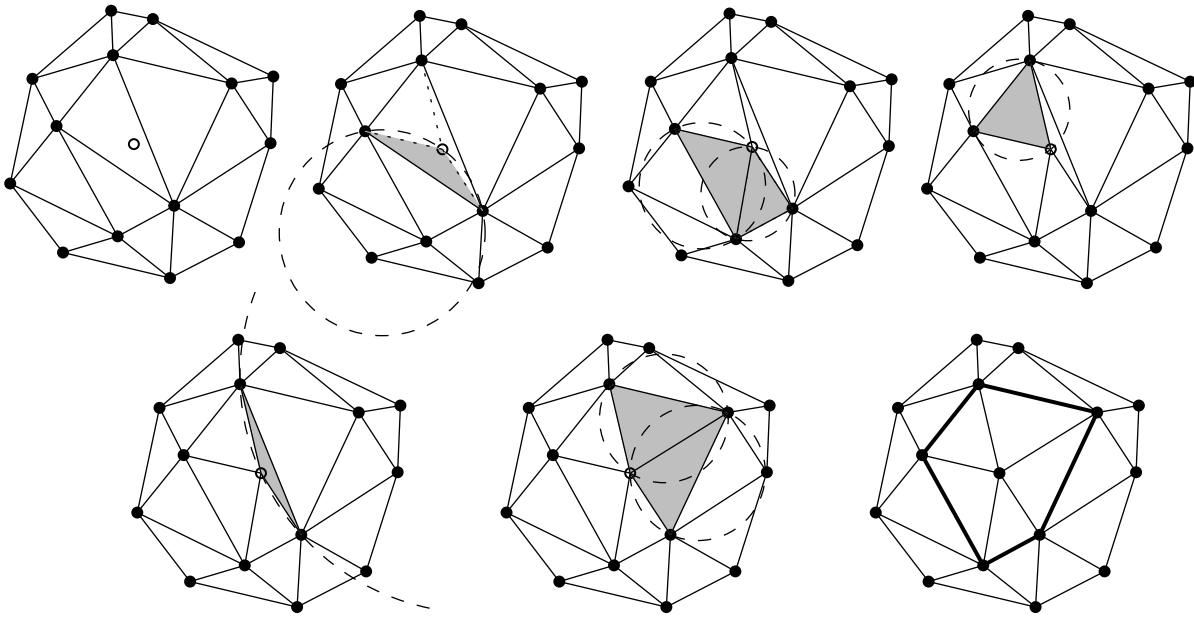


Figure 11: Step-by-step insertion, with flips, of a single point in a DT in two dimensions.

Algorithm 1: Algorithm to insert one point in a DT

Input: A DT(S) \mathcal{T} , and a new point p to insert

Output: $\mathcal{T}^p = \mathcal{T} \cup \{p\}$ // the DT with point p

- 1 find triangle τ containing p ;
 - 2 insert p in τ by splitting it in to 3 new triangles;
 - 3 push 3 new triangles on a stack;
 - 4 **while** stack is non-empty **do**
 - 5 $\tau = \{p, a, b\} \leftarrow$ pop from stack;
 - 6 $\tau_a = \{a, b, c\} \leftarrow$ get adjacent triangle of τ having the edge ab ;
 - 7 **if** c is inside circumcircle of τ **then**
 - 8 flip the triangles τ and τ_a ;
 - 9 push 2 new triangles on stack;
-

side tests as described in the book to navigation from one triangle to the other, until we find the the good one. The idea is shown in Figure 12

Flips. The *flip* operation we use to modify the triangulation is a simple local topological operation that modifies the configuration of two adjacent triangles. Consider the set $S = \{a, b, c, d\}$ of points in the plane forming a quadrilateral, as shown in Figure 13. There exist exactly two ways to triangulate S : the first one contains the triangles abc and bcd ; and the second one contains the triangles abd and acd . Only the first triangulation of S is Delaunay because d is outside the circumcircle of abc . A *flip* is the operation that transforms the first triangulation into the second, or vice-versa.

Controlling the triangles. To control which triangles have to be checked, we use a stack. When the stack is empty, then there are no more triangles to be tested, and we are guaranteed that all the triangles in the triangulation have an empty circumcircle.

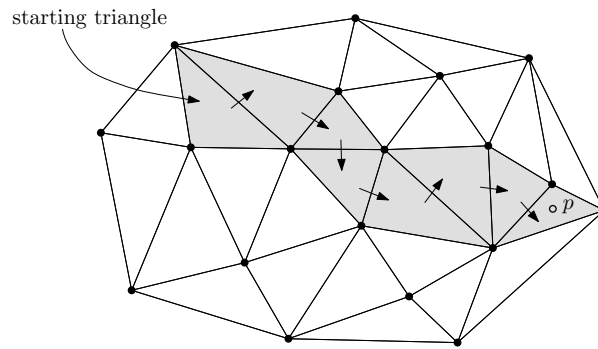


Figure 12: The Walk algorithm for a DT in two dimensions. The query point is p .

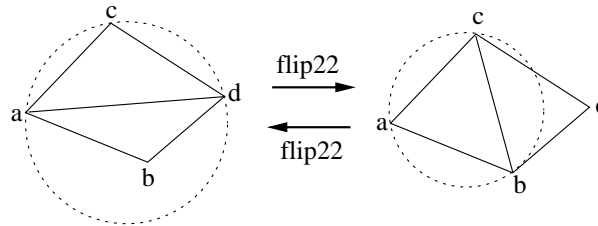


Figure 13: A flip22

Predicates Constructing a DT and manipulating it essentially require two basic geometric tests (called *predicates*): **ORIENTATION** determines if a point p is left, right or lies on the line segment defined by two points a and b ; and **INCIRCLE** determines if a point p is inside, outside or lies on a circle defined by three points a , b and c . Both tests can be reduced to the computation of the determinant of a matrix:

$$\text{ORIENTATION}(a, b, p) = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ p_x & p_y & 1 \end{vmatrix} \tag{2}$$

$$\text{INCIRCLE}(a, b, c, p) = \begin{vmatrix} a_x & a_y & a_x^2 + a_y^2 & 1 \\ b_x & b_y & b_x^2 + b_y^2 & 1 \\ c_x & c_y & c_x^2 + c_y^2 & 1 \\ p_x & p_y & p_x^2 + p_y^2 & 1 \end{vmatrix} \tag{3}$$

5 Data structures for storing a DT

A triangulation is simply a subdivision of the plane into polygons, and thus any data structure used in GIS can be used to store a triangulation.

Simple Features: while many use this (PostGIS and any triangulation you see in Shapefiles), this is not very smart: (1) the topological relationships between the triangles are not stored; (2) the vertices are repeated for each triangle (and we know that for a Poisson distribution of points in the plane a given point has exactly 6 incident triangles).

edge-based structures: all the edge-based topological data structure discussed in GEO1002 (DCEL, half-edge, winged-edge, etc) can be used. These usually lead to large storage space.

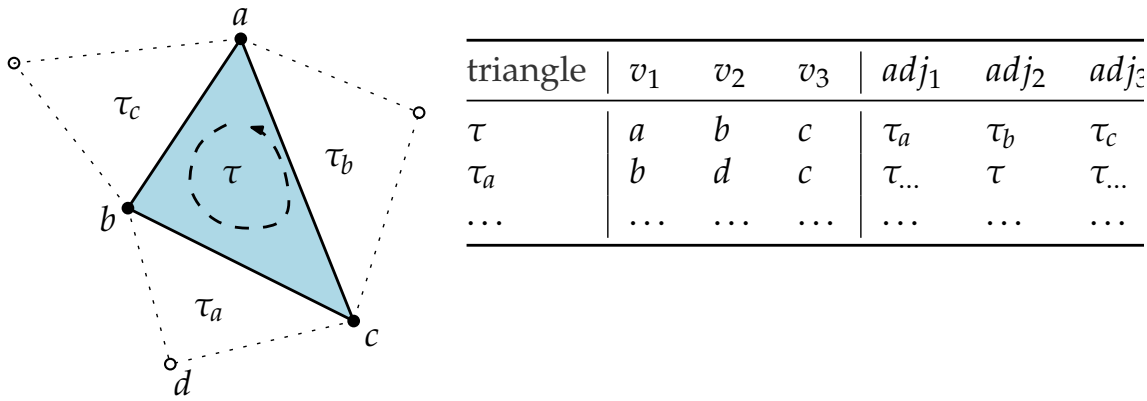


Figure 14: The triangle-based data structure to store efficiently a triangulation (and the adjacency relationships between the triangles).

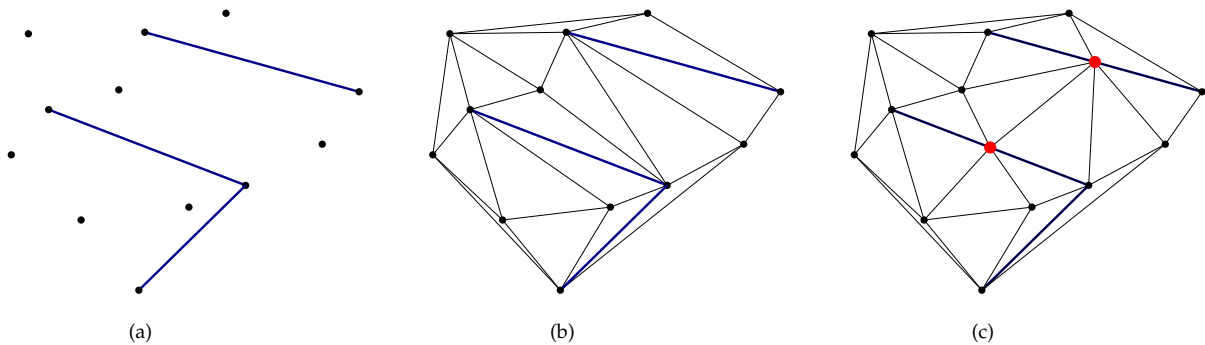


Figure 15: (a) A set S of points and straight-line segments. (b) Constrained DT of S . (c) Conforming DT of S ; the Steiner points added are in red.

Observe that in practice, if only the DT is wanted (and not the constrained one, see below), practitioners will often simply store the sample points and reconstruct on-the-fly the DT, since it is unique (if we omit points not in general position that is).

However, because it is simpler to manage triangles over arbitrary polygons (they always have exactly 3 vertices and 3 neighbours), data structures specific for triangulations have been developed and are usually used.

The simplest data structure, as shown in Figure 14, considers the triangle as being its atom and stores each triangle with 3 pointers to its vertices and 3 pointers to its adjacent triangles.

6 Constrained and Conforming Delaunay Triangulations

Given as input a set S of points and straight-line segments in the plane, different triangulations of S (so that the segments are respected) can be constructed. We are mostly interested in the *constrained Delaunay triangulation* (ConsDT) and the *conforming Delaunay triangulation* (ConfDT).

Constrained DT (ConsDT). Given a set S of points and straight-line segments in \mathbb{R}^2 , the ConsDT permits us to decompose the convex hull of S into non-overlapping triangles, and every segment of S appears as an edge in $\text{ConsDT}(S)$. ConsDT is similar to the Delaunay triangulation, but the triangles in ConsDT are not necessarily Delaunay (ie their circumcircle might contain other points from S). The empty circumcircle for a ConsDT is less strict: a triangle is Delaunay if its circumcircle contains no other points in S that are *visible* from the

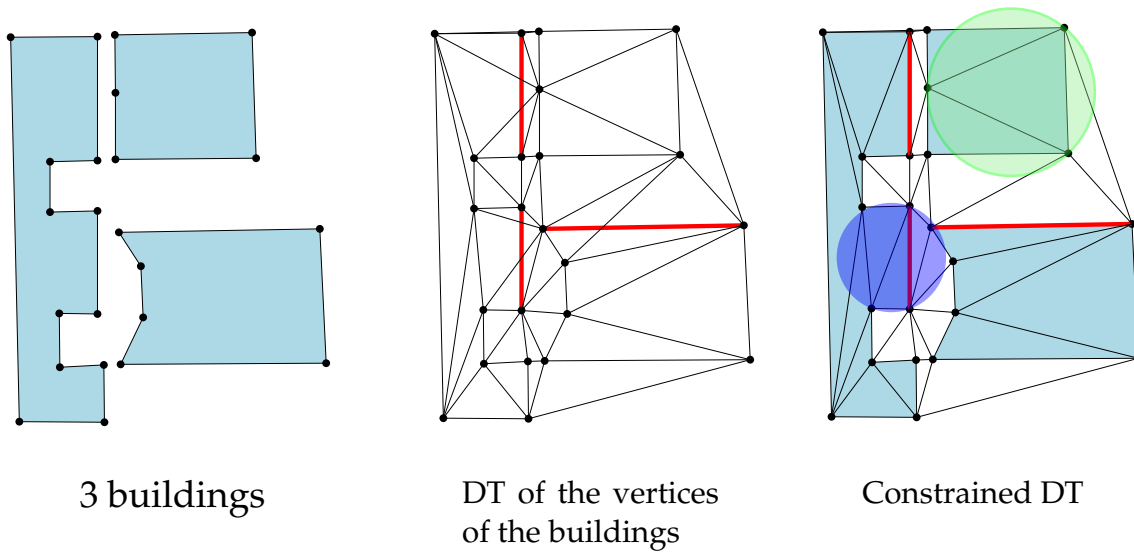


Figure 16: The ConsDT of a set of segments. On the right, the triangle whose circumcircle is green is a Delaunay (no other points in its interior) and so is the triangle whose circumcircle is in blue (there is one point in its interior, but it cannot be seen because of the constrained segment).

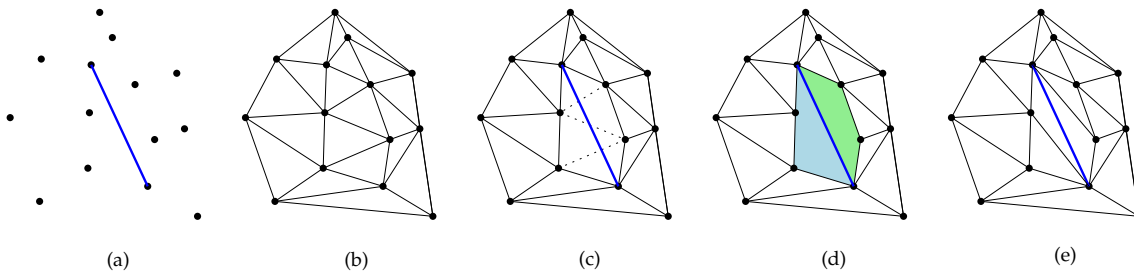


Figure 17: Steps to construct a ConsDT.

triangle. The constrained segments in S act a visibility blockers. Figure 16 shows one example.

Without going into details about one potential algorithm, one way to construct a $\text{ConsDT}(S)$ is (see Figure 17:

1. construct $\text{DT}(S^p)$, where S^p is the set containing all the points in S and the end points of the line segments (Figure 17b)
2. insert each line segment, each insertion will remove edges from $\text{DT}(S^p)$. In Figure 17c 3 edges are removed.
3. this creates 2 polygons that need to be retriangulated, in Figure 17d there is a blue and a green one.
4. retriangulate each separately, the Delaunay criterion needs to be verified only for the vertices incident to the triangles incident to the hole/polygon.

Observe that the ConsDT can be used to triangulate polygons with holes(see Figure 18, it suffices to remove the triangle outside the exterior boundary, but inside the convex hull.

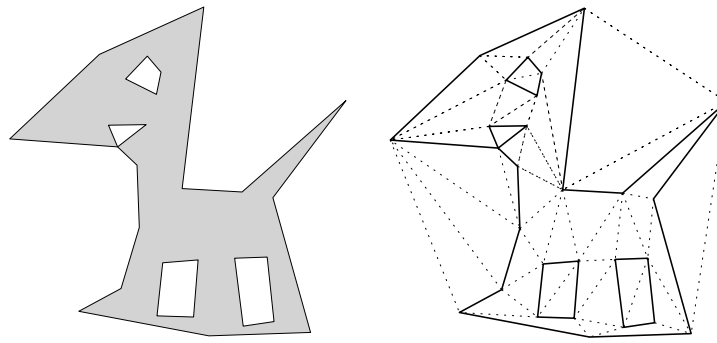


Figure 18: **Left:** One polygon with 4 holes (interior rings). **Right:** its ConsDT.

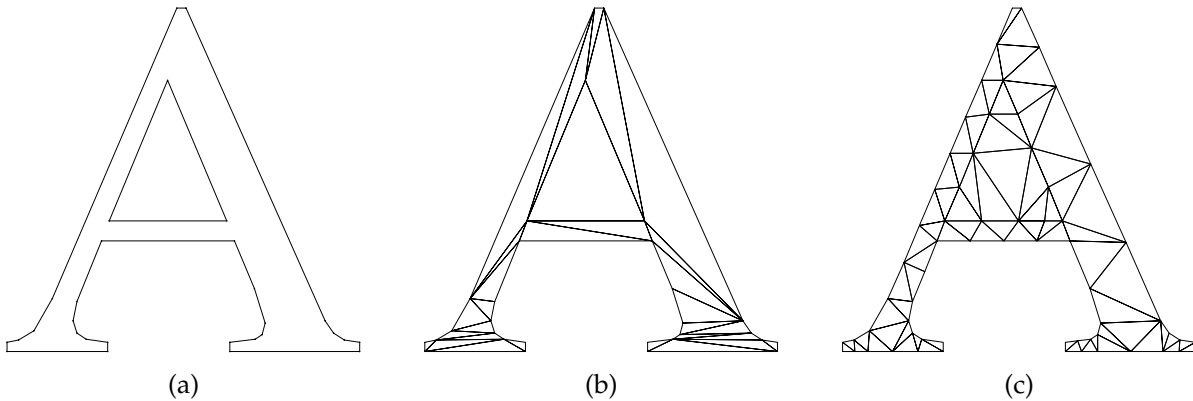


Figure 19: **(a)** Input polygon (or set of segments). **(b)** ConsDT of the input. **(c)** ConfDT of the input.

Conforming DT (ConfDT). A ConfDT adds new points to the input S (called *Steiner points*) to ensure that the input segments are present in the triangulation. As Figures 19 and 15c show, the input straight-line segments will be split into several collinear segments. The Steiner points have to be carefully chosen (where to put them is beyond the scope of this course). Observe that each triangle in a ConfDT respect the Delaunay criterion, but that more triangles are present. If 2 segments are nearly parallel, many points could be necessary (for m segments, up to m^2 could be necessary).

7 Triangulation of a polygon

We describe here algorithms to decompose a polygon into non-overlapping triangles (that are not necessarily Delaunay). While this is not directly related to the modelling of terrains (where we do not have polygons, usually), the topic is relevant in GIS and, as seen below, constrained are often used in terrains.

It is known that any simple polygon (even with holes) can be triangulated without adding new vertices.

7.1 The trivial case of a convex polygon

If the polygon to triangulate is convex, then triangulating it is trivial with a *fan-shaped triangulation*. As shown in Figure 20, the algorithm is simple: starting from an arbitrary vertex v of the polygon P , and add edges joining v to all other vertices of P , except the previous and next

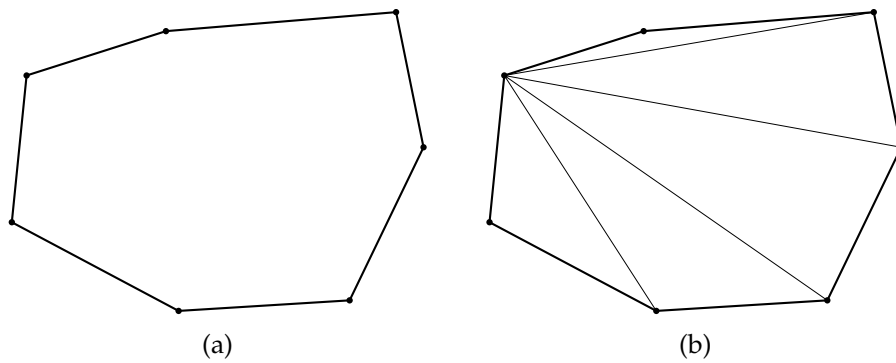


Figure 20: Fan-shaped triangulation of a convex polygon.

Algorithm 2: A greedy algorithm to triangulate a simple polygon.

Input: Simple polygon P with n vertices

Output: Triangulation \mathcal{T} of P

- 1 from the set D of $m = \frac{n(n-3)}{2}$ diagonals of P ;
 - 2 sort D into ascending order of length d_1, \dots, d_m ;
 - 3 triangulation $\mathcal{T} \leftarrow P$;
 - 4 **for** $i \leftarrow 1$ to m **do**
 - 5 **if** d_i does not intersect segments in \mathcal{T} AND d_i is an internal diagonal of P **then**
 - 6 $\mathcal{T} \leftarrow \mathcal{T} \cup d_i$;
-

vertices in the ordered boundary of P . A polygon P with n vertices will be triangulated into $(n - 2)$ triangles.

7.2 Greedy algorithm for polygons with holes

A greedy algorithm makes at each step the *locally* optimal choice, and does not look at what has been done before (it never goes back). It is very simple to understand and implement, but can yield a very slow implementation.

A greedy algorithm to triangulation a simple polygon P is described in Algorithm 2. The algorithm basically finds all the potential diagonals for P (all the pairs of vertices), sort them based on their length (ascending length), and at each step of the algorithm the shortest diagonal is inserted if it does not intersect with other already inserted diagonals, and if it is inside P . Diagonals that have been inserted are never removed. When all the possible diagonals have been inserted, each triangle must be constructed by finding 3 segments (boundary of P and diagonals). The algorithm is valid for polygon with holes, since all the possible combination of diagonals could be tried.

7.3 Ear clipping for polygons without holes

An alternative—and faster—method is by using the “ear clipping” algorithm. As shown in Figure 21, an ear is defined as 3 consecutive points of a polygon forming a triangle; abc is a valid ear of the input polygon P , but cde is not (because the triangle is outside the polygon) and neither is fgh (because d is inside the ear).. The idea of the ear clipping algorithm is to identify a valid ear, remove it from the polygon and repeat this process (with the modified polygon) until only one triangle is left. The implementation is simple: simply keep a list of

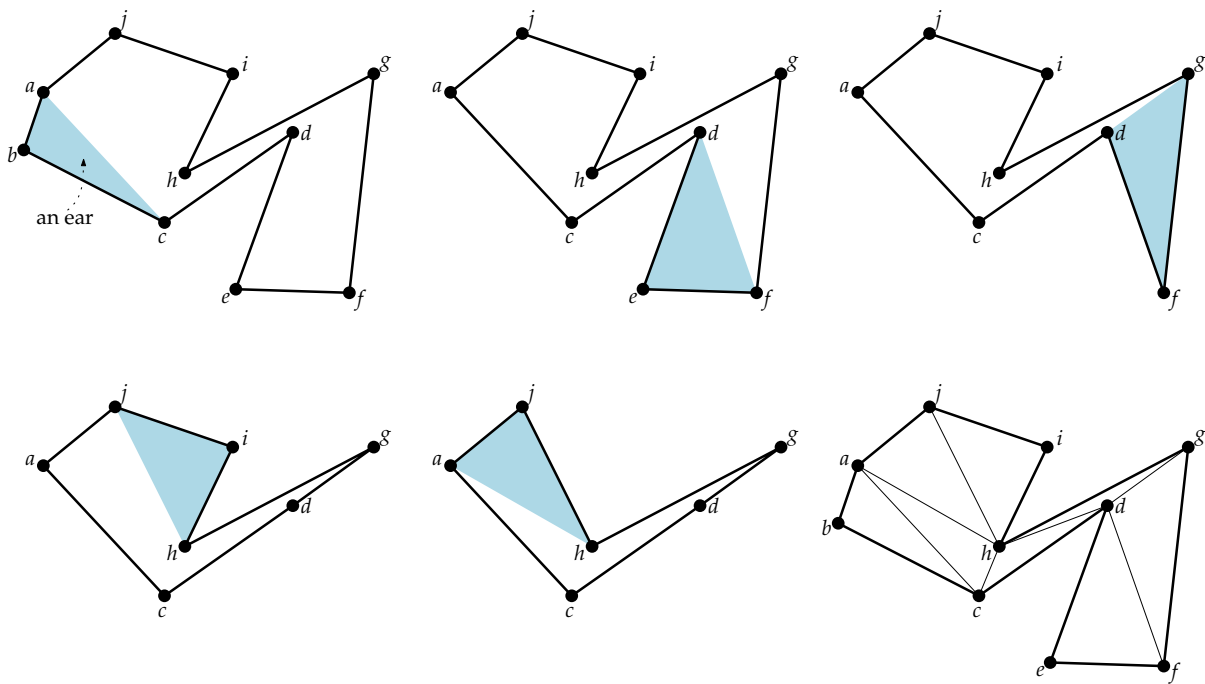


Figure 21: Ear clipping algorithm and the various steps to obtain the triangulation; that last steps are missing.

ordered vertices and each removal of an ear simply removes one vertex (in Figure 21 at the first step the vertex b is removed).

Observe that this algorithm does not work with polygon having holes.

Finding a valid ear of a simple polygon P is relatively easy: if the points are ordered CCW on the boundary of P , then 3 consecutive points must have a positive ORIENTATION; and to verify that the ear is completely inside P the other points of P must be outside that triangle (for instance in Figure 21 the point d would be inside the triangle fgh).

8 Notes and comments

For the construction of the DT, the incremental algorithm we use was first described by Lawson (1972). Guibas and Stolfi (1985) describe a divide-and-conquer algorithm, and Fortune (1987) a sweep-line one.

The local optimality of a DT, which implies globally optimality in the case of the DT, was proven by Delaunay (1934) himself. The *max-min angle optimality* of the DT was firstly observed by Sibson (1978). This parabolic lifting was first observed by Brown (1979) (who used a spherical transformation), further described by Seidel (1982); Edelsbrunner and Seidel (1986).

The fact that the DT is computed only in 2D (without taking into account the elevation of the vertices) has been criticised by several as sub-optimal for the modelling of terrain. To remedy to this, Dyn et al. (1990) develop a *data-dependent algorithm*, and Gudmundsson et al. (2002) propose using *higher-order* Delaunay triangulations (a modification where the empty circumcircle is also based on the neighbours of the neighbours). Nevertheless, there are still results stating that the Delaunay triangulation is still the one that minimises the roughness of a surface (Wang et al., 2001; Rippa, 1990)

The Algorithm 2, is taken from Worboys and Duckham (2004, p. 202).

Shewchuk (1997) shows that while the triangle-based data structure requires twice as much

code as with the quad-edge (to store and construct a ConsDT), the result is that the code runs twice as fast and the memory requirement as about 2X less. CGAL (<https://www.cgal.org/>), among many others, uses the triangle-based data structure.

References & further reading

- Brown KQ (1979). Voronoi diagrams from convex hulls. *Information Processing Letters*, 9(5):223–228.
- Delaunay BN (1934). Sur la sphère vide. *Izvestia Akademia Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7:793–800.
- Dyn N, Levin D, and Rippa S (1990). Data dependent triangulations for piecewise linear interpolation. *IMA Journal of Numerical Analysis*, 10(1):137–154.
- Edelsbrunner H and Seidel R (1986). Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1:25–44.
- Fortune S (1987). A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174.
- Gudmundsson J, Hammar M, and van Kreveld M (2002). Higher order Delaunay triangulations. *Computational Geometry—Theory and Applications*, 23:85–98.
- Guibas LJ and Stolfi J (1985). Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123.
- Lawson CL (1972). Transforming triangulations. *Discrete Applied Mathematics*, 3:365–372.
- Rippa S (1990). Minimal roughness property of the Delaunay triangulation. *Computer Aided Geometric Design*, 7:489–497.
- Seidel R (1982). *Voronoi diagrams in higher dimensions*. Ph.D. thesis, Diplomarbeit, Institut für Informationsverarbeitung, Technische Universität Graz, Austria.
- Shewchuk JR (1997). *Delaunay Refinement Mesh Generation*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburg, USA.
- Sibson R (1978). Locally equiangular triangulations. *The Computer Journal*, 21:243–245.
- Voronoi G (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les paralléloèdres primitifs. *Journal für die Reine und Angewandte Mathematik*, 134:198–287.
- Wang K, Lo Cp, Brook GA, and Arabnia HR (2001). Comparison of existing triangulation methods for regularly and irregularly spaced height fields. *International Journal of Geographical Information Science*, 15(8):743–762.
- Worboys MF and Duckham M (2004). *GIS: A computing perspective*. CRC Press, second edition edition.