VORONOI DIAGRAMS AND DELAUNAY TRIANGULATIONS

Franz Aurenhammer • Rolf Klein • Der-Tsai Lee

World Scientific

VORONOI DIAGRAMS AND DELAUNAY TRIANGULATIONS

Franz Aurenhammer
Graz University of Technology, Austria

Rolf Klein
University of Bonn, Germany

Der-Tsai Lee
Academia Sinica, Taiwan

World Scientific

NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI

Published by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

Library of Congress Cataloging-in-Publication Data

Aurenhammer, Franz, 1957–

Voronoi diagrams and Delaunay triangulations / Franz Aurenhammer, Graz University of Technology, Austria, Rolf Klein, University of Bonn, Germany, Der-Tsai Lee, Academia Sinica, Taiwan.

pages cm

Includes bibliographical references and index.

ISBN 978-9814447638 (hardcover : alk. paper)

1. Voronoi polygons. 2. Spatial analysis (Statistics) I. Klein, Rolf, 1953– II. Lee, Der-Tsai. III. Title.

QA278.2.A97 2013

516.22--dc23

2013018154

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

Typeset by Stallion Press

Email: enquiries@stallionpress.com

Printed in Singapore

CONTENTS

Chapter 1

INTRODUCTION

This book is devoted to an important and influential geometrical structure called the *Voronoi diagram*, whose origins in the Western literature date back to at least the 17th century. In his work on the principles of philosophy [257], René Descartes (Renatus Cartesius) claims that the solar system consists of vortices. His illustrations show a decomposition of space into convex regions, each consisting of matter revolving around one of the fixed stars; see Figure 1.1.

Even though Descartes has not explicitly defined the extension of these regions, the underlying idea seems to be the following. Let some *space*, and a set of *sites* in this space be given, together with a notion of the *influence* a site p exerts on every location x of the space. Then the *region* of the site p consists of all points x for which the influence of p is the strongest. Figure 1.2 illustrates the simplest case: Sites are points in the plane, and influence is modeled by Euclidean distance. The regions of the sites are convex polygons covering the entire plane.

'*The world is full of Voronoi diagrams*' ... this phrase seems true wherever we look, and it is the intention of this book to lead the reader into the fascinating realm of Voronoi diagrams. Whether we look to the sky, where gravitational fields structure the macrocosmos, or consider the spread of animals in their habitats, simply watch the interference pattern of waves on a quiet pond, or even peer deep within the structure of matter, we will find a Voronoi diagram-like pattern which structures the world.

Indeed, this fundamental concept has emerged independently, and proven useful, in various fields of science. Most applications have their own notions of 'space', 'sites', and 'influence', resulting in Voronoi diagrams whose structures differ greatly. Understanding their properties is the key to their effective application and to the development of fast construction algorithms. Various names have been given to Voronoi diagrams, depending on the particular domain, such as *Thiessen polygons* in geography and

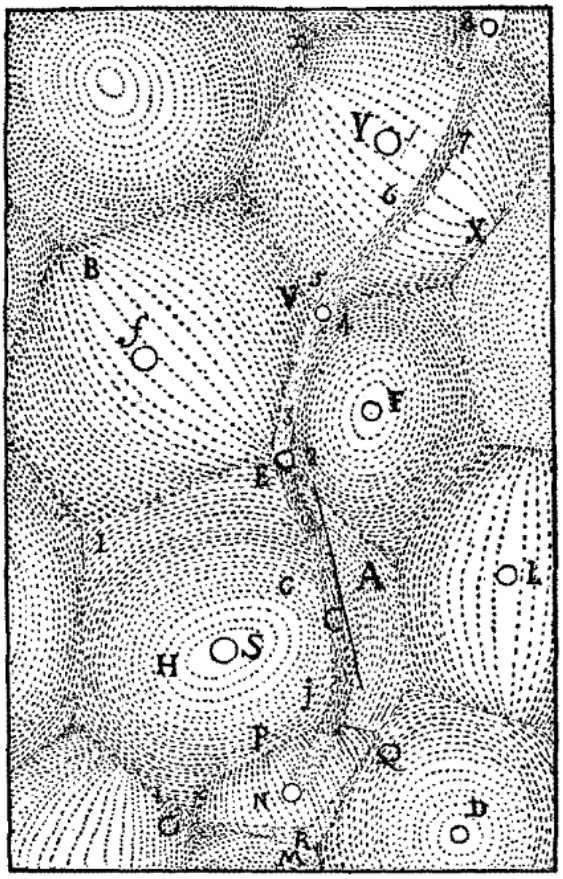


Figure 1.1. Descartes' decomposition of space into vortices.

meteorology (Thiessen [683], 1911), *domains of action* or *Wirkungsbereiche* in crystallography (Niggli [562], 1927), *Wigner–Seitz zones* in chemistry and physics (Wigner and Seitz [705], 1933), *Johnson–Mehl model* in mineralogy (Johnson and Mehl [433], 1939), and *medial axis transform* in biology and physiology (Blum [134], 1973).

Voronoi diagrams are important to both theory and applications, and play a unique interdisciplinary role. Several thousands of research articles have been published about them in different communities. Results thus dispersed might not always be widely known, and might fade into oblivion. While no single book can present all known results on Voronoi diagrams and their relatives, our aim is to thoroughly cover the structural and algorithmic viewpoints. In addition to being a versatile space partitioning structure, Voronoi diagrams are also aesthetically pleasing, and many people feel attracted to them, even regarding their artistic aspects. We have tried to communicate this quality in this book.

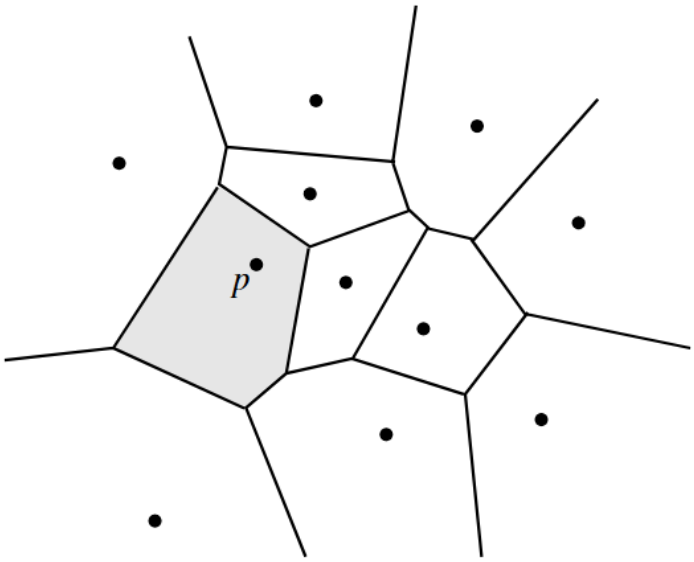


Figure 1.2. A Voronoi diagram of point sites in the Euclidean plane.

One and a half centuries ago, the mathematicians Carl Friedrich Gauß [353] (1840) and Gustav Lejeune Dirichlet [280] (1850), and later Georgi Feodosjewitsch Voronoi [693, 694] (1908) were the first to formally introduce this concept. They used it to study quadratic forms: Here the sites are integer lattice points, and influence is measured by Euclidean distance. The resulting structure was called a *Dirichlet tessellation* or *Voronoi diagram*, which became its standard name today.

Voronoi [693] also considered the *geometrical dual* of this structure, where any two (point) sites are connected whose regions have a boundary in common. Later, Boris Delaunay (Delone) [253] obtained the same structure by defining that two sites are connected if they lie on a circle whose interior contains no other sites. After him, the dual of the Voronoi diagram was denoted *Delaunay tessellation* or *Delaunay triangulation*.

With the advent of modern computers, the important role of Voronoi diagrams in structuring, representing, and displaying multidimensional data was rediscovered by computational geometers. Voronoi diagrams are a well-established geometric data structure nowadays. About one out of sixteen articles in computational geometry is dedicated to them, ever since Shamos and Hoey [637] introduced them to the field. In the two-dimensional case, for instance in the Euclidean plane, the Voronoi diagram does not require significantly more storage than does its underlying set of sites, and thus captures the inherent proximity information in a comprehensive and computationally useful manner. Its applications in more practically oriented areas of computer science are numerous, for example, in geographic

information systems, robotics, computer graphics, and data classification and clustering, to name a few.

Besides its direct applications in diverse fields of science, the Voronoi diagram and its dual can be used for solving numerous, and surprisingly different, geometric and graph-theoretical problems. Due to their close relationship to polytopes and arrangements of hyperplanes in higher dimensions, many questions (and solutions) from convex and discrete geometry carry over to Voronoi diagrams. Moreover, the Delaunay triangulation, seen as a combinatorial graph, is related to several prominent connectivity graphs. We discuss various respective applications, often including alternative solutions for their special merits. Along the way, we give a state-of-the-art account of the literature on Voronoi diagrams in computational geometry. This fills the need for a technically sound and well-structured book, which is up-to-date on the theory and mathematical applications of Voronoi diagrams.

The reader is invited on a guided tour of gently increasing difficulty through a fascinating area. Insight will be given into the ideas and principles of Voronoi diagrams, without the baggage of too much technical detail. When later faced with a geometric partitioning problem, readers should find our book helpful in deciding whether their problem shows Voronoi characteristics and which type of Voronoi diagram applies. They might find a ready-to-use solution in our book, or follow up the links to the literature provided, or they might work out their own solution based on the algorithms they have seen. The book targets researchers in mathematics, computer science, natural and economical sciences, instructors and graduate students in those fields, as well as the ambitious engineers looking for alternative solutions. A brief discussion of algorithmic implementation questions, and of currently available geometric computation libraries, is included in the final chapter. Since human intuition is aided by visual perception, especially where geometric topics are concerned, the diversity and appeal of Voronoi diagrams is liberally illustrated with appropriate figures.

The presentation and structure of this book strives to highlight the intrinsic potential of Voronoi diagrams and Delaunay triangulations, which lies in their structural properties, in the existence of efficient algorithms for their construction, and in their relationship to seemingly unrelated concepts. We therefore organized topics by concept, rather than by application.

Another book which nicely complements ours, but from a much more applied perspective, is by Okabe *et al.* [571] (2000). It contains a wealth of applications of Voronoi diagrams, several of them not (or only marginally) covered here, like *Poisson Voronoi diagrams* and *locational*

optimization problems. The more than one thousand and six hundred references listed in [571] constitute a bibliography still quite distinct from ours — demonstrating once more the broad scope of Voronoi diagrams. In addition, the book by Gavrilova [354] (2008) contains a collection of articles on diverse applications in the natural sciences, with numerous citations. A careful study of Delaunay triangulations, mainly oriented towards their algorithmic applications in *surface meshing* and *finite element methods*, is contained in George and Borouchaki [356] (1998). The recent book on Delaunay mesh generation by Cheng, Dey, and Shewchuk [202] (2013) takes into account the various new developments to date.

Shorter and early treatments of Voronoi diagrams and Delaunay triangulations, closer to the spirit of the present book, are the surveys by Aurenhammer [93], Fortune [343], and Aurenhammer and Klein [100]. Also, Chapters 5 and 6 of Preparata and Shamos [596], Chapter 13 of Edelsbrunner [300], Chapters 7 and 9 of de Berg *et al.* [244], and Part V of Boissonnat and Yvinec [151] could be consulted.

Although interesting and insightful in its own right, we decided to refrain from a detailed historical treatment of Voronoi diagrams in this book. The interested reader is encouraged to enjoy the historical perspectives presented in [93], and later in more detail, in [571].

We start in Chapter 2 with a simple case — the Voronoi diagram and the Delaunay triangulation of n points in the plane, under Euclidean distance. We state elementary structural properties that follow directly from the definitions. Further properties will be revealed in Chapter 3, where different algorithmic schemes for computing these structures are presented. In Chapter 4 we complete our presentation of the classical two-dimensional case, with advanced properties of planar Voronoi diagrams and Delaunay triangulations. We next turn to generalizations, to sites more general than points in Chapter 5, and to higher dimensions in Chapter 6. Generalized spaces and distances are treated elaborately in Chapter 7. In Chapter 8, important geometric applications of the Voronoi diagram and the Delaunay triangulation are discussed, along with respective related structures and concepts. The reader interested mainly in these applications can proceed directly to Chapter 8, after Chapter 2 or 3. Chapter 9 presents relevant topics which, for clarity of exposition, are best described separately. Chapter 10 offers alternative solutions in high dimensions, where the attractiveness of Voronoi diagrams is partially lost due to their high combinatorial and computational worst-case complexity. Finally, Chapter 11 concludes the book, gives a short discussion of algorithmic implementation issues, and mentions some important open problems.

Chapter 2

ELEMENTARY PROPERTIES

In this chapter, we present definitions and basic properties of the Voronoi diagram and its dual, the Delaunay triangulation.

Only the simplest case is considered — point sites in the plane under the Euclidean distance. Yet, the properties discussed are of general importance, as many (but not all) of them will carry over to other types of Voronoi diagrams and their relatives, presented later in this book.

2.1. Voronoi diagram

Let us start with giving some standard notation and explanations. Throughout this chapter, we will denote by S a set of $n \geq 3$ point sites $p, q, r, \ldots$ in the Euclidean plane, $\mathbf{R}^2$. For points $p = (p_1, p_2)$ and $x = (x_1, x_2)$, their *Euclidean distance* is given as

$$d(p, x) = \sqrt{(p_1 - x_1)^2 + (p_2 - x_2)^2}.$$

The straight-line segment that connects two points p and q will be written as $\overline{pq}$, or sometimes just as pq.

For $p, q \in S$, let $B(p, q)$ be the *bisector* of p and q (also called their *separator*), which is the locus of all points in $\mathbf{R}^2$ at equal distance from both p and q. $B(p, q)$ is the perpendicular line through the midpoint of the line segment $\overline{pq}$. It separates the halfplane

$$D(p, q) = \{x \mid d(p, x) \leq d(q, x)\}$$

closer to p from the halfplane $D(q, p)$ closer to q.

We next specify maybe the most important notion in this book. The *Voronoi region* of p among the given set S of sites, for short $\mathrm{VR}(p, S)$, is the intersection of the $n - 1$ halfplanes $D(p, q)$, where q ranges over all the

other sites in S. More formally,

$$\mathrm{VR}(p,S) = \bigcap_{q\in S,\, q\neq p} D(p,q).$$

$\mathrm{VR}(p,S)$ consists of all points $x \in \mathbf{R}^2$ for which p is a nearest neighbor site. As being the finite intersection of halfplanes, which are convex sets in $\mathbf{R}^2$, the region $\mathrm{VR}(p,S)$ is a convex, possibly unbounded *polygon* in the plane. Different Voronoi regions are interior-disjoint, as they are separated by the bisector of the two sites that own them.

(A set M in $\mathbf{R}^2$, or in general d-space $\mathbf{R}^d$, is *convex* if it contains, with each pair of points $x, y \in M$, the line segment $\overline{xy}$. Set M is called *bounded* if there exists a circle, or sphere, respectively, of finite radius which encloses M. Otherwise, M is *unbounded*. We say that M is a *closed* set if M contains its (topological) boundary. M minus its boundary is an *open* set, called the *interior* of M.)

Definition 2.1. The common boundary part of two Voronoi regions is called a *Voronoi edge*, if it contains more than one point.

The *Voronoi diagram* of S, for short $V(S)$, is defined as the union of all Voronoi edges.

Endpoints of Voronoi edges are called *Voronoi vertices*; they belong to the common boundary of three or more Voronoi regions.

If a Voronoi edge e borders the regions of p and q then $e \subset B(p,q)$ holds. That is, $V(S)$ is a *planar straight-line graph* whose edges emanate from Voronoi vertices. $V(S)$ is sometimes also referred to as the *Voronoi edge graph* in the literature.

(Recall that a *graph* consists of a set of *vertices*, which in principle could be any objects, and a set of *edges* that pair up certain vertices, in order to display a relation between the two objects. A graph is called *planar* if it can be geometrically embedded in the plane without edge crossings. For sources on (combinatorial) graphs and graph-related algorithms, the books by Diestel [275], Gibbons [363], or West [703] may be consulted.)

There is an intuitive way of looking at the Voronoi diagram $V(S)$. Let x be an arbitrary point in the plane. We center a circle, C, at x and let its radius grow, from 0 on. At some stage the expanding circle will, for the first time, hit one or more sites of S. Now there are three different cases.

Lemma 2.1. *If the circle C expanding from point x hits exactly one site, p, then x belongs to the interior of region $\mathrm{VR}(p,S)$. If C hits exactly two sites, p and q, then x is an interior point of a Voronoi edge separating the*

regions of p and q. If C hits three or more sites simultaneously, then x is a Voronoi vertex adjacent to those regions whose sites have been hit.

Proof. If only site p is hit then p is the unique element of S closest to x. Consequently, $x \in D(p, r)$ holds for each site $r \in S$ with $r \neq p$. If C hits exactly p and q, then x is contained in either halfplane $D(p, r), D(q, r)$, where $r \notin \{p, q\}$, and also in $B(p, q)$, the common boundary of $D(p, q)$ and $D(q, p)$. By Definition 2.1, x belongs to the closure of the regions of both p and q, but of no other site in S. In the third case, the argument is analogous. □

This lemma shows that the Voronoi diagram forms a *decomposition* (or *partition*) of the plane; see Figure 2.1: Regions do not overlap, and for each point x there is at least one closest site in S, meaning that x is covered by this region.

Conversely, if we imagine n circles expanding from the sites at the same speed, the fate of each point x of the plane is determined by those sites whose circles reach x first. This '*expanding waves*' view, or *wavefront model*, has been systematically used by Chew and Drysdale [215] and Thurston [685].

The Voronoi vertices are of degree at least three, by Lemma 2.1. (The *degree* of a vertex is the number of its incident edges.) Vertices of degree higher than three do not occur if no four point sites are cocircular. The

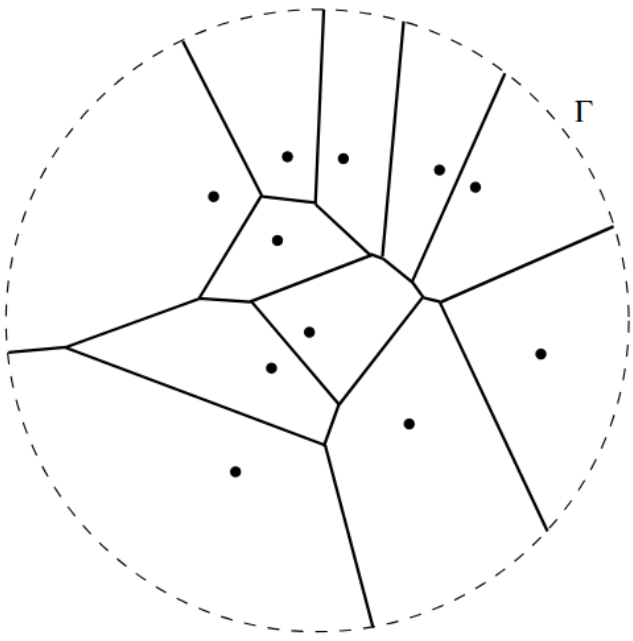


Figure 2.1. Voronoi diagram of 11 sites and bounding curve Γ.

Voronoi diagram $V(S)$ is disconnected if all point sites are collinear; in this case it consists of parallel lines.

From the Voronoi diagram of S one can easily derive the *convex hull* of S, which is defined as the smallest convex set containing S. As S is finite, its convex hull is a *convex polygon*, spanned by $h \leq n$ *extreme* points of S.

Lemma 2.2. *A point p of S lies on the boundary of the convex hull of S iff (i.e., if and only if) its Voronoi region $\mathrm{VR}(p, S)$ is unbounded.*

Proof. The Voronoi region of p is unbounded iff there exists some point $q \in S$ such that $V(S)$ contains an unbounded piece of $B(p, q)$ as a Voronoi edge. Let $x \in B(p, q)$, and let $C(x)$ denote the circle through p and q centered at x, as shown in Figure 2.2. Point x belongs to $V(S)$ iff $C(x)$ contains no other site. As we move x to the right along $B(p, q)$, the part of $C(x)$ contained in halfplane R keeps growing. If there is another site r in R, it will eventually be reached by $C(x)$, causing the Voronoi edge to end at x. Otherwise, all other sites of S must be contained in the closure of the left halfplane L. Then p and q both lie on the convex hull of S. □

If the point set S is in *convex position* (i.e., all its points are vertices of the convex hull of S) then all regions of $V(S)$ are unbounded, by Lemma 2.2, and their edges form a *cycle-free* and connected graph, that is to say, a *tree*.

Sometimes it is convenient to imagine a simple closed curve Γ around the 'interesting' part of the Voronoi diagram, so large that it intersects only the unbounded Voronoi edges; see Figure 2.1. (A curve is called *simple* if it contains no self-intersections.) While walking along Γ, the vertices of the

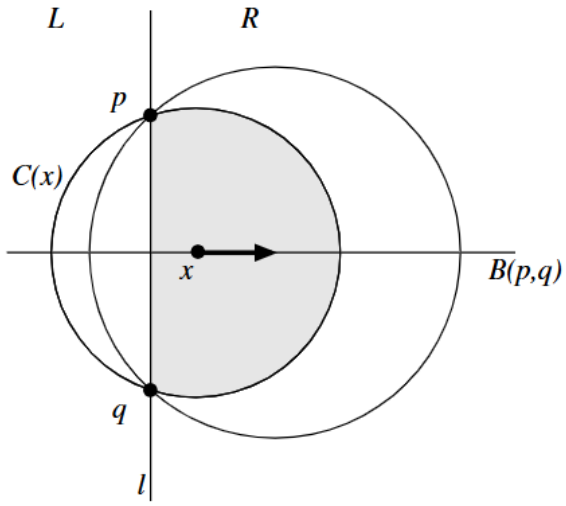


Figure 2.2. As x moves to the right, the intersection of circle $C(x)$ with the left halfplane L shrinks, while $C(x) \cap R$ grows.

convex hull of S can be reported in cyclic order. After removing the halflines outside Γ, a connected embedded planar graph with $n+1$ faces results. Its faces are the n Voronoi regions and the unbounded face outside Γ. We call this graph the *finite* Voronoi diagram.

One virtue of the Voronoi diagram is its small size.

Lemma 2.3. *The Voronoi diagram $V(S)$ has $O(n)$ many edges and vertices. The average number of edges in the boundary of a Voronoi region is less than 6.*

Proof. By *Euler's polyhedron formula* (see e.g. [363]) for planar graphs, the following relation holds for the numbers v, e, f, and c of vertices, edges, faces, and connected components, respectively

$$v - e + f = 1 + c.$$

We apply this formula to the finite Voronoi diagram. Each vertex has at least three incident edges; by adding up we obtain $e \geq 3v/2$, because each edge is counted twice. Substituting this inequality together with $c = 1$ and $f = n + 1$ yields

$$v \leq 2n - 2 \quad \text{and} \quad e \leq 3n - 3.$$

Adding up the numbers of edges contained in the boundaries of all $n+1$ faces results in $2e \leq 6n - 6$, because each edge is again counted twice. Thus, the average number of edges that border a region is at most $(6n-6)/(n+1) < 6$. The same bounds apply to $V(S)$. □

2.2. Delaunay triangulation

Now we turn to the Delaunay tessellation. In general, a *triangulation* of S is a planar graph with vertex set S and straight-line edges, which is maximal in the sense that no further straight-line edge can be added without crossing others. Triangulations are also called *triangular networks* in the literature.

Each triangulation of S contains the edges of the convex hull of S. Its bounded faces are triangles, due to maximality. Their number equals exactly $2n - h - 2$, where h counts the vertices of the convex hull. The number of edges is $3n - h - 3$. Note that both numbers, which easily follow from Euler's polyhedron formula, are independent from the way of triangulating the point set S, by the characteristics of this formula.

We call a connected subset of edges of a triangulation a *tessellation* of S if it contains the edges of the convex hull, and if each point of S has at least two incident edges.

Definition 2.2. The *Delaunay tessellation* $\mathrm{DT}(S)$ is obtained by connecting with a line segment any two points p, q of S, for which a circle exists that passes through p and q but does not contain any other site of S in its interior or boundary. The edges of $\mathrm{DT}(S)$ are called *Delaunay edges*.

Equivalently, and also common in the literature, $\mathrm{DT}(S)$ can be defined to contain all polygonal faces spanned by S whose *circumcircles* are empty of points of S. This is the so-called *empty-circle property* of the Delaunay tessellation. Another equivalent characterization of $\mathrm{DT}(S)$ is a direct consequence of Lemma 2.1.

Lemma 2.4. *Two points of S are joined by a Delaunay edge iff their Voronoi regions are edge-adjacent.*

Since each Voronoi region has at least two neighbors, at least two Delaunay edges must emanate from each point of S. By the proof of Lemma 2.2, each edge of the convex hull of S is Delaunay. Finally, two Delaunay edges can only intersect at their endpoints, because they allow for circumcircles whose respective closures do not contain other sites. This shows that $\mathrm{DT}(S)$ is in fact a *tessellation* of S.

Two Voronoi regions can share at most one Voronoi edge, by convexity. Therefore, Lemma 2.4 implies that $\mathrm{DT}(S)$ is the graph-theoretical *dual* of $V(S)$, realized by straight-line edges. Delaunay edges correspond to Voronoi edges, Delaunay faces to Voronoi vertices (the centers of their circumcircles), and Delaunay vertices (the n sites) to Voronoi regions — in a bijective way.

An example is depicted in Figure 2.3; the Voronoi diagram $V(S)$ is drawn by dashed lines, and $\mathrm{DT}(S)$ by solid lines. Note that a Voronoi vertex (like w) need not be contained in its associated face of $\mathrm{DT}(S)$. The sites p, q, r, s are cocircular, giving rise to a Voronoi vertex v of degree 4. Consequently, its corresponding Delaunay face is bordered by four edges. This cannot happen if the points of S are in *general position*, i.e., no three of them are collinear, and no four of them are cocircular.

Theorem 2.1. *If the point set S is in general position then $\mathrm{DT}(S)$, the dual of the Voronoi diagram $V(S)$, is a triangulation of S, called the* Delaunay triangulation. *Three points of S give rise to a Delaunay triangle exactly if the circle they define does not enclose any other point of S.*

Observe that $\mathrm{DT}(S)$ can be a triangulation even when S is not in general position. Its number of triangles varies, as mentioned before, with the number, h, of point sites which are *extreme* in S, i.e., which lie on the boundary of the convex hull of S. A maximum of $2n - 5$ is attained if the

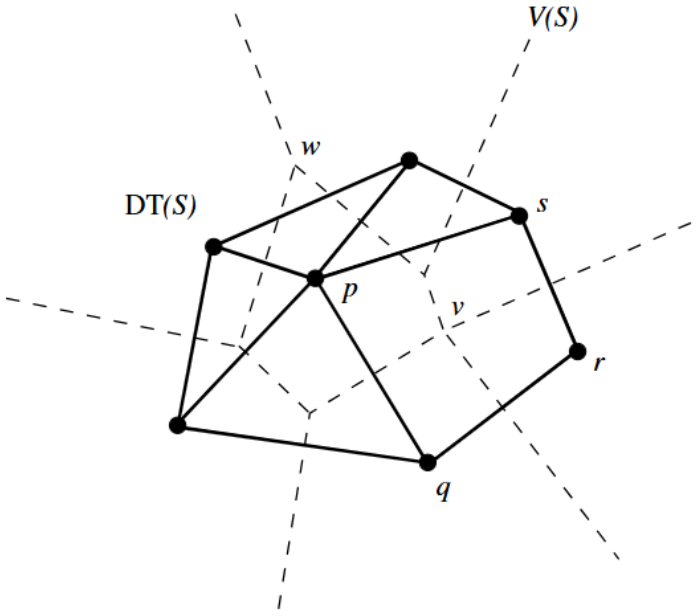


Figure 2.3. Voronoi diagram and Delaunay tessellation.

convex hull is triangular ($h = 3$). On the other end of the spectrum, if S is in *convex position* ($h = n$), then the convex hull is an n-gon, C, and $\mathrm{DT}(S)$ partitions C into only $n - 2$ triangles, using diagonals of C.

As in every triangulation of a planar point set S, vertices of high degree may occur in $\mathrm{DT}(S)$ for special positions of S. A site $p \in S$ can have $n - 1$ incident Delaunay edges (and triangles), because its Voronoi region $\mathrm{VR}(p, S)$ can be adjacent to all other regions in $V(S)$. Still, 'almost all' sites will be of small constant degree in $\mathrm{DT}(S)$, because their *average* degree is bounded by 6; see Lemma 2.3.

The Delaunay triangulation possesses various interesting features, concerning its optimization properties and subgraph structures, but also its flexibility in adapting to more general settings and higher dimensions. We will discuss this later at appropriate places, mainly in Sections 4.2 and 8.2 and, regarding algorithmic issues, in Section 3.2 and Subsection 6.1.2.

We have seen, in this short chapter, two structures for a finite planar point set S which are most fundamental and elementary, apart maybe from the convex hull of S. The Voronoi diagram $V(S)$ and the Delaunay triangulation $\mathrm{DT}(S)$ display the proximity influence that S exerts in a simple and intuitive way, with a description of only linear size. Their hidden intrinsic potential will be gradually revealed in the chapters to come. We start in Chapter 3, with showing that they lend themselves to several efficient methods of construction.

Chapter 3

BASIC ALGORITHMS

In this chapter we present several ways of computing a geometric representation of the Voronoi diagram and its dual, the Delaunay tessellation. For simplicity, we assume of the n point sites of S that no four of them are cocircular, and that no three of them are collinear. According to Theorem 2.1 we can then refer to $\mathrm{DT}(S)$ as the Delaunay triangulation. All algorithms presented herein can be made to run without this *general position* assumption. Also, they can be generalized to metrics other than the Euclidean, and to sites other than points. This will be discussed at appropriate places in later chapters.

Implementation issues, including degenerate configurations that might occur in the input or during the execution of algorithms, along with general numerical questions that arise in the computation of geometric objects, will be discussed in some detail in Section 11.2. They are only marginally addressed in the present chapter, not least for the sake of clarity in the presentation of the algorithmic techniques and their analyses. For a rich source on basic *data structures* (several of which will be used later on), we refer to the books by Cormen *et al.* [234] and Mehlhorn and Sanders [538].

We mainly seek for a representation of Voronoi diagrams in exact vector geometry, rather than as a binary image. Voronoi diagrams on *pixel maps* can be obtained by exact geometric algorithms followed by pixel extraction, or can be computed and visualized by graphics methods, beyond the scope of this book; see Section 11.1.

Data structures well suited for working with *planar graphs* like the Voronoi diagram are the *doubly connected edge list* (DCEL), by Muller and Preparata [552], and the *quad-edge* structure by Guibas and Stolfi [394]. In either structure, a record is associated with each edge e that stores the following information: the names of the two endpoints of e; references to the edges clockwise or counterclockwise next to e about its endpoints;

finally, the names of the faces to the left and to the right of e. The space requirement of both structures is $O(n)$.

Either structure allows to efficiently traverse the edges incident to a given vertex, or bordering a given face. The quad-edge structure offers the additional advantage of describing, at the same time, a planar graph and its *dual*, so that it can be used for constructing both the Voronoi diagram and the Delaunay triangulation. From the DCEL of $V(S)$ we can derive the set of triangles constituting the Delaunay triangulation in linear time. Conversely, from the set of all Delaunay triangles the DCEL of the Voronoi diagram can be constructed in time $O(n)$. Therefore, each algorithm for computing one of the two structures can be used for computing the other one, within $O(n)$ extra time effort.

3.1. A lower time bound

Before constructing the Voronoi diagram we want to establish a lower bound for its computational complexity.

Suppose that n real numbers $x_1, \ldots, x_n$ are given, which are to be sorted. We construct the point set $S = \{p_i = (x_i, {x_i}^2) \mid 1 \leq i \leq n\}$, which lies on the unit parabola and thus is in *convex position*; see Figure 3.1(i). By Lemma 2.2, the Voronoi diagram $V(S)$ forms a tree, from which one can derive, in $O(n)$ time, the vertices of the *convex hull* of S, in counterclockwise order. From the leftmost point in S on, this vertex sequence contains all points p_i, sorted by increasing values of x_i.

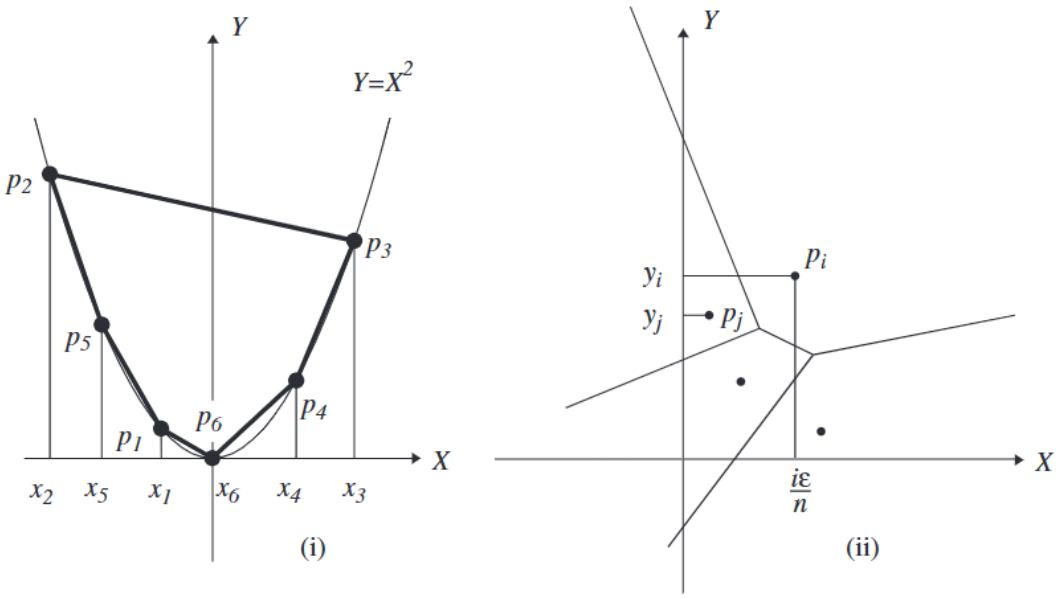


Figure 3.1. Proving the $\Omega(n \log n)$ lower bound for constructing the Voronoi diagram: by transformation (i) from sorting, and (ii) from ε-closeness.

This argument due to Shamos [636] shows that constructing the convex hull and, *a fortiori*, computing the Voronoi diagram, is at least as difficult as sorting n real numbers, which requires $\Omega(n \log n)$ time in the algebraic computation tree model [596].

However, a fine point is lost in this reduction. After sorting n points by their x-values, their convex hull can be computed in linear time [286], whereas sorting does not help in constructing the Voronoi diagram. The following result has been independently found by Djidjev and Lingas [281] and by Zhu and Mirzaian [716].

Theorem 3.1. *It takes time $\Omega(n \log n)$ to construct the Voronoi diagram of n points $p_1, \ldots, p_n$ whose x-coordinates are strictly increasing.*

Proof. The proof is by reduction from the *ε-closeness problem* [596], which is known to be in $\Omega(n \log n)$. Let $y_1, \ldots, y_n$ be positive real numbers, and let $\varepsilon > 0$. The question is if there exists $i \neq j$ such that $|y_i - y_j| < \varepsilon$ holds. We form the sequence of points

$$p_i = \left(\frac{i\varepsilon}{n}, y_i \right), \quad \text{for } 1 \leq i \leq n,$$

and compute their Voronoi diagram; see Figure 3.1(ii). In time $O(n)$, we can determine the Voronoi regions that are intersected by the y-axis, in bottom-up order (such techniques will be detailed in Section 3.3).

If, for each p_i, its projection onto the y-axis lies in the Voronoi region of p_i then the values y_i are available in sorted order, and we can easily answer the question. Otherwise, there is a point p_i whose projection lies in the region of some other point p_j. Because of

$$|y_i - y_j| \leq d((0, y_i), p_j) < d((0, y_i), p_i) = \frac{i\,\varepsilon}{n} \leq \varepsilon,$$

in this case the answer is positive. □

On the other hand, sorting n *arbitrary* point sites by x-coordinates is not made easier by their Voronoi diagram, as Seidel [626] has shown.

With Definition 2.1 in mind one could think of computing each Voronoi region as the intersection of $n - 1$ halfplanes. This would take time $\Theta(n \log n)$ per region, see [596]. In the following sections we describe various algorithms that compute the *entire* Voronoi diagram within this time; due to Theorem 3.1, these algorithms are (asymptotically) worst-case optimal.

3.2. Incremental construction

A natural idea first studied by Green and Sibson [378] is to construct the Voronoi diagram by *incremental insertion*, i.e., to obtain $V(S)$ from $V(S\backslash\{p\})$ by inserting the site p. In the beginning, when there are only two sites, their Voronoi diagram is just their bisector line.

Basically, the cyclic sequence of edges of the new Voronoi region $\mathrm{VR}(p, S)$ has to be constructed, and the invalidated parts of the diagram $V(S\backslash\{p\})$ inside $\mathrm{VR}(p, S)$ deleted, in a way quite similar to constructing *merge chains* of edges, as is detailed in the next section. Figure 3.2 illustrates the insertion process. As the region of p can have up to $n-1$ edges, for $n = |S|$, this leads to a runtime of $O(n^2)$.

(Symbol $|\cdot|$ denotes the *cardinality*, when applied to finite sets, that is, the number of elements in a set.)

Several authors fine-tuned the technique of inserting Voronoi regions, and efficient and numerically robust implementations are available nowadays; see Ohya *et al.* [570] and Sugihara and Iri [668]. In fact, overall runtimes of $O(n)$ can be expected for 'well-distributed' sets of sites.

The insertion process is, maybe, better described and implemented in the dual environment, for the Delaunay triangulation: Construct the triangulation $\mathrm{DT}_i = \mathrm{DT}(\{p_1, \ldots, p_{i-1}, p_i\})$ by inserting the site p_i into

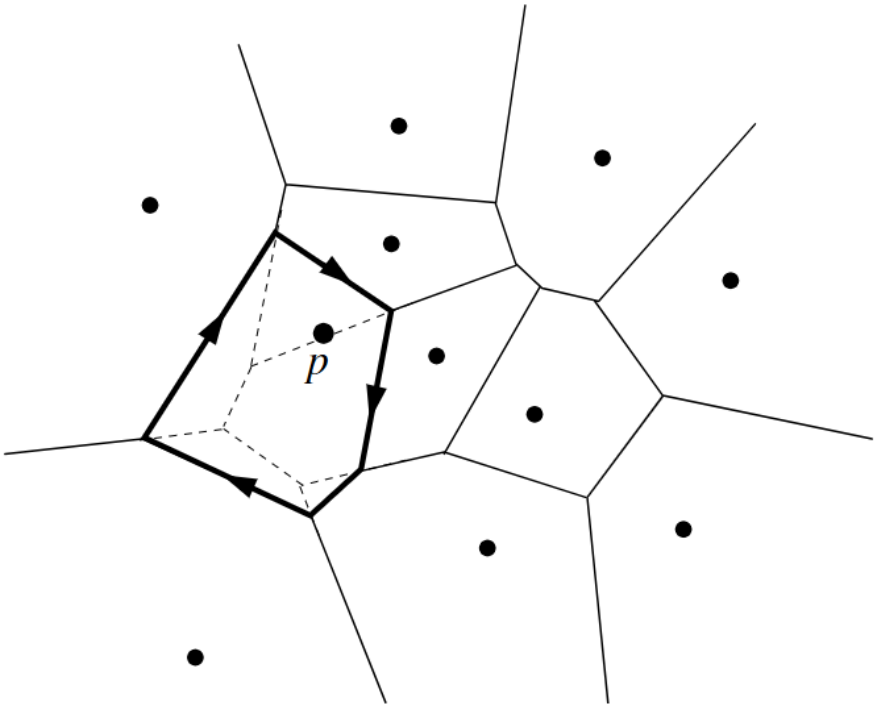


Figure 3.2. Inserting a Voronoi region. Invalidated portions of the diagram are drawn in dashed style.

DT_{i-1}. Clearly, DT_3 is a single triangle, and $\mathrm{DT}_n = \mathrm{DT}(S)$ is the final result. The advantage over a direct construction of $V(S)$ is that Voronoi vertices that appear in intermediate diagrams but not in the final one need not be constructed and stored. We follow Guibas and Stolfi [394] and construct DT_i by exchanging edges, using Lawson's [486] original *edge flipping* procedure, until all edges invalidated by p_i have been removed.

To this end, it is useful to extend the notion of triangle to the *unbounded face* of the Delaunay triangulation, which is the complement of the convex hull of S in the plane, $\mathbf{R}^2 \backslash \mathrm{conv}(S)$. If $\overline{pq}$ is an edge of $\mathrm{conv}(S)$, we call the supporting halfplane H not containing S an *infinite triangle* with edge $\overline{pq}$. Its 'circumcircle' is H itself, the limit of all circles through p and q whose centers tend to infinity within H; cf. Figure 2.2. As a consequence, each edge of a Delaunay triangulation is now incident to two triangles.

Those triangles of DT_{i-1} (finite or infinite) whose *circumcircles* contain the new site, p_i, are said to be *in conflict* with p_i. According to Theorem 2.1, they will no longer be Delaunay triangles.

Let $\overline{qr}$ be an edge of DT_{i-1}, and let $T(q, r, t)$ be the triangle incident to $\overline{qr}$ that lies on the other side of $\overline{qr}$ than p_i; see Figure 3.3. If its circumcircle $C(q, r, t)$ contains p_i then each circle through q, r contains at least one of p_i and t; see Figure 3.3 again. Consequently, $\overline{qr}$ cannot belong to DT_i, due to Definition 2.2. Instead, $\overline{p_i t}$ will be a new Delaunay edge, because there exists a circle contained in $C(q, r, t)$ that contains only p_i and t in its interior or boundary. This process of replacing edge $\overline{qr}$ by $\overline{p_i t}$ is called an *edge flip*.

The necessary edge flips can be carried out efficiently if we know the triangle $T(q, s, r)$ of DT_{i-1} that contains p_i, see Figure 3.4. (That is, we have

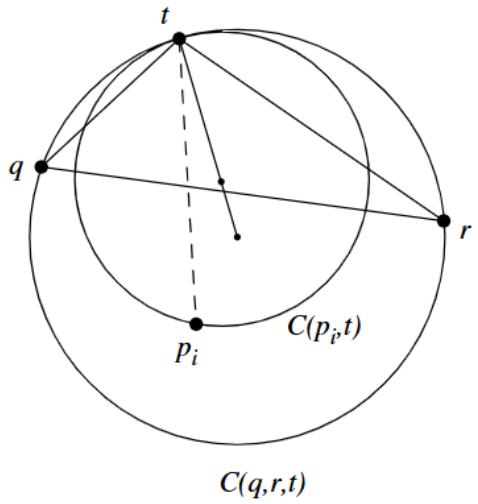


Figure 3.3. If triangle $T(q, r, t)$ is in conflict with p_i then the former Delaunay edge $\overline{qr}$ must be replaced by $\overline{p_i t}$.

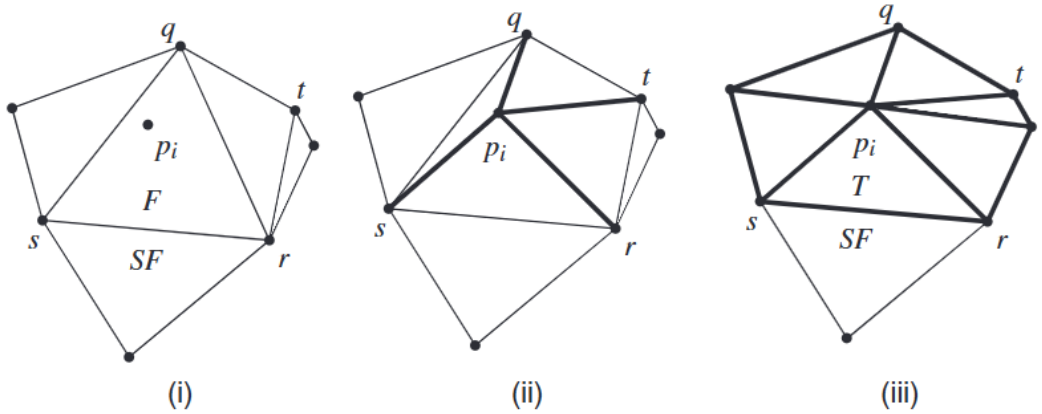


Figure 3.4. Updating DT_{i-1} after inserting the new site p_i. In (ii) the new Delaunay edges connecting p_i to q, r, s have been added, and edge $\overline{qr}$ has already been flipped. Two more flips are necessary before the final state shown in (iii) is reached.

to perform *point-location* of p_i in the temporary triangulation DT_{i-1}.) The line segments connecting p_i to q, r, and s will be new Delaunay edges, by the same argument as described above. Next we check if, e.g., edge $\overline{qr}$ must be flipped. If so, the edges $\overline{qt}$ and $\overline{tr}$ are tested, and so on. We continue until no further edge currently forming a triangle with, but not containing p_i, needs to be flipped, and obtain the triangulation DT_i.

Lemma 3.1. *If the triangle of* DT_{i-1} *containing* p_i *is known, the structural work needed for computing* DT_i *from* DT_{i-1} *is proportional to the* degree m *of* p_i *in* DT_i.

Proof. Continued edge flipping replaces $m - 2$ conflicting triangles of DT_{i-1} by m new triangles in DT_i that have p_i as a vertex; cf. Figure 3.4. □

Lemma 3.1 yields an obvious $O(n^2)$ time algorithm for constructing the Delaunay triangulation of n points: We can determine the triangle of DT_{i-1} containing p_i within linear time, by inspecting all candidates. Moreover, the degree of p_i is trivially bounded by $n - 1$.

The last argument is quite crude. There can be single vertices in DT_i that do have a high degree, but their *average* degree is bounded by 6, as Lemmas 2.3 and 2.4 show. This fact calls for *randomization*. Suppose we pick p_n at random in S, then choose p_{n-1} randomly from $S - \{p_n\}$, and so on. The result is a random permutation $(p_1, p_2, \ldots, p_n)$ of the set S of sites.

If we insert the sites in this order, each vertex of DT_i has the same chance of being p_i. Consequently, the *expected value* of the degree of p_i is $O(1)$, and the expected total number of structural changes in the construction of DT_n is only $O(n)$, due to Lemma 3.1.

In order to find the triangle that contains p_i it is sufficient to inspect all triangles that are in conflict with p_i. The following lemma shows that the expected total number of all *conflicting triangles* so far constructed is only logarithmic.

Lemma 3.2. *For each $k < i$, let t_k denote the expected number of triangles in DT_k but not in DT_{k-1} that are in conflict with p_i. Then,*

$$\sum_{k=1}^{i-1} t_k = O(\log i).$$

Proof. Let C denote the set of triangles of DT_k that are in conflict with p_i. A triangle $T \in C$ belongs to $\mathrm{DT}_k \backslash \mathrm{DT}_{k-1}$ iff it has p_k as a vertex. As p_k is randomly chosen in DT_k, this happens with probability $3/k$. Thus, the expected number of triangles in $C \backslash \mathrm{DT}_{k-1}$ equals $3 \cdot |C|/k$. Since the expected size of C is less than 6 we have $t_k < 18/k$, hence

$$\sum_{k=1}^{i-1} t_k < 18 \cdot \sum_{k=1}^{i-1} 1/k = \Theta(\log i). \qquad \square$$

Suppose that T is a triangle of DT_i incident to p_i, see Figure 3.4(iii). Its edge $\overline{sr}$ is in DT_{i-1} incident to two triangles: to its *father*, F, that has been in conflict with p_i; and to its *stepfather*, SF, who is still present in DT_i. Any further site in conflict with T must be in conflict with its father or with its stepfather, as illustrated by Figure 3.5.

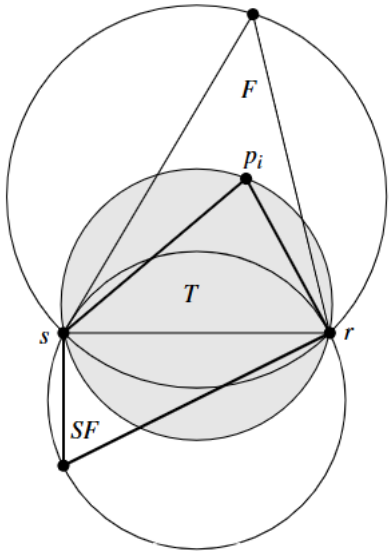


Figure 3.5. The circumcirle of T is contained in the union of the circumcircles of its father F and its stepfather SF.

This property can be exploited for quickly accessing all conflicting triangles. The *Delaunay tree* due to Boissonnat and Teillaud [148] is a *directed acyclic graph* that contains one node for each Delaunay triangle ever created during the incremental construction. (In other words, this graph reflects the *partial order* of the triangles imposed by the *construction history* of the current Delaunay triangulation.) Pointers run from fathers and stepfathers to their sons. The four triangles of DT_3 (three of which are infinite) are the sons of a dummy root node.

When p_i must be inserted, a Delaunay tree including all triangles up to DT_{i-1} is available. We start at its root and descend as long as the current triangle is in conflict with p_i. The above property guarantees that each conflicting triangle of DT_{i-1} will be found.

The expected number of steps this search requires is only $O(\log i)$, due to Lemma 3.2. Once DT_i has been computed, the Delaunay tree can easily be updated to include the new triangles. Thus, we have the following result.

Theorem 3.2. *The Delaunay triangulation of a set of n points in the plane can be constructed in expected time $O(n \log n)$, using expected linear space. The average is taken over the different orders of inserting the n sites.*

Note that we did not make any assumptions concerning the *distribution* of the sites in the plane; the incremental algorithm achieves its $O(n \log n)$ time bound for *every* possible input set. Only under a 'poor' insertion order can a quadratic number of structural changes occur, but this is unlikely.

The user annoyed by the uncertainty hidden in an *expected storage requirement* can apply a common trick to convert it into deterministic. Set some parameter c, and let a memory manager terminate the execution — and rerun the randomized construction — once the occupied space exceeds $c \cdot n$ (see Lemma 2.3 for a suitable choice of c). The expected runtime will still remain in $O(n \log n)$, though the constant in O will increase in dependency of c.

As a nice feature, the insertion algorithm is *online*. That is, it is capable of constructing DT_i from DT_{i-1} without knowledge of $p_{i+1}, \ldots, p_n$. In fact, *site deletions* can be handled similarly (as is sketched in Subsection 6.5.1, and also in Subsection 6.5.3 as a special case of an on-line algorithm which works in the more general setting of so-called *higher-order Voronoi diagrams*). This allows for *dynamizing* Delaunay triangulations and Voronoi diagrams, that is, maintaining these structures under insertion and deletion series of sites. Dynamization can also be based on the *divide & conquer* construction of Voronoi diagrams; see Section 3.3 for this earlier (and less efficient) approach, and Section 9.1.

Randomized geometric algorithms, though conceptually simple, tend to be tricky to analyze. Since Clarkson and Shor [229] introduced their technique, many researchers have been working on generalizing and simplifying the methods used. To mention but a few results, Boissonnat *et al.* [142] and Guibas *et al.* [390] have refined the methods of 'storing the past' in order to locate new conflicts quickly, Clarkson *et al.* [228] and Schwarzkopf [624] have generalized and simplified the analytic framework, and Seidel [631] systematically applied the technique of backward analysis first used by Chew [210]. A nice source is the monograph by Mulmuley [556]. The method in [390] for storing the past is applied in Section 6.5 for constructing order-k planar Voronoi diagrams.

If the set S of sites can be expected to be well distributed in the plane, *bucketing techniques* for accessing the triangle that contains a newly inserted site p_i have been used for speed-up. Joe [431], who implemented Sloan's algorithm [657], Su and Drysdale [664], who used a variant of Bentley *et al.*'s *spiral search* [124], and Lemaire and Moreau [502], who also gave probabilistic results for the higher-dimensional case, report on fast experimental runtimes. Snoeyink and van Kreveld [660] describe a simple way to precompute, in time $O(n \log n)$, an insertion order of the sites which then guarantees a *deterministic* $O(n)$-time incremental construction of the Delaunay triangulation — a task useful in the compression and transmission of triangular networks.

The arising issues of numerical stability have been addressed in Fortune [342], Sugihara [666], Jünger *et al.* [436], Shewchuk [648], and Avnaim *et al.* [108]. The main geometric primitive used by the algorithm is the *incircle test*, i.e., determining whether site p_i is enclosed by the circle defined by three other sites. As another practical issue, constructing Delaunay triangulations for '*imprecise point sets*' has been studied in Buchin *et al.* [172] and papers cited therein. In this setting, the sites are not known exactly but assumed to reside inside predefined regions (e.g., measurement tolerances), like small circles or squares.

Incremental insertion works well in the higher-dimensional case, and also for certain types of generalized Voronoi diagrams and their duals. We will see examples in Sections 6.1 and 6.5. Alternative methods for finding a *starting triangle*, i.e., a triangle of the current triangulation that contains the newly inserted site p_i, are discussed in Devillers *et al.* [266] and are reviewed, in the three-dimensional setting, in Section 6.1.

A technique similar to incremental insertion is *incremental search*. It starts with a single Delaunay triangle, and then incrementally discovers new ones, by growing triangles from edges of previously discovered triangles. This basic idea is used, e.g., in Maus [524] and in Dwyer [296]. It leads

to efficient expected-time Delaunay algorithms, also in higher dimensions; see [296].

The paper [664] gives a thorough experimental comparison of available Delaunay triangulation algorithms.

3.3. Divide & conquer

The first deterministic worst-case optimal algorithm for computing the Voronoi diagram has been presented by Shamos and Hoey [637]. In their *divide & conquer approach*, the set of point sites, S, is split by a dividing line into subsets L and R of about the same size. Then, the Voronoi diagrams $V(L)$ and $V(R)$ are computed *recursively*, that is, the same strategy is applied to the (smaller) point sets L and R. If only three or two points are left in a set, their diagram is constructed directly, in $O(1)$ time.

The essential part is in finding the split line, and in *merging* $V(L)$ and $V(R)$, to obtain $V(S)$. If these tasks can be carried out in time $O(n)$ then the overall running time, $T(n)$, is only $O(n \log n)$, as we have the recurrence relation

$$T(n) = 2 \cdot T(n/2) + O(n).$$

During the recursion, vertical or horizontal split lines can be easily found if the sites in S are sorted by their x- and y-coordinates beforehand.

The merge step involves computing the so-called *merge chain* $B(L, R)$, that is, the set of all Voronoi edges of $V(S)$ that separate regions of sites in L from regions of sites in R.

Suppose that the split line is vertical, and that L lies to its left.

Lemma 3.3. *The edges of $B(L, R)$ form a single* y-monotone *polygonal chain. (That is, any line parallel to the x-axis intersects the chain in only one point.) In $V(S)$, the regions of all sites in L are to the left of $B(L, R)$, whereas the regions of the sites of R are to its right.*

Proof. Let b be an arbitrary edge of $B(L, R)$, and let $l \in L$ and $r \in R$ be the two sites whose regions are adjacent to b. Since l has a smaller x-coordinate than r, the edge b cannot be horizontal, and the region of l must be to its left. □

Thus, $V(S)$ can be obtained by gluing together $B(L, R)$, the part of $V(L)$ to the left of $B(L, R)$, and the part of $V(R)$ to its right; see Figure 3.6, where $V(R)$ is depicted by dashed lines.

The polygonal chain $B(L, R)$ is constructed by finding a starting edge at infinity, and by tracing $B(L, R)$ through $V(L)$ and $V(R)$.

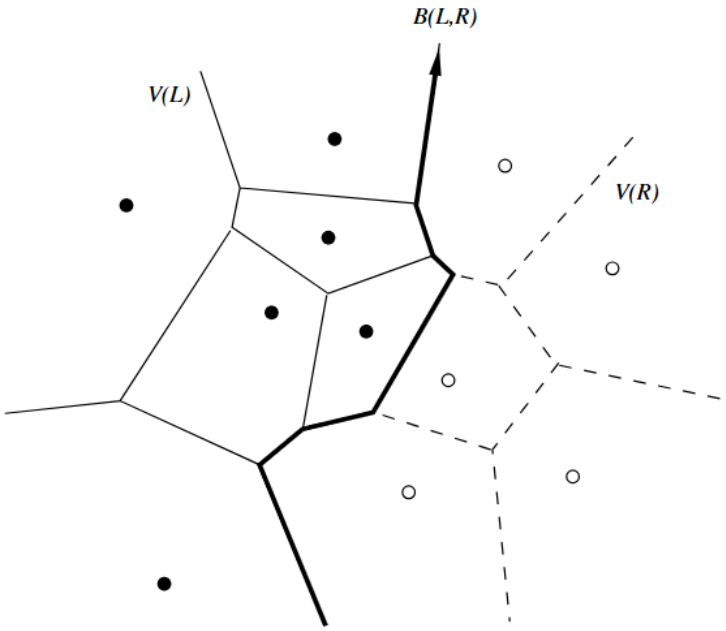


Figure 3.6. Merging $V(L)$ and $V(R)$ into $V(S)$.

Due to Shamos and Hoey [637], an unbounded starting edge of $B(L,R)$ can be found in $O(n)$ time by determining a line tangent to the *convex hulls* of L and R, respectively. Here we describe an alternative method by Lee [493], which was extended later by Chew and Drysdale [215], to be applicable for generalized Voronoi diagrams (Section 7.2). The unbounded regions of $V(L)$ and $V(R)$ are scanned simultaneously in cyclic order. For each non-empty intersection $\mathrm{VR}(l,L) \cap \mathrm{VR}(r,R)$, we test if it contains an unbounded piece of $B(l,r)$. If so, this must be an edge of $B(L,R)$, by Definition 2.1. Since $B(L,R)$ has two unbounded edges, by Lemma 3.3, this search will be successful. It takes time $|V(L)| + |V(R)| = O(n)$.

Now we describe how $B(L,R)$ is traced. Suppose that the current edge b of $B(L,R)$ has just entered the region $\mathrm{VR}(l,L)$ at point v while running within $\mathrm{VR}(r,R)$; see Figure 3.7. We determine the points v_L and v_R where b leaves the regions of l and of r, respectively. The point v_L is found by scanning the boundary of $\mathrm{VR}(l,L)$ counterclockwise, starting from v. In our example, v_R is closer to v than v_L, so that it must be the endpoint of edge b.

From v_R, $B(L,R)$ continues with an edge b_2 separating l and r_2. Now we have to determine the points $v_{L,2}$ and $v_{R,2}$ where b_2 hits the boundaries of the regions of l and r_2. The crucial observation is that $v_{L,2}$ cannot be situated on the boundary segment of $\mathrm{VR}(l,L)$ from v to v_L that we have just scanned; this can be inferred from the convexity of $\mathrm{VR}(l,S)$. Therefore, we

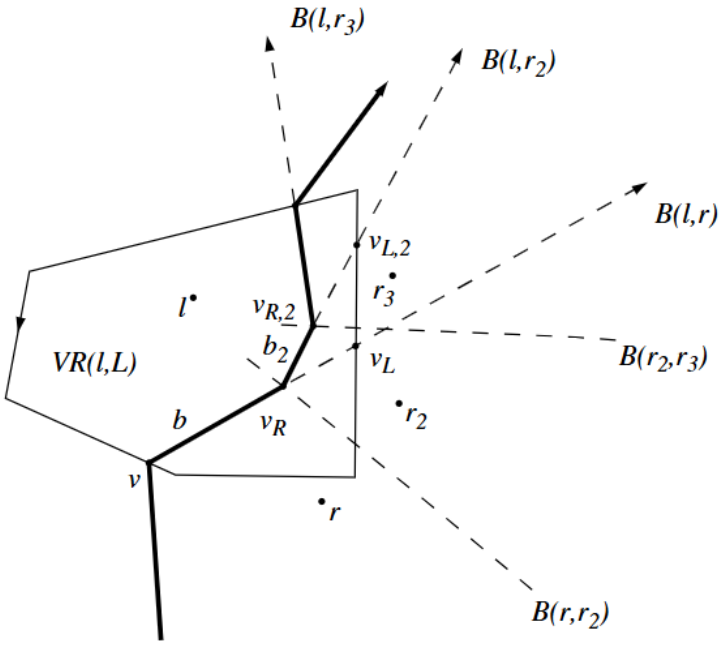


Figure 3.7. Computing the merge chain $B(L, R)$.

need to scan the boundary of $\mathrm{VR}(l, L)$ *only from* v_L *on*, in counterclockwise direction.

The same reasoning applies to $V(R)$; only here, region boundaries are scanned clockwise.

Even though the same region might be visited by $B(L, R)$ several times, no part of its boundary is scanned more than once. The edges of $V(L)$ that are scanned all lie to the right of $B(L, R)$. This part of $V(L)$, together with $B(L, R)$, forms a planar graph each of whose faces contains at least one edge of $B(L, R)$ in its boundary. As a consequence of Lemma 2.3, the size of this graph does not exceed the size of $B(L, R)$, times a constant. The same holds for $V(R)$. Therefore, the cost of constructing $B(L, R)$ is bounded by its size, once a starting edge is given. This leads to the following result.

Theorem 3.3. *The divide & conquer algorithm allows the Voronoi diagram of n point sites in the plane to be constructed within time $O(n \log n)$ and linear space, in the worst case. Both bounds are optimal.*

Of course, the divide & conquer paradigm can also be applied to the computation of the Delaunay triangulation $\mathrm{DT}(S)$. Guibas and Stolfi [394] give an implementation that uses the *quad-edge data structure* and only two

geometric primitives: an orientation test and an *incircle test*. Fortune [342] showed how to perform these tests accurately with finite precision.

Dwyer's implementation [295] uses vertical and horizontal split lines in turn, and Katajainen and Koppinen's [447] merges square buckets in a *quad-tree* order. Both papers report on favorable results.

Using the '*history*' of constructing a Voronoi diagram by divide & conquer, a *dynamic Voronoi diagram* algorithm can be designed. Gowda *et al.* [376] pursue this approach. Based on a general dynamization paradigm in Overmars [573], they propose the so-called *Voronoi tree* as a data structure for supporting insertions and deletions of sites in $O(n)$ time, when n is the current size of the point set. The root of this binary tree stores the diagram $V(S)$ for the entire set S of sites, and its two children are the roots of the Voronoi trees for the subsets L and R that S got split into. Insertion or deletion of a site p amounts to an update in the Voronoi tree, which can be accomplished by traversing a path between its root and the leaf corresponding to p.

A related concept is the *Voronoi diagram for moving sites*, also called the *kinetic Voronoi diagram*, where the diagram is to be updated during a continuous movement of its sites along certain trajectories. We will elaborate on this topic to some extent in Section 9.1.

Divide & conquer algorithms are candidates allowing for *parallelization*. Efficient algorithms for computing in parallel the Voronoi diagram or the Delaunay triangulation have been proposed. We refer to the paper by Blelloch *et al.* [133] for references and for a practical parallel algorithm for computing $\mathrm{DT}(S)$. They highlight an algorithm by Edelsbrunner and Shi [314] that uses a *lifting map* for S (see Section 3.5) to construct a chain of Delaunay edges that divides S. They show experimentally that their implementation is comparable in *work* (i.e., product of runtime and processor number) to the best sequential algorithms.

In all the divide & conquer approaches described in this section, the emphasis is on the *bottom-up* phase — how to accomplish the merge step. This step gets considerably complicated for Voronoi diagrams in more general settings, because the merge chain may cycle and even heavily disconnect.

An alternative approach, with emphasis on the *top-down* phase — how to perform the divide step — has been recently considered in Aichholzer *et al.* [34, 27]. It is based on constructing the *medial axis* of a general planar shape, and exploits the tree structure of a medial axis to calibrate the divide step. The conquer step is trivial and consists of simply concatenating two partial medial axes. This algorithm, which also

works for quite general planar objects as sites, is described in detail in Section 5.5.

3.4. Plane sweep

The well-known *plane sweep algorithm* (also called *sweep line* algorithm) by Bentley and Ottmann [121] computes the intersections of n line segments in the plane by moving a vertical line, H, from left to right across the plane. The line segments currently intersected by H are stored in y-order. This order must be updated whenever H reaches an endpoint of a line segment, or an intersection point. To discover the intersection points in time, it is sufficient to check, after each update of the y-order, those pairs of line segments that have just become neighbors on H.

It is tempting to apply the same approach to Voronoi diagrams, by keeping track of the Voronoi edges that are currently intersected by the vertical sweep line. The problem is in discovering new Voronoi regions in time. By the time the sweep line hits a new site, it has been intersecting Voronoi edges of its region for a while.

Fortune [344] was the first to find a way around this difficulty. He suggested a *planar transformation* under which each point site becomes the leftmost point of its Voronoi region, so that it will be the first point hit during a left-to-right sweep. His transformation does not change the combinatorial structure of the Voronoi diagram.

Later, Seidel [629] and Cole [230] have shown how to avoid this transformation altogether. They consider the Voronoi diagram of the point sites to the left of the sweep line H and of H itself, considered an *additional site* of straight-line shape; see Figure 3.8. Because the bisector of a line and a non-incident point is a parabola, the boundary of the Voronoi region of H is a connected chain of parabolic segments whose top- and bottommost edges tend to infinity. This chain is called the *wavefront*, W.

Let p be a point site to the left of H. Any point to the left of, or on, the parabola $B(p, H)$ is not farther from p than from H; hence, it is *a fortiori* closer to p than to any site to the right of H. Consequently, as the sweep line moves on to the right, the waves must follow because the sets to the left of $B(p_i, H)$ grow. On the other hand, each Voronoi edge to the left of W that currently separates the regions of two sites p_i, p_j will be (part of) a Voronoi edge in $V(S)$.

During the sweep, there are two types of events that cause the structure of the wavefront to change, namely when a new wave appears in W, or when an old wave disappears. The former one, called a *site event*, happens each time the sweep line hits a new site, e.g., p_6 in Figure 3.8. At that very

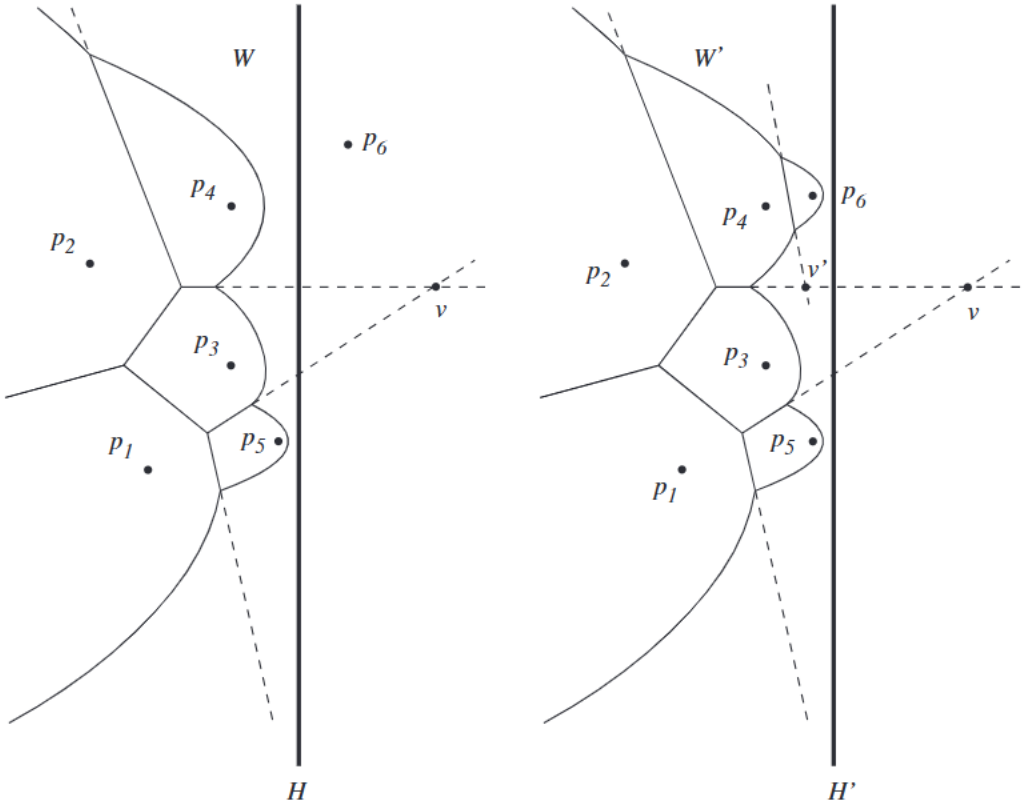


Figure 3.8. Voronoi diagrams of the sweep line H, and of the points to its left, which have been swept over already.

moment, $B(H, p_6)$ is a horizontal line through p_6 (as a degenerate case). A little later, its left halfline unfolds into a parabola that must be inserted into the wavefront by gluing it onto the wave of p_4 (which thereby is split into two waves of W.)

For describing the other type of event, let p, q be two point sites whose waves are neighbors in W. Their bisector, $B(p, q)$, gives rise to a Voronoi edge to the left of W. Its prolongation into the region of H is called a *spike*. In Figure 3.8 spikes are depicted as dashed lines; one can think of them as tracks along which the waves are moving. A wave disappears from W when it arrives at the point where its two bounding spikes intersect. Its former neighbors become now adjacent in the wavefront. This is called a *spike event*.

In Figure 3.8, the wave of p_3 would disappear at intersection point v, if the new site, p_6, that causes a site event prior to that did not exist. But after the wave of p_6 has been inserted, a spike event at point v' will take place, and the corresponding wave of p_4 will disappear.

While keeping track of the wavefront one can easily maintain the portion of the Voronoi diagram to the left of W. Spikes correspond to

Voronoi edges. At each event, a new adjacency between two waves occurs, and with it, a new spike. That is, a new Voronoi edge has been identified. In particular, a Voronoi vertex gets constructed at each spike event. As soon as all point sites have been detected and all spike intersections have been processed, $V(S)$ is obtained by removing the wavefront and extending to infinity all spikes that have been intersecting it.

Even though one site may contribute several waves to the wavefront, the following holds.

Lemma 3.4. *The size of the wavefront is $O(n)$.*

Proof. Since any two parabolic bisectors $B(p,H), B(q,H)$ can cross at most twice, the number of waves of the wavefront is bounded by $\lambda_2(n) = 2n-1$, where $\lambda_s(n)$ denotes the maximum length of a *Davenport–Schinzel sequence* over n symbols in which no two symbols appear s times each in alternating positions; see [81]. □

The wavefront can be implemented by a *balanced search tree* that stores the parabola segments in y-order. This enables us to insert a wave, or remove a wave segment, in time $O(\log n)$.

Before the sweep starts, the point sites are sorted by increasing x-coordinates and inserted into an event *priority queue*. (See e.g. [234] or [538] for a description of basic data structures.) After each update of the wavefront, the newly adjacent spikes are tested for intersection. If they intersect at some point v, we insert into the event queue the time (i.e., the position x of the sweep line) when the wave segment between the two spikes arrives at v. Since the point v is a Voronoi vertex of $V(S)$, there are only $O(n)$ events caused by spike intersections. In addition, each of the n sites causes an event. For each active spike we need to store only its first intersection event. Thus, the size of the event queue never exceeds $O(n)$. We obtain the following result.

Theorem 3.4. *Plane sweep provides an alternative way of computing the Voronoi diagram of n points in the plane within $O(n \log n)$ time and $O(n)$ space.*

McAllister *et al.* [527] have pointed out a subtle difference between the sweep technique and the two methods mentioned before. The divide & conquer algorithm computes $\Theta(n \log n)$ many vertices, even though only a linear number of them appears in the final diagram. The randomized incremental construction method performs an expected $\Theta(n \log n)$ number of conflict tests. Both tasks, constructing a Voronoi vertex and testing a subset of sites for conflict, are usually handled by subroutines that deal

directly with point coordinates, bisector equations, etc. They can become quite costly if we consider sites more general than points, and distance measures more general than the Euclidean distance; see Chapters 5 and 7.

The sweep algorithm, on the other hand, processes $O(n)$ spike events, and thus constructs only that many vertices.

The data structure to store the Voronoi diagram can be any standard data structure for planar subdivisions. See, for example, the *doubly connected edge list* [552] or the *quad-edge* data structure [394].

Note finally that the plane-sweep technique reduces a two-dimensional 'static' problem (for instance, the construction of a Voronoi diagram in the plane) to a one-dimensional 'dynamic' problem (here, the maintenance of the wavefront representation on the sweep line). This enables us to use efficient one-dimensional data structures, like balanced binary trees and priority queues. A kind of opposite philosophy underlies the algorithmic technique presented in the subsequent section: An interpretation of the planar problem in 3-space is sought, in order to gain insight into its structural properties, and to apply known but seemingly unrelated algorithms.

3.5. Lifting to 3-space

The following approach employs the powerful method of *geometric transformation*, which leads to an optimal algorithm for constructing the planar Delaunay triangulation.

Let $P = \{(x_1, x_2, x_3) \mid x_1^2 + x_2^2 = x_3\}$ denote the paraboloid depicted in Figure 3.9. For each point $x = (x_1, x_2)$ in the plane, let $x' = (x_1, x_2, x_1^2 + x_2^2)$ denote its lifted image on P.

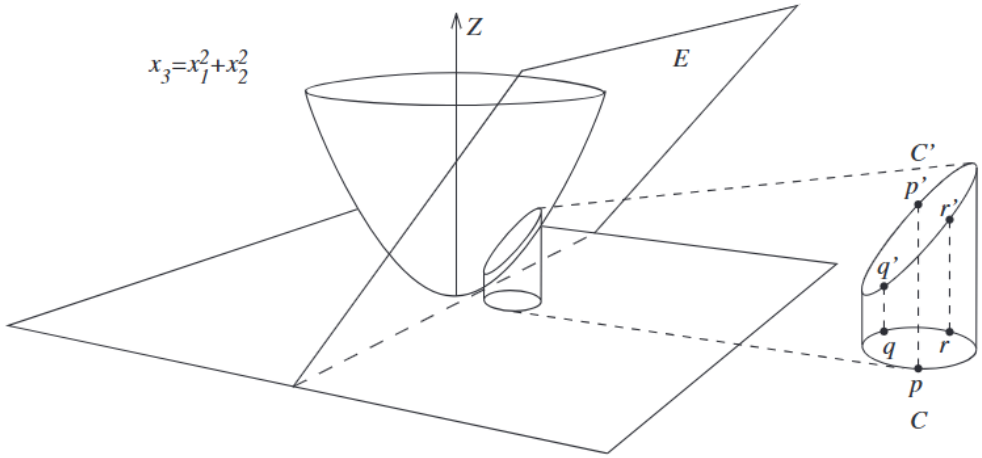


Figure 3.9. Lifting circles onto the paraboloid.

Lemma 3.5. *Let C be a circle in the plane. Then C' is a planar curve on the paraboloid P.*

Proof. Suppose that C is given by the equation

$$r^2 = (x_1 - c_1)^2 + (x_2 - c_2)^2 = x_1^2 + x_2^2 - 2x_1c_1 - 2x_2c_2 + c_1^2 + c_2^2.$$

By substituting $x_1^2 + x_2^2 = x_3$ we obtain

$$x_3 - 2x_1c_1 - 2x_2c_2 + c_1^2 + c_2^2 - r^2 = 0$$

for the points of C'. This equation defines a plane in 3-space. □

This lemma has an interesting consequence. Let S be a finite point set in the plane, and denote by S' its lifted image on P. The convex hull of S' is a *convex polyhedron* in 3-space. All points of S' appear as polyhedron vertices, because S' is in *convex position*, by construction. (Compare Figure 3.1 (i) in Section 3.1 for a picture in one dimension less.) By the *lower convex hull* of S' we mean that part of the convex hull which is visible from $x_3 = -\infty$.

Theorem 3.5. *The Delaunay triangulation* $\mathrm{DT}(S)$ *of S equals the vertical projection onto the x_1x_2-plane of the lower convex hull of S'.*

Proof. Let p, q, r denote three point sites of S. By Lemma 3.5, the lifted image, C', of their circumcircle C lies on a plane, E, that cannot be vertical. Under the lifting mapping, the points inside C correspond to the points on the paraboloid P that lie *below* the plane E.

By Theorem 2.1, p, q, r define a triangle of the Delaunay triangulation iff their circumcircle contains no further site. Equivalently, no lifted site s' is below the plane E that passes through p', q', r'. But this means that p', q', r' define a facet of the lower convex hull of S'. □

Because there exist $O(n \log n)$ time algorithms for computing the convex hull of n points in 3-space — see e.g. Preparata and Shamos [596], and for a survey of the various available construction methods — see Seidel [633], we have obtained another optimal algorithm for the Voronoi diagram.

The connection between Voronoi diagrams and convex hulls has first been studied by Brown [169] who used the *inversion transform*. The simpler lifting mapping has been used, e.g., in Edelsbrunner and Seidel [310]. We shall see several applications and generalizations in Chapter 6. In [310] also the following fact is observed.

For each point $s = (s_1, s_2)$ of S, consider the paraboloid

$$P_s = \{(x_1, x_2, x_3) \mid (x_1 - s_1)^2 + (x_2 - s_2)^2 = x_3\}.$$

If these paraboloids were opaque, and of pairwise different colors, an observer looking from $x_3 = -\infty$ upwards would see the Voronoi diagram $V(S)$. In fact, for $p, q \in S$, the projection $x = (x_1, x_2)$ of a point $(x_1, x_2, x_3) \in P_p \cap P_q$ belongs to their bisector $B(p, q)$; and there is no site s closer to x than p and q iff (x_1, x_2, x_3) lies below all paraboloids P_s. That is, $V(S)$ is the vertical projection onto the plane of the *lower envelope* of these paraboloids.

Instead of the paraboloids P_s one could use the surfaces

$$\{(x_1, x_2, x_3) = f((x_1 - s_1)^2 + (x_2 - s_2)^2)\}$$

generated by *any* function f that is strictly increasing. For example, $f(x) = \sqrt{x}$ gives rise to cones of slope 45° with apices at the sites. This setting illustrates the concept of circles expanding from the sites at equal speed, as mentioned after the proof of Lemma 2.1. Coordinate x_3 represents time.

In order to visualize a Voronoi diagram on a graphic screen one can feed the n surfaces to a z-buffer, and eliminate by brute force those parts not visible from below; cf. Section 11.1 for so-called *pixel Voronoi diagrams*.

Finally, we would like to mention a nice connection between the two ways of obtaining the Voronoi diagram by means of lower envelopes of paraboloids explained above; it goes back to [310]. For a point $w = (w_1, w_2, w_3)$, let $\hat{w}$ denote its mirror image $(w_1, w_2, -w_3)$. If we apply to 3-space the mapping which sends

$$x = (x_1, x_2, x_3) \text{ to } (x_1, x_2, x_3 - (x_1 - s_1)^2 - (x_2 - s_2)^2)$$

then, for each point s in the plane, the paraboloid P_s corresponds to the tangent plane of the paraboloid $\hat{P}$ at the point $\hat{s}'$, where s' denotes the lifted image of s; compare the plane equation derived in the proof of Lemma 3.5, letting $c = s$ and $r = 0$.

Lower envelope representations of generalized Voronoi diagrams will be briefly treated in Section 7.5. The corresponding surfaces and the resulting diagram, respectively, are sometimes called *Voronoi surfaces* and their *minimization diagram* in the literature. They are very useful for obtaining algorithms and, in addition, bounds on the combinatorial complexity of Voronoi diagrams.

Chapter 4

ADVANCED PROPERTIES

We have seen, in Chapters 2 and 3, quite a few basic properties of the Voronoi diagram for point sites in the plane. Many more shall be described in later chapters, along with generalizations of the Voronoi diagram concerning its sites and its underlying distance function, as certain properties extend to such settings in a natural way. The present chapter is devoted to some advanced features of the classical Voronoi diagram and its dual, the Delaunay triangulation.

4.1. Characterization of Voronoi diagrams

The process of constructing the Voronoi diagram for n point sites can be seen as an assignment of a planar convex region to each of the sites, according to the 'nearest neighbor rule'. We now address the following, in some sense inverse, question: Given a partition of the plane into n convex regions (which are then necessarily polygonal), do there exist sites, one for each region, such that the nearest neighbor rule is fulfilled? In other words, when is a given *convex partition* the Voronoi diagram of some set of sites?

Whether a *given* set of sites induces a given convex partition as its Voronoi diagram is, of course, easy to decide by exploiting symmetry properties among the sites. For the same reason, it is easy to check whether a given triangulation is Delaunay, by exploiting the *empty-circle property* of its triangles, stated in Theorem 2.1. Conditions for a given graph to be isomorphic to the Delaunay triangulation of *some* set of sites are mentioned, e.g., in the survey article by Fortune [343]. Below we concentrate on the recognition of Voronoi diagrams *without* knowing the sites. The process of restoring the sites, if they exist, is also called *Voronoi diagram inversion*; see Schoenberg *et al.* [619] for an overview.

Questions of this kind arise in *facility location* and in the recognition of biological *growth models* (as reported, e.g., in Suzuki and Iri [671]) and,

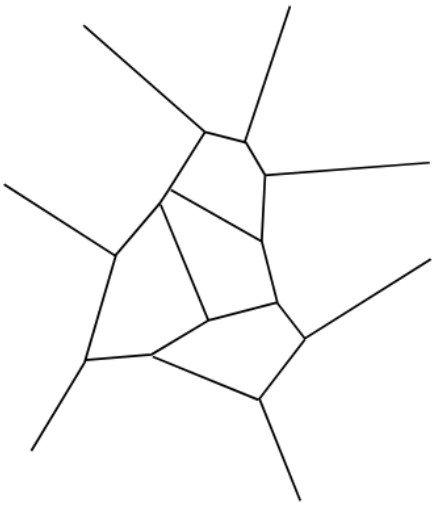


Figure 4.1. A Voronoi diagram?

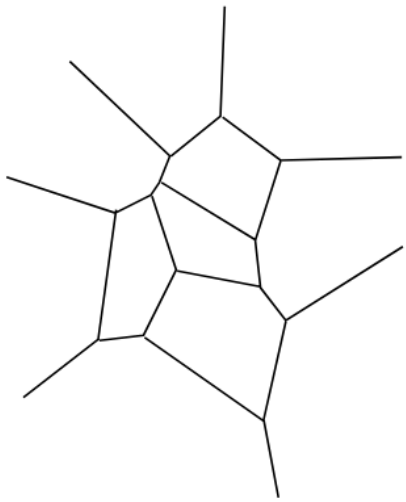


Figure 4.2. This one, yes.

in particular, in the so-called *gerrymander problem* mentioned in Ash and Bolker [79]: When the sites are regarded as polling places and election law requires each person to vote at the respective closest polling place, the election districts form a Voronoi diagram. If the legislature draws the district lines first, how can we tell whether election law is satisfied? (Figures 4.1 and 4.2).

Let R_i and R_j be two of the given regions. Assume that they share a common edge, and let h_{ij} be the straight line containing that edge. Further, let σ_{ij} denote the reflection at line h_{ij}.

Lemma 4.1. *A convex partition $R_1, \ldots, R_n$ of the plane defines a Voronoi diagram if and only if there exists a point p_i for each region R_i such that*

(1) $p_i \in R_i$ (*containment condition*),
(2) $\sigma_{ij}(p_i) = p_j$ *if R_j is adjacent to R_i* (*reflection condition*).

Proof. If we do have a Voronoi diagram then its defining sites exist and obviously fulfill (1) and (2). To prove the converse, assume that points $p_1, \ldots, p_n$ fulfilling both conditions exist. Take any region R_i and any point x therein. We show that $d(x, p_i)$ is minimum.

To get a contradiction, suppose p_j, $j \neq i$, is closest to x. Consider an edge of R_j that is intersected by the line segment $\overline{xp_j}$, and let R_k be the region adjacent to R_j at that edge; see Figure 4.3. Note that $k = i$ may happen. By convexity of R_j and by (1), the line h_{jk} separates p_j from x. Hence, by (2), we get $d(x, p_k) < d(x, p_j)$, which is a contradiction.

We conclude that p_i is closest to x among $p_1, \ldots, p_n$, which implies that R_i is the region of p_i in the Voronoi diagram $V(\{p_1, \ldots, p_n\})$. □

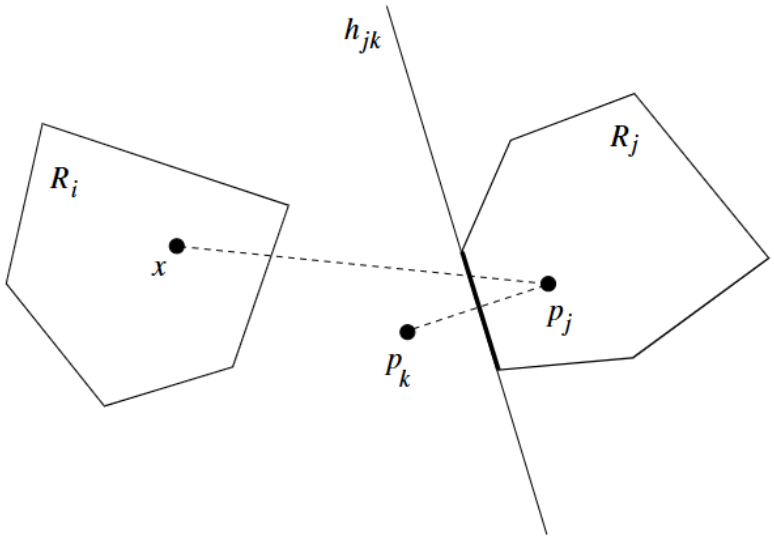


Figure 4.3. Region R_i must be closest to point x.

Based on Lemma 4.1, the recognition problem can now be formulated as a *linear programming problem*; see Hartvigsen [398]. We first exploit the reflection condition to get a system of linear equations.

Reflection at a line is an affine transformation, so we may write $\sigma_{ij}(x)$ as $A_{ij}x + b_{ij}$, for an appropriate matrix A_{ij} and vector b_{ij}. Consider a *depth-first search* order (see e.g. [363]) of the regions, in a way such that for each region R_i an adjacent region R_{i+1} is known. To get a linear system in x, put

$$\begin{aligned} p_1 &= x, \\ p_2 &= A_{12}x + b_{12} := C_2x + d_2, \\ p_3 &= A_{23}(A_{12}x + b_{12}) + b_{23} := C_3x + d_3 \end{aligned}$$

and so on. This expresses all points p_i in terms of p_1 by using $n - 1$ adjacencies among the regions. Each of the remaining adjacencies now gives an equation of the form

$$A_{ij}(C_ix + d_i) + b_{ij} = C_jx + d_j.$$

This system has at most $3n-3-(n-1) = 2n-2$ equations by Lemma 2.3. If it has no solution, or a unique solution, then we are done. In the former case, we cannot have a Voronoi diagram. In the latter, we get the coordinates of the first candidate site $p_1 = x$. The corresponding other sites are obtained simply by reflection. It remains to test these sites for containment in their regions.

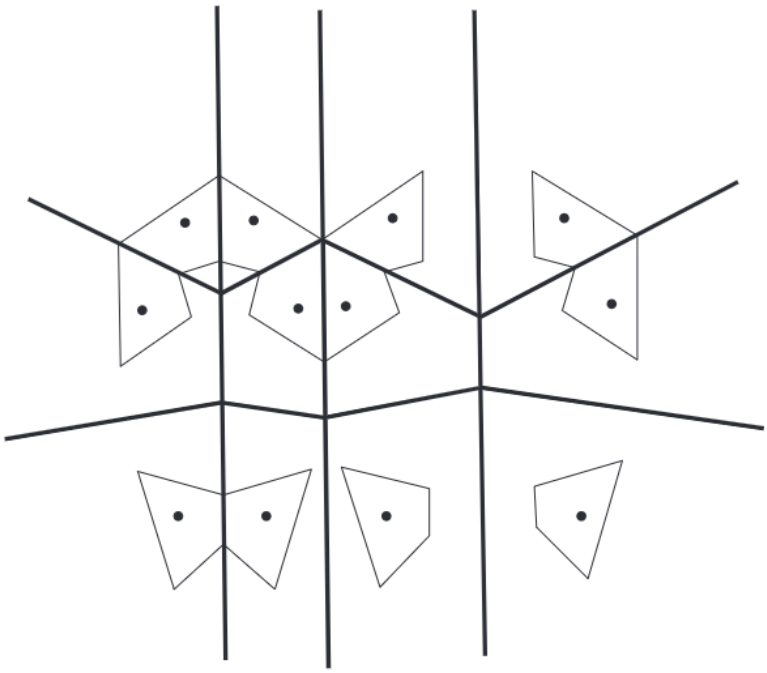


Figure 4.4. Sites might be chosen within the polygonal areas.

Setting up the system, solving it, and testing for containment can be accomplished in time $O(n)$ by standard methods. Note that only the equations of the lines bounding the regions and the adjacency information among the regions are needed. No coordinates of the region vertices are required. This is particularly interesting for the recognition problem in *higher dimensions*, to which the method above generalizes naturally.

The solution space of the linear system above may have dimension 1 or 2. Figure 4.4 reveals that certain symmetries among the regions lead to non-unique solutions. Now the containment condition is exploited to get, in addition, a set of inequalities for x. Consider each region R_i as the intersection of all halfplanes bounded by the lines h_{ij}. Then $p_i \in R_i$ gives a set of inequalities of the form

$$p_i^T t_{ij} \leq a_{ij},$$

which, by plugging in $p_i = C_i x + d_i$, yields

$$(C_i^T t_{ij})x \leq a_{ij} - d_i^T t_{ij}.$$

Finding a feasible solution of the corresponding linear program means finding a possible site $p_1 = x$ for R_1 which, by reflection, gives all the desired sites. Since we deal with a linear program with $O(n)$ constraints and of

constant dimension (actually two), also only linear time (Megiddo [531]) is spent in this more complicated case.

Theorem 4.1. *Let $\mathcal{C}$ be a partition of the plane into n convex regions, given by halfplanes supporting the regions and by adjacencies among regions. $O(n)$ time suffices for deciding whether $\mathcal{C}$ is a Voronoi diagram, and also for restoring a suitable set of sites in case of its existence.*

This result is clearly optimal, and the underlying method easy to program. A generalization to higher dimensions is straightforward. Still, the method has to be used with care, as even a slight deviation from the correct Voronoi diagram (stemming from imprecise measurement or numerical errors) will cause the method to classify $\mathcal{C}$ as non-Voronoi. Suzuki and Iri [671] give a completely different method capable of *approximating* $\mathcal{C}$ by a Voronoi diagram.

Lemma 4.1 extends to more general Voronoi-like partitions. The characterizing configuration of points is then called a *reciprocal figure*, valuing Clerk Maxwell's work [525] of 1864. A nice survey on this subject is Ash *et al.* [80]. In particular, if the partition $\mathcal{C}$ is polygonal, then the containment and reflection conditions can be relaxed to

(1b) *orthogonality*: if polygons R_i and R_j share an edge e then the line segment $\overline{p_i p_j}$ is orthogonal to e, and
(2b) *orientation*: any ray which is parallel to the vector $(p_j - p_i)$ and intersects e meets R_i first.

Interestingly, the existence of a reciprocal figure $\{p_1, \ldots, p_n\}$ for $\mathcal{C}$, which is now also called an *orthogonal dual*, is equivalent to the property that $\mathcal{C}$ can be realized as the *equilibrium state* of a spider web: The edges of $\mathcal{C}$ allow positive tensions that balance out at all vertices of $\mathcal{C}$ — a property of use in the statics of *plane frameworks*; see e.g. Crapo and Whiteley [235].

Consult Figure 4.5. Suitable tensions for the edges $e_{ij} = R_i \cap R_j$ are the lengths of the vectors $(p_j - p_i)$, or constant multiples thereof. These tensions trivially add up to zero for the regions $R_1, \ldots, R_k$ around each vertex v, because the corresponding line segments $\overline{p_i p_{i+1}}$ border a face dual to v (a triangle in the non-degenerate case).

Moreover, by *Maxwell's theorem* [525], the existence of such tensions characterizes partitions of the plane that can be obtained as the *boundary projections* of convex polyhedra in 3-space; see also [235].

Clearly, for the Voronoi diagram $V(S)$ its set S of sites is a possible orthogonal dual. On the other hand, the *vertices* of $V(S)$ constitute an orthogonal dual for the Delaunay triangulation $\mathrm{DT}(S)$; the nonconvex

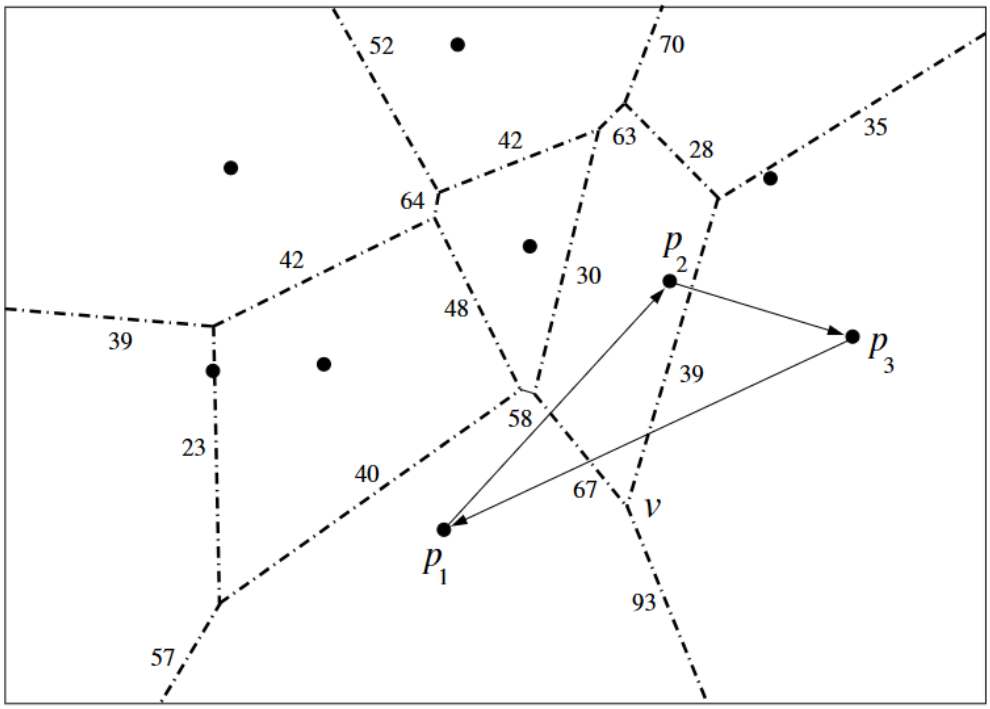


Figure 4.5. Spider web with orthogonal dual (•). Any scaled and translated copy is a valid orthogonal dual, too. Numbers at edges express tensions in equilibrium state at each vertex. This is witnessed for vertex v by the side lengths of its dual triangle $p_1p_2p_3$.

unbounded face of $\mathrm{DT}(S)$ can be treated consistently. Indeed, in Section 3.5 we have seen the existence of a projection polyhedron for $\mathrm{DT}(S)$, namely, the convex hull of the point set S lifted to $\mathbf{R}^3$.

The correspondences above still hold in higher dimensions; see Aurenhammer [88, 89]. By the relationship of Voronoi diagrams to convex polyhedra in the next dimension (Subsection 6.2.2), a partition $\mathcal{C}$ admits an orthogonal dual if and only if $\mathcal{C}$ is the *power diagram* of some set of spheres; the dual points p_i serve as sphere centers.

Various types of convex cell complexes can be identified as polyhedral projections in this way, including simple complexes, arrangements of hyperplanes, and higher-order Voronoi diagrams — structures we will encounter in Chapter 6. Moreover, projection polyhedra can be found efficiently [89].

The topic of lifting (not necessarily convex) cell complexes is briefly addressed in Subsection 6.3.3. For equilibrium states in nonconvex polygonal planar partitions, the *Maxwell–Cremona theorem* gives an elegant answer; negative tensions of edges (stresses) will now occur in the characterization. This theorem has surprising applications different from its direct use in *terrain recognition* and *scene analysis* — for instance to

the long-standing open problem of *untangling simple polygons* in the plane, which has been solved in Connelli *et al.* [231].

4.2. Delaunay optimization properties

The Delaunay triangulation, $\mathrm{DT}(S)$, of a set S of n sites in the plane possesses a host of nice and useful properties many of which are well known and understood nowadays. As being the geometric dual of the Voronoi diagram $V(S)$, $\mathrm{DT}(S)$ comprises the proximity information inherent to S in a compact manner. Apart from the present section, various properties of $\mathrm{DT}(S)$ and their applications are described in Chapter 8 and, in particular, in Section 8.2. Here we look at $\mathrm{DT}(S)$ as a triangulation *per se* and concentrate on parameters which are optimized by $\mathrm{DT}(S)$ over all possible triangulations of the point set S.

Recall that a *triangulation* (or *triangular network*), T, of S is a maximal set of non-crossing line segments spanned by the sites in S. In other words, T defines a partition of the convex hull of S into triangles whose vertex set is exactly S. The number of different such partitions is large (in fact exponential, $\Omega(2.33^n)$) for *every* n-point set S in general position [43], which makes the optimality results to be presented even more valuable.

Let us call T *locally Delaunay* if, for each of its convex quadrilaterals Q, the corresponding two (adjacent) triangles have *circumcircles* empty of vertices of Q. Clearly, $\mathrm{DT}(S)$ is locally Delaunay because the circumcircles for its triangles are totally empty of sites; see Theorem 2.1. Interestingly, the local property also implies the global one.

Theorem 4.2. *A triangulation of S is locally Delaunay if and only if it equals the Delaunay triangulation,* $\mathrm{DT}(S)$.

Proof. Let T be a triangulation of S and assume that T is locally Delaunay. We show that, for each triangle Δ of T, its circumcircle $C(\Delta)$ is empty of sites in S.

Assuming the contrary, let s be inside $C(\Delta)$ for some $s \in S$ and some Δ in T. Observe $s \notin \Delta$ and let e be the edge of Δ closest to s. Suppose, without loss of generality, that (Δ, e, s) maximizes the angle spanned by e at s, for all such triples (triangle, edge, site).

See Figure 4.6. Because of s, edge e cannot be an edge of the convex hull of S. Let triangle Δ' be adjacent to Δ at e, and let s' be the third vertex of Δ'. As T is locally Delaunay, s' lies outside $C(\Delta)$, hence $s \neq s'$. Now, observe that s is enclosed by $C(\Delta')$, and let e' be the edge of Δ' closest to s. The angle at s for (Δ', e', s) is larger than that for (Δ, e, s), which gives a contradiction. □

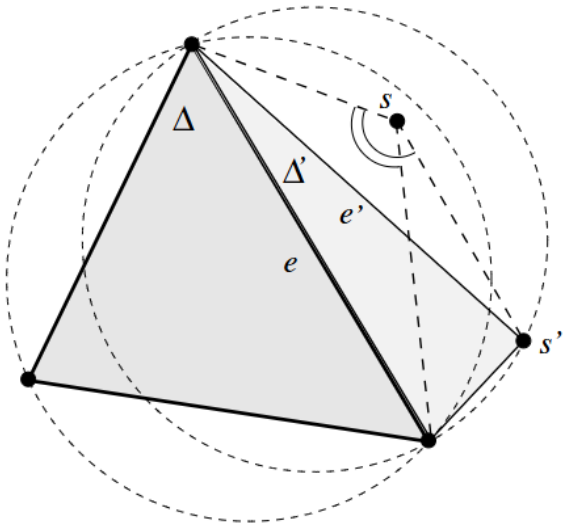


Figure 4.6. Illustration of the proof of Theorem 4.2.

An *edge flip* in a triangulation T of S is the exchange of the two diagonals in one of T's convex quadrilaterals; see Section 3.2. Call an edge flip *good* if, after the flip, the triangulation inside the quadrilateral (consisting of only 2 triangles) is locally Delaunay. Repeated exchange of diagonals in the same quadrilateral always produces an alternating sequence of good and not good flips. Theorem 4.2 now can be used to prove that $\mathrm{DT}(S)$ optimizes various quality measures, by showing that each good flip increases quality. Any sequence of good flips then terminates at the global optimum, the Delaunay triangulation. The length of such a sequence is bounded by $\binom{n}{2}$, as an edge that gets flipped away cannot reappear; see the argument on *flip distances* in planar triangulations given in Subsection 6.3.3.

One of the most prominent quality measures concerns the angles occurring in a triangulation. Recall that the number of edges (and thus of triangles) does not depend on the way of triangulating S, and let t be the number of triangles for S. The *angularity* of a triangulation is defined to be the sorted list of angles $(\alpha_1, \ldots, \alpha_{3t})$ of its triangles, in ascending order. A triangulation is called *equiangular* if it possesses lexicographically largest angularity among all possible triangulations for S.

As a matter of fact, every good flip increases angularity. Figure 4.7 gives evidence for this fact. Lawson [486] called a triangulation *locally equiangular* if no flip can increase angularity. Locally equiangular thus is equivalent to locally Delaunay. Sibson [652] and Lee [492] first proved Theorem 4.2, showing that locally equiangular triangulations are Delaunay and hence are unique. Edelsbrunner [300] observed that $\mathrm{DT}(S)$ is equiangular (in the global sense) as the global property implies the local one.

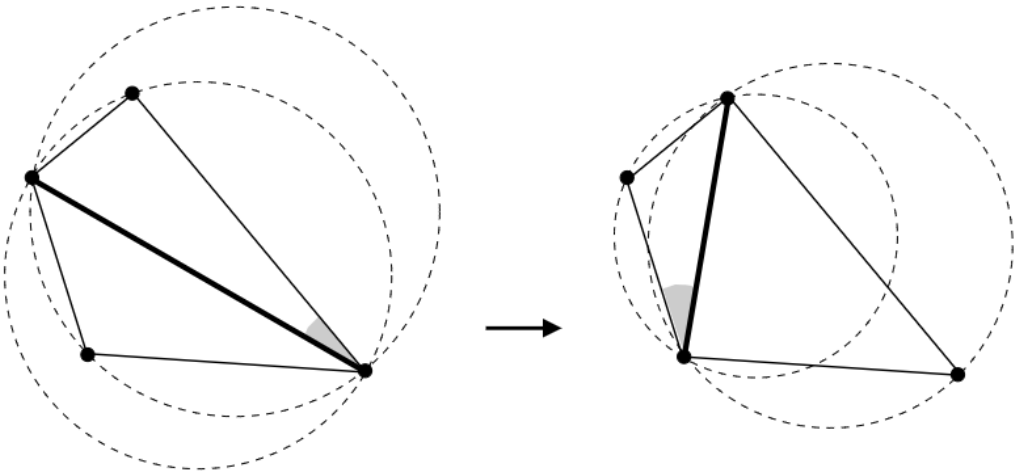


Figure 4.7. Equiangularity and the empty circle property.

In case of cocircularities among the sites, $\mathrm{DT}(S)$ is not a full triangulation; see Chapter 2. Mount and Saalfeld [550] showed that $\mathrm{DT}(S)$ can be completed by retaining local equiangularity, in $O(n \log n)$ time.

Theorem 4.3. *Let S be a finite set of sites in the plane. A triangulation of S is equiangular only if it is a completion of* $\mathrm{DT}(S)$.

The equiangular triangulation obviously maximizes the minimum angle that occurs in all the triangles. This property is desirable for *mesh generation* applications in *terrain modeling* or for the *finite element method*, as was first observed in Lawson [486] and McLain [528]. By Theorem 4.3, such triangulations can be computed in $O(n \log n)$ time with the Delaunay triangulation algorithms in Chapter 3.

More recently, it has been observed that several other parameters are optimized by $\mathrm{DT}(S)$. All the properties listed below can be proved by observing that every good edge flip locally optimizes the respective parameter.

Consider the *circumcircle* for each triangle in a triangulation, and measure *coarseness* by the largest such circle that arises; see Figure 4.7 again. As a matter of fact, $\mathrm{DT}(S)$ minimizes coarseness among all possible triangulations for S; see D'Azevedo and Simpson [240]. We may define coarseness also by taking *smallest enclosing circles* rather than circumcircles. (Note that the smallest enclosing circle differs from the circumcircle iff the triangle is obtuse.) D'Azevedo and Simpson proved that $\mathrm{DT}(S)$ minimizes coarseness in this sense, and Rajan [598] showed that this property of Delaunay triangulations — unlike others — generalizes to higher dimensions.

Similarly, *fatness* of a triangulation may be defined as the sum of *inradii*, that is, radius of the largest circle inscribed to each triangle. Lambert [480] showed that $\mathrm{DT}(S)$ maximizes fatness, or equivalently, the mean inradius.

Given an individual function value (height) $h(p)$ for each site $p \in S$, every triangulation T of S defines a *triangular surface* in 3-space. The *roughness* of such a surface may be measured by

$$\sum_{\Delta \in T} |\Delta| \cdot (\alpha^2 + \beta^2),$$

where $|\Delta|$ is the area of Δ, and α and β denote the slopes that the corresponding triangle in 3-space forms with the x-axis and the y-axis, respectively. In other words, roughness is the integral of the squared gradients. As has been shown by Rippa [605], roughness is minimum for the surface obtained from $\mathrm{DT}(S)$, for *any* fixed heights $h(p)$. For a simpler proof, see Powar [594]. See also Musin [558] for various functionals on triangular surfaces which are optimized by $\mathrm{DT}(S)$.

Let us mention that, in addition, $\mathrm{DT}(S)$ provides a means for *smoothing* the corresponding triangular surface. As was shown in Sibson [653], each point x within the convex hull of S can be expressed as the *weighted centroid* of its Delaunay neighbors p in $\mathrm{DT}(S \cup \{x\})$. Weights $w_p(x)$ can be computed from area properties of the corresponding diagram regions, and as functions of x, are continuously differentiable; see also Farin [336]. The corresponding interpolant to the spatial points $(p, h(p))$ is called the *natural neighbor interpolant*, and is given by

$$\phi(x) = \sum_{p \in S} w_p(x) h(p).$$

This useful property of $\mathrm{DT}(S)$ is shown to generalize to *regular triangulations* (duals of power diagrams for S, see Section 6.2), and to *higher-order Voronoi diagrams* (Section 6.5) in Aurenhammer [91]. For more material on natural neighbor interpolants and their applications, see, e.g., Boissonnat and Cazals [140].

On the negative side, $\mathrm{DT}(S)$ in general fails to fulfill optimization criteria similar to those mentioned above, such as minimizing the maximum angle, or minimizing the longest edge. Edelsbrunner *et al.* [315, 317] give near-quadratic time algorithms for computing triangulations optimal in that sense. $\mathrm{DT}(S)$ is not even *locally minimal*, in the sense that it does not always include the shorter diagonal for each of its convex quadrilaterals.

Kirkpatrick [455] proved that $\mathrm{DT}(S)$ may differ arbitrarily strongly from a *minimum-weight triangulation*, which is defined to have minimum

total edge length. Computing a minimum-weight triangulation is an important and interesting problem, whose complexity remained unknown for a long time; see Garey and Johnson [352]. Recently, Mulzer and Rote [557] showed its NP-hardness. (For the class of so-called *NP-hard problems*, it is very unlikely that *polynomial-time algorithms* exist, i.e., algorithms with a running time of $O(n^k)$, with k being a constant.) Subsets of edges of $\mathrm{DT}(S)$ which always have to belong to a minimum-weight triangulation are exhibited in Subsection 8.2.3.

On the other hand, the widely used *greedy triangulation*, which is obtained by inserting non-crossing edges in increasing length order, can be constructed from $\mathrm{DT}(S)$ in $O(n)$ time, by a result in Levcopoulos and Krznaric [504].

Finally, let us mention that the Delaunay triangulation avoids an undesirable property that might be shared by other triangulations. Fix a point v in the plane, called the viewpoint. For two triangles Δ and Δ' in a given triangulation, write $\Delta < \Delta'$ if Δ fully or partially hides Δ' as seen from v. This defines a relation, called the *in-front/behind relation*, on the triangles. De Floriani *et al.* [245] observed that this relation is *acyclic* if the triangulation is Delaunay, no matter where the viewpoint is chosen. That is, the relation gives a *partial order*, also called a *shelling order* (with center v) of the triangles, see e.g. Brugesser and Mani [171] and Subsection 6.5.1.

An example of a triangulation which is cyclic in spite of being minimum-weight can be found in Aichholzer *et al.* [32].

Edelsbrunner [301] generalized the result in [245] to *regular simplicial complexes* in d dimensions, a class that includes higher-dimensional Delaunay triangulations as a special case; see Section 6.3. An application stems from a popular algorithm in computer graphics that eliminates hidden objects by first partially ordering the objects according to the in-front/behind relation and then displaying them from back to front, thereby overpainting invisible parts. In particular, this algorithm will work well for *α-shapes* (Subsection 8.2.2) and their weighted variants (Section 6.6). A rapid algorithm for displaying 3D Delaunay triangulations, or parts thereof, on a standard raster display is described in Karasick *et al.* [445]; cf. also Section 11.1 for *pixel Voronoi diagrams*.

For early structural discussions of planar triangulations, the reader is referred to the theses by Tan [675] and Lambert [481], respectively. A survey on optimization properties of triangulations is given in Aurenhammer and Xu [106].

Chapter 5

GENERALIZED SITES

In order to meet practical needs, the concept of Voronoi diagram has been modified and generalized in many ways, for example by changing the underlying space, the distance function used, or the shape of the sites. Chapters 5 to 7 give a systematic treatment of generalized Voronoi diagrams.

It is commonly agreed that most geometric scenarios can be modeled with sufficient accuracy by *polygonal objects*. Two typical and prominent examples are the description of the workspace of a robot moving in the plane, and the geometric information contained in a geographical map. In both applications, robot motion planning and geographical information systems, the availability of proximity information for the scenario is crucial. This is among the reasons why considerable attention has been paid to the study of Voronoi diagrams for polygonal objects.

Still, in some applications the scenario can be modeled more appropriately when *curved objects*, for instance, circular arcs are also allowed. Many Voronoi diagram algorithms working for line segments can be modified to work for curved objects as well.

5.1. Line segment Voronoi diagram

Let G be a *planar straight-line graph* on n points in the plane, that is, a set of non-crossing line segments spanned by these points. For instance, G might be a *tree* (a connected graph without edge cycles), or a collection of disjoint line segments or polygons (edge cycles), or a complete *triangulation* of the points (a maximal non-crossing set of edges). The number of segments of G is maximal, at most $3n-6$, in the last case. We will discuss several types of diagrams for planar straight-line graphs in the present and following sections.

The classical type is the (*closest site*) *Voronoi diagram*, $V(G)$, of G. It consists of all points in the plane which have more than one closest segment

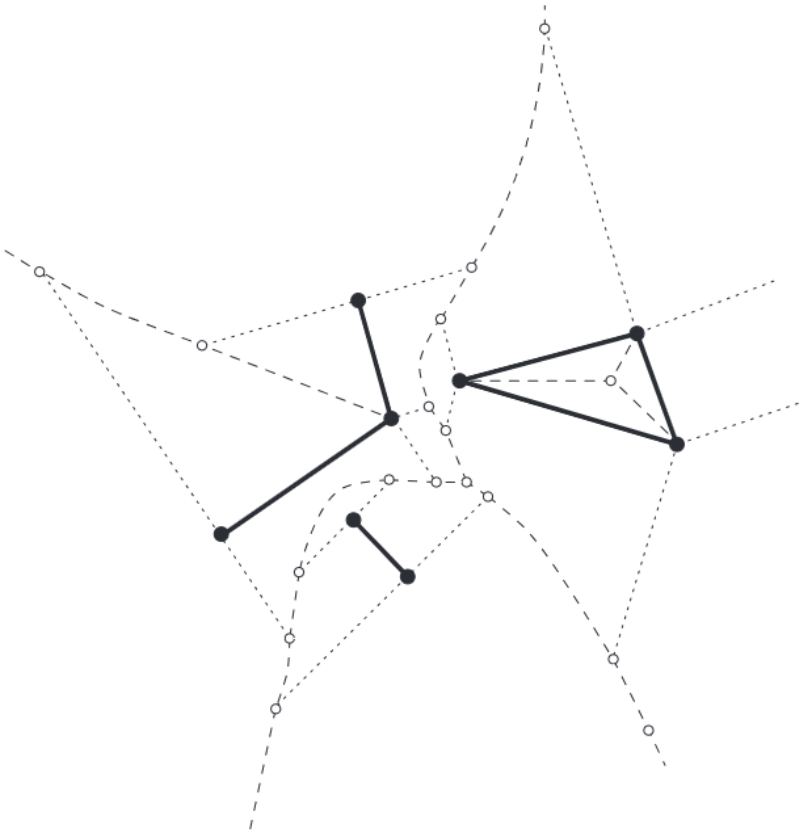


Figure 5.1. Line segment Voronoi diagram.

in G. $V(G)$ is known under different names in different areas, for example, as the *line Voronoi diagram* or *skeleton* of G, or as the *medial axis* or the *symmetric axis* when G is a simple polygon.

(A polygon is called *simple* if it is homeomorphic to a disk. Simple polygons have connected boundaries; in particular, they contain no 'holes'.)

Applications in such diverse areas as biology, geography, pattern recognition, computer graphics, and motion planning exist; see e.g. Kirkpatrick [454] and Lee [494] for early references, and the recent book by Siddiqi and Pizer [654].

See Figure 5.1. $V(G)$ is formed by straight-line edges and parabolically curved edges, both shown as dashed lines. Straight-line edges are part of either the perpendicular bisector of two segment endpoints, or of the angular bisector of two segments. Curved edges consist of points equidistant from a segment endpoint and a segment's interior. There are two types of vertices, namely of *type* 2 having degree two, and of *type* 3 having degree three (provided that the point set which spans G is in *general position*). Both are equidistant from a triple of objects (segment or segment endpoint), but for type-2 vertices the triple contains a segment along with one of its endpoints.

Together with G's segments, the edges of $V(G)$ partition the plane into regions. These can be refined by introducing certain normals through segment endpoints (shown dotted in Figure 5.1), in order to delineate *faces* each of which is closest to a particular segment or segment endpoint. Two such normals start at each segment endpoint where G forms a reflex angle, and also at each *terminal* of G which is an endpoint belonging to only one segment in G. A normal either ends at a type-2 vertex of $V(G)$ or extends to infinity.

Observe that each face f is *visible* from its defining component c in G; that is, for each point $x \in f$ there is a point $y \in c$ such that their straight-line connection $\overline{xy}$ stays entirely within f. (If c is a segment endpoint then $y = c$, and if c is a segment then take for y the point on c closest to x.) As a consequence, the union of the three faces for a segment of G plus its two endpoints is a *simply connected* set, i.e., it is connected and contains no holes.

It is well known that the number of faces, edges, and vertices of $V(G)$ is linear in n, the number of segment endpoints for G. The number of vertices is shown to be at most $4n - 3$ in Lee and Drysdale [496]. An exact bound, which also counts the 'infinite' vertices at unbounded edges and segment normals, is given below.

Lemma 5.1. *Let G be a planar straight-line graph on n points in the plane, and let G realize t terminals and r reflex angles. The number of (finite and infinite) vertices of $V(G)$ is exactly $2n + t + r - 2$.*

Proof. Suppose first that G consists of e disjoint segments (that do not touch at their endpoints). Then there are e regions, and each type-3 vertex belongs to three of them. By *Euler's formula* for planar graphs, there are exactly $2e - 2$ such vertices, if we also count those at infinity. To count the number of type-2 vertices, observe that each segment endpoint is a terminal and gives rise to two segment normals each of which, in turn, yields one (finite or infinite) vertex of type 2. Hence there are $4e$ such vertices, and $6e - 2$ vertices in total.

Now let G be a general planar straight-line graph with e segments. We simulate G by disjoint segments, by shortening each segment slightly such that the segment endpoints are in *general position*. Then we subtract from $6e - 2$ the number of vertices which have been generated by this simulation.

Consider an endpoint p that is incident to $d \geq 2$ segments of G. Obviously, p gives rise to d copies in the simulation.

The Voronoi diagram of these copies has $d - 2$ finite vertices, which are new vertices of type 3. As the sum of the degrees $d \geq 2$ in G is $2e - t$, we get $2e - t - 2(n - t)$ new vertices in this way.

Each convex angle at p gives rise to two new normals emanating at the respective copies of p, and thus to two (finite) type-2 vertices. A possible reflex angle at p gives rise to one (finite or infinite) type-3 vertex, on the perpendicular bisector of the corresponding copies of p. There are r reflex angles in G, and thus $2e - t - r$ convex angles. This gives $r + 2(2e - t - r)$ new vertices in addition.

The lemma follows by simple arithmetic. □

Surprisingly, the number of *edges* of G does not influence the bound in Lemma 5.1. The maximum number of vertices, $3n - 2$, is achieved, for example, if G is a set of disjoint segments ($t = n$ and $r = 0$), or if G is a *simple polygon* P ($t = 0$ and $r = n$). In the latter case, the majority of applications concerns the part of $V(P)$ interior to P. This part is commonly called the *medial axis* of P. The medial axis of an n-gon with r reflex interior angles has a *tree-like* structure and realizes exactly $n + r - 2$ vertices and $2(n + r) - 3$ edges, a bound first mentioned in Lee [494].

Several algorithms for computing $V(G)$, for general or restricted planar straight-line graphs G, have been proposed and tested for practical efficiency. $V(G)$ can be computed in $O(n \log n)$ time and $O(n)$ space by *divide & conquer algorithms* (Kirkpatrick [454], Lee [494], and Yap [712]), by using the *plane sweep technique* (Fortune [344]), and by *randomized incremental insertion* (Boissonnat *et al.* [142] and Klein *et al.* [466]); cf. Chapter 3 on basic algorithms.

Burnikel *et al.* [173] give an early overview of existing methods for computing $V(G)$. They also discuss implementation details of an algorithm in Sugihara *et al.* [669] that first inserts all segment endpoints, and then all the segments, of G in random order. An enhanced fast implementation of this algorithm is given in Held [399]. A method of comparable simplicity and practical efficiency (though with a worst-case running time of $\Theta(n^2)$) is proposed in Gold *et al.* [366, 367]. They first construct a Voronoi diagram for point sites by selecting one endpoint for each segment, and then maintain the diagram while expanding the endpoints, one by one, to their corresponding segments. During an expansion, the resulting topological updates in the diagram can be carried out efficiently.

In fact, *Voronoi diagrams for moving sites*, which are sometimes also called *kinetic Voronoi diagrams* in the literature, are well-studied concepts; they will be discussed in Section 9.1.

Recently, a practical and efficient divide & conquer algorithm, which works for general curved sites, and constructs the diagram $V(G)$ as a special straight-line case, has been devised in Aichholzer *et al.* [25]. We will detail this method in Section 5.5.

If G is a connected graph then $V(G)$ can be computed in *randomized* time $O(n \log^* n)$; see Devillers [261]. (Here $\log^* n$ denotes the *iterated logarithm*, that is, the number of times the (dual) logarithm has to be applied to n in order to obtain a number smaller than 1.) Moreover, $O(n)$ time randomized, and deterministic, algorithms for the medial axis of a simple polygon have been designed by Klein and Lingas [465] and Chin *et al.* [218], settling open questions of long standing. The case of a convex polygon is considerably easier; see Section 5.2.

An efficient $O(n \log^2 n)$ work *parallel algorithm* for computing $V(G)$ is given in Goodrich *et al.* [373]. This is improved to $O(\log n)$ parallel (randomized) time using $O(n)$ processors in Rajesekaran and Ramaswami [599]. The latter result also implies an optimal parallel construction method for the Voronoi diagram for point sites.

The line segment Voronoi diagram $V(G)$ is a *planar graph* embedded in the plane, and as such has a well-defined combinatorial *dual graph*. However, this dual reflects the adjacencies between the regions of $V(G)$ in a way not always desirable in applications. More specifically, the dual is a 'triangulation' of the segments in G that may contain multiple edges, due to multiple adjacencies between Voronoi regions. This shortcoming is remedied in Chew and Kedem [216], who define a so-called *segment Delaunay triangulation* $\mathrm{DT}(G)$ for G, by placing certain additional points on G's segments. Based on this idea, Brévilliers *et al.* [165] define general segment triangulations, and show that *local optimality* that characterizes the classical Delaunay triangulation (Section 4.2) extends to these structures. Moreover, they prove in [166] that $\mathrm{DT}(G)$, and thus, also the primal structure, $V(G)$, can be constructed by improving flips in segment triangulations.

Along with the study of closest-site Voronoi diagrams go their *farthest-site* counterparts. In that model, each site s gets allotted the region of all points for which s is the farthest site (rather than the closest) in the input set. See Section 6.5 for the case of point sites, where the diagram has a simple *tree structure* — just the sites on the *convex hull* of the input point set contribute nonempty regions.

For *line segments* as sites, their farthest-site Voronoi diagram is investigated in Aurenhammer *et al.* [95]. The properties of this diagram deviate from the obvious. For example, regions may be disconnected, and region emptiness cannot be characterized by convex hull properties; see Figure 5.2. Moreover, and unlike the closest line segment case, the number of edges and vertices of the diagram remains $O(n)$ in the worst case when the segments in the input set G do *not* form a planar graph (i.e., define crossings).

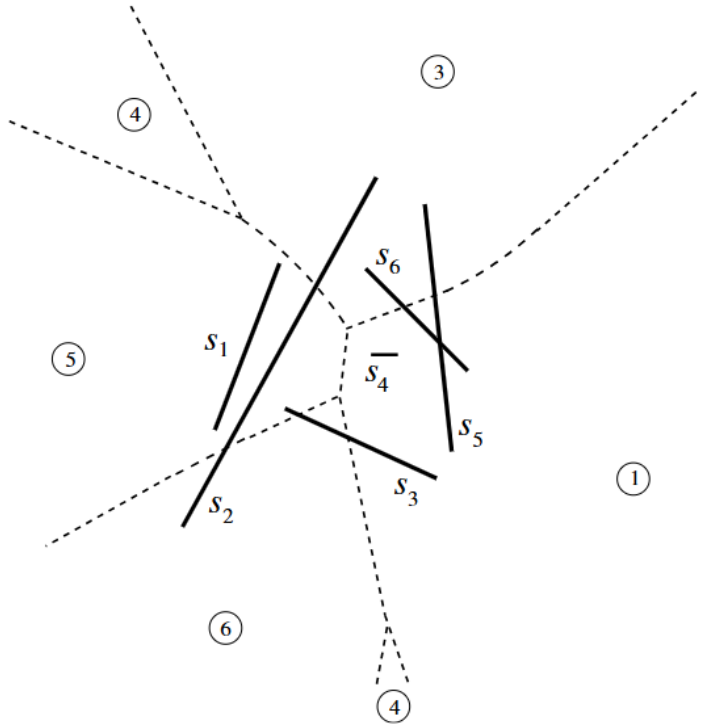


Figure 5.2. Farthest-segment Voronoi diagram for six sites. Encirculated numbers indicate affiliation of regions to segments (from [95]).

The diagram can be computed in time $O(n \log n)$, by applying a generalized (and primal) version of the ear-clipping algorithm for the farthest-site Delaunay triangulation in Subsection 6.5.1.

Basically, a convex projection surface in 3-space is constructed, by using, for each point x in the plane, its distance to the farthest segment in G as a third coordinate for x — an algorithmic idea that also works for the medial axis or the straight skeleton of a *convex* polygon; cf. Sections 5.2 and 5.3. For nonconvex simple polygons such an algorithm would however fail, due to the nonconvexity of the resulting projection surface.

The farthest-segment Voronoi diagram is still a tree; all its connected faces are unbounded. The upper bound of $8n + 4$ in [95] on their number has been improved recently to $6n - 6$ in Papadopoulou and Dey [580].

The farthest-site Voronoi diagram for *simple polygons* as sites is studied in Cheong *et al.* [208]. Its size is still $O(n)$, though its structure is more complicated, leading to an $O(n \log^3 n)$ time algorithm.

Line segment Voronoi diagrams of *order* k (cf. Section 6.5) are another natural generalization, covering the closest line segment Voronoi diagram ($k = 1$) and the farthest-segment Voronoi diagram ($k = |G| - 1$). Surprisingly, they did not receive much attention for general k until

recently, when Papadopoulou and Zavershynskyi [584] showed that their combinatorial complexity remains in $O(k(n-k))$, as in the point site case.

In dimensions more than two, the known results on Voronoi diagrams for generalized sites are sparse. Exceptions are power diagrams (Section 6.2), Voronoi diagrams for spheres, and the medial axis in three dimensions (both Section 6.6).

5.2. Convex polygons

Voronoi diagrams for a single *convex polygon* have a particularly simple structure. Tailor-made algorithms for their construction have been designed.

Let C be a convex n-gon in the plane. The part of the Voronoi diagram of C's sides which lies inside C is called the *medial axis*, $M(C)$, of C. It is a *tree* whose edges, by convexity of C, are pieces of *angular bisectors* of the sides. In fact, $M(C)$ coincides with the *straight skeleton* $\mathrm{SK}(C)$ of C (discussed in Section 5.3) in this case. See Figure 5.3 for an illustration. $M(C)$ realizes exactly n faces, $n-2$ vertices, and $2n-3$ edges.

There is a simple *randomized incremental algorithm* by Chew [210] that computes $M(C)$ in $O(n)$ time. The algorithm first removes, in random order, the halfplanes whose intersection is C. Removing a halfplane means removing the corresponding side e of C, and extending the two sides e' and e'' adjacent to e so that they become adjacent in the new convex polygon. This can be done in constant time per side. (If the extensions of e' and e'' do not intersect then the obtained polygon is unbounded but still convex; we omit the easy modifications necessary for that case.)

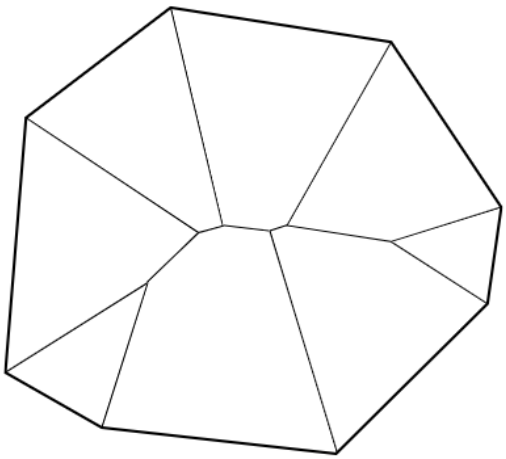


Figure 5.3. Medial axis of a convex polygon.

The *adjacency history* of C is stored. That is, for each removed side e, one of its formerly adjacent sides is recorded. In a second stage, the sides are put back in reverse (still randomized) order, and the medial axis is maintained during these insertions.

Let us focus on the insertion of the ith side, e_i. We have to integrate the face, $f(e_i)$, of e_i into the medial axis of the $i-1$ sides that have been inserted before e_i. From the adjacency history, we already know a side e' of the current polygon that will be adjacent to e_i after its insertion. Hence we know that the angular bisector of e_i and e' will contribute an edge to $f(e_i)$.

Having a starting edge available in $O(1)$ time, the face $f(e_i)$ now can be constructed in time proportional to its size. We construct $f(e_i)$ edge by edge, by simply tracing and deleting parts of the old medial axis interior to $f(e_i)$. As the medial axis of an i-gon has $2i-3$ edges, and each edge belongs to two faces, the expected number of edges of a randomly chosen face is less than 4. Thus $f(e_i)$ can be constructed in constant expected time, which gives an $O(n)$ randomized algorithm for computing $M(C)$.

The same technique also applies to the *Voronoi diagram* for the vertices *of a convex n-gon* C, that is, to the Voronoi diagram of a set S of n point sites in *convex position*. By Lemma 2.2, all regions in $V(S)$ are unbounded, such that the edges of $V(S)$ have to form a *tree*. Hence $V(S)$ has the same number of edges and (finite) vertices as the medial axis of C.

For each $p \in S$, its region $\mathrm{VR}(p,S)$ shares an unbounded edge with the regions $\mathrm{VR}(p',S)$ and $\mathrm{VR}(p'',S)$, respectively, where p' and p'' are the neighbors of p on the boundary of the convex hull of S (which is the polygon C). An adjacency history can be computed in $O(n)$ time, by removing the sites in random order and maintaining their convex hull. For each site that is re-inserted, the expected number of edges is less than 4, as before. So an $O(n)$ randomized construction algorithm is obtained.

The diagrams $V(S)$ and $M(C)$ can also be computed in *deterministic* linear time; see Aggarwal *et al.* [17]. The details of this algorithm are much more involved, however.

5.3. Straight skeletons

In comparison to the Voronoi diagram for point sites, which is composed of straight edges, the occurrence of curved edges in the line segment Voronoi diagram $V(G)$ is a disadvantage in the computer representation and construction, and sometimes also in the application, of $V(G)$.

There have been several attempts to *linearize* and simplify $V(G)$, mainly for the sake of efficient point location and motion planning; see

Canny and Donald [181], Kao and Mount [439], de Berg *et al.* [242], and McAllister *et al.* [527]. The *compact Voronoi diagram* in [527] is particularly suited to these applications. It is defined for the case where G is a set of k disjoint convex polygons. Its size is only $O(k)$, rather than $O(n)$, and it can be computed in time $O(k \log n)$; see Subsection 8.1.1 for more details.

As a different alternative to $V(G)$, we now describe the *straight skeleton*, $\mathrm{SK}(G)$, of a planar straight-line graph G. This structure is introduced, and discussed in more detail, in Aichholzer and Aurenhammer [29]. $\mathrm{SK}(G)$ is composed of *angular bisectors* and thus does not contain curved edges. In general, its size is even less than that of $V(G)$.

Its use as a type of *skeleton* for G partially stems from the fact that G can be reconstructed from $\mathrm{SK}(G)$ in a simple manner. This fact is considered important, for example, in picture processing; see Pfaltz and Rosenfeld [590], who observe this property for the medial axis of a simple polygon. The recently introduced *triangulation axis* of a polygon P also shares this feature. This skeletal axis is piecewise linear, too, and is defined via some optimal triangulation of P. Its size is (sometimes drastically) smaller than the size of the medial axis or the straight skeleton of P; see Aigner *et al.* [44].

The relevance of the straight skeleton $\mathrm{SK}(G)$ in shape and object recognition is discussed, e.g., in Tanase and Veltkamp [677] and Demuth *et al* . [256]. Applications in robotics and to origami design have been described in Barequet *et al.* [113], and in Lang [483] and Demaine *et al.* [255], respectively. A lifted 3D model of $\mathrm{SK}(G)$ turns out to be of use in architecture and geographic information systems, as will be sketched later. (Interestingly, and seemingly having faded into oblivion, utilizing straight skeletons as a means for designing roof constructions dates back more than a hundred years, as is documented in a monograph for engineers by Peschka [589], and later in a textbook on descriptive geometry by Müller [553].) Straight skeletons also appear implicitly in a certain Voronoi diagram for time distances, the so-called *city Voronoi diagram* for point sites under a rectilinear transportation network; see Aichholzer *et al.* [38]. We will give details on the city Voronoi diagram in Section 7.6.

$\mathrm{SK}(G)$ is defined as the *interference pattern* of certain *wavefronts* propagated from the segments and segment endpoints of G. Let F be a connected component (called a *figure*) of G. Imagine F as being surrounded by a belt of (infinitesimally small) width ε. For example, a single segment s gives rise to a rectangle of length $|s| + 2\varepsilon$ and width 2ε, and a simple polygon P gives rise to two offset copies of P with minimum distance 2ε. In general, if F partitions the plane into c connected domains then F gives rise to c simple polygons, called *wavefronts* for F.

The wavefronts arising from all the figures of G are now propagated simultaneously, at the same speed, and in a self-parallel manner. Wavefront vertices move on angular bisectors of wavefront edges which, in turn, may increase or decrease in length during the propagation. This situation continues as long as the wavefronts do not change combinatorially. Basically, there are two types of changes, called events.

(1) *Edge event*: A wavefront edge collapses to length zero. (The wavefront may vanish altogether, due to three simultaneous edge events.)
(2) *Split event*: A wavefront edge splits due to interference or self-interference. In the former case, two wavefronts merge into one, whereas in the latter case a wavefront splits into two.

After either type of event, we are left with a new set of wavefronts which are propagated recursively.

The *edges* of $\mathrm{SK}(G)$ now are the pieces of angular bisectors traced out by wavefront vertices. Each *vertex* of $\mathrm{SK}(G)$ corresponds to some point where an edge event or a split event takes place. $\mathrm{SK}(G)$ is a unique structure defining a polygonal partition of the plane; see Figure 5.4.

During the propagation, each wavefront edge e sweeps across a certain area which we call the *face* of e. Each segment of G gives rise to two wavefront edges and thus to two faces, one on each side of the segment.

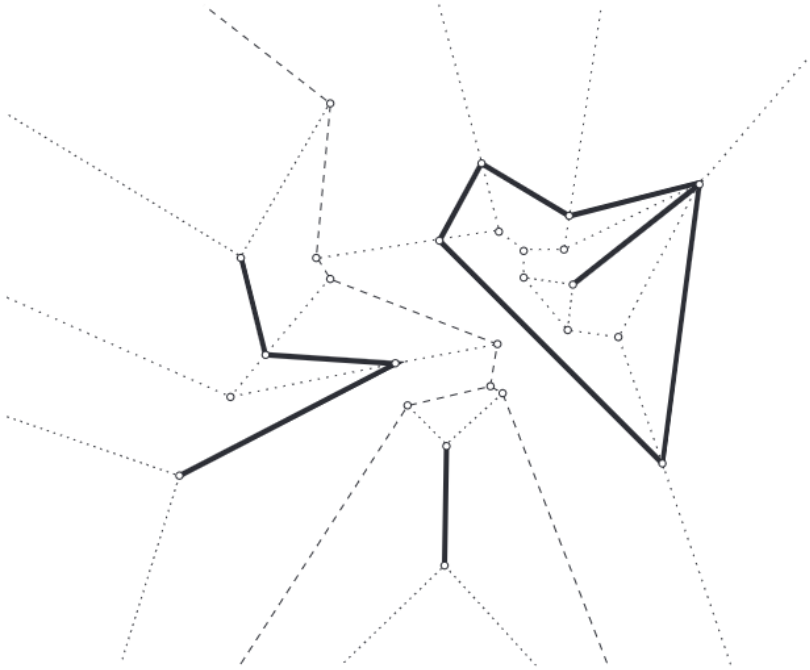


Figure 5.4. Straight skeleton for three figures.

Each terminal of G (endpoint of degree one) gives rise to one face. Faces can be shown to be *monotone polygons*, i.e., each intersection of a face with lines having some fixed direction is connected. In our case, this direction is normal to the generating wavefront edge. As a consequence, faces are *simply connected* sets. This implies a total of $2m + t = O(n)$ faces, if G realizes m edges and t terminals. There is also an exact bound on the number of vertices of $\mathrm{SK}(G)$.

Lemma 5.2. *Let G be a planar straight-line graph on n points, and t of which are terminals. The number of (finite and infinite) vertices of* $\mathrm{SK}(G)$ *is exactly* $2n + t - 2$.

From Lemma 5.1 it is apparent that $\mathrm{SK}(G)$ has r vertices less than $V(G)$, if G defines r reflex angles. In particular, if G is a *simple polygon* with r reflex interior angles, then the part of $\mathrm{SK}(G)$ interior to G is a *tree* with only $n - 2$ vertices, whereas the medial axis of G has $n + r - 2$ vertices.

By construction, $\mathrm{SK}(G)$ encodes all self-parallel inward and outward offsets for G, also called *mitered offsets* in the polygon case; see e.g. Devadoss and O'Rourke [260]. However, this offsetting process is different from the classical offsetting process, which is based on a wavefront model (or *growth model*) for the Voronoi diagram or the medial axis $V(G)$ of G; cf. the end of Section 5.5, and the 'expanding waves' view in Chapter 2. The propagation speed of all points on the wavefront is the same in that model, whereas, in the model for $\mathrm{SK}(G)$, the speed of each wavefront vertex, u, is controlled by the interior angle, $\beta(u)$, between its incident wavefront edges. The sharper the angle, the faster is the movement of the vertex. More precisely, the speed $s(u)$ of vertex u is given by the formula

$$s(u) = 1/\sin\frac{\beta(u)}{2}$$

and tends to infinity if the angle $\beta(u)$ approaches 0. This behavior makes $\mathrm{SK}(G)$ completely different from the Voronoi diagram of G. It can be shown that, without prior knowledge of its structure, $\mathrm{SK}(G)$ *cannot* be defined by means of distances from G, by lack of its *non-procedural definition*; see [28]. Also, $\mathrm{SK}(G)$ does not fit into the framework of *abstract Voronoi diagrams* described in Section 7.5: The bisector of two segments of G would be the interference pattern of the rectangular wavefronts they send out, but these curves do not fulfill condition (ii) in Definition 7.7. As a consequence, the well-developed machinery for constructing Voronoi diagrams (see Chapter 3) does not apply to $\mathrm{SK}(G)$.

An algorithm that simulates the wavefront propagation is given in [29]. It maintains a triangulation, T, of the wavefront vertices in 'free space' —

the part of the plane not yet swept over by any wavefront. Collision of offsetting figures can be predicted by the collapse of triangles in T. Though the known upper bounds are much higher, T typically undergoes only $O(n)$ updates (of several kinds, including collapses). A runtime of $O(n \log n)$ is then achieved, so the method is simple and practically efficient. We conjecture that the techniques for kinetic *collision detection* in Kirkpatrick *et al.* [459], which are based on a so-called *pseudo-triangulation* (Section 6.3) of the free space, are capable of theoretically speeding up this approach.

More sophisticated though more involved construction methods running in subquadratic time are given in Eppstein and Erickson [326], Cheng and Vigneron [205], and recently in Vigneron and Yan [691] who achieve a running time of $O(n^{4/3+\varepsilon})$. These works also describe additional structural properties of the straight skeleton. The so-called *motorcycle graph* is introduced and utilized, which captures an important aspect of straight skeletons. This planar graph resolves the interplay of the reflex vertices' trajectories, and thus is of substantial help in predicting the (by nature non-local) split events. The paper by Vyatkina [695] shows that $\mathrm{SK}(G)$ can be decomposed into (pruned) medial axes of certain convex polygons. Recently, practical implementations of straight skeleton algorithms have been developed in Huber and Held [410] and in Palfrader *et al.* [577].

Whether $\mathrm{SK}(G)$ can be computed in $O(n \log n)$ time, and possibly even faster inside a simple polygon, is still an open problem. In fact, only the trivial lower bound, $\Omega(n)$, is known in the polygon case.

In view of the notoriously difficult task of computing the straight skeleton $\mathrm{SK}(G)$ efficiently, simpler instances for G have been studied as well. Clearly, if G is a *convex polygon*, then $\mathrm{SK}(G)$ is just the *medial axis* of G, which can be constructed in $O(n)$ time (Section 5.2).

More interesting is the case of a *monotone polygon*, which is solvable in $O(n \log n)$ time; see Das *et al.* [239]. Basically, the polygon's boundary is decomposed into two monotone chains of edges. The straight skeleton of each chain can be constructed without the trouble of split events, and the two partial skeletons merged in a simple way, similar as in the Voronoi diagram case; see Section 3.3.

$\mathrm{SK}(G)$ has a three-dimensional interpretation, obtained by defining the height of a point x in the plane as the unique time when x is reached by a wavefront. In this way, $\mathrm{SK}(G)$ lifts up to a *polygonal surface* Σ_G, where points on G have height zero. In a geographical application (e.g., *terrain modeling*), G may delineate rivers, lakes, and coasts, and Σ_G represents a corresponding terrain with fixed slope; see Figures 5.5 and 5.6 (taken from [29]). Σ_G has the nice property that every raindrop that hits a terrain facet f drains off to the segment or terminal of G defining f; see [28]. This

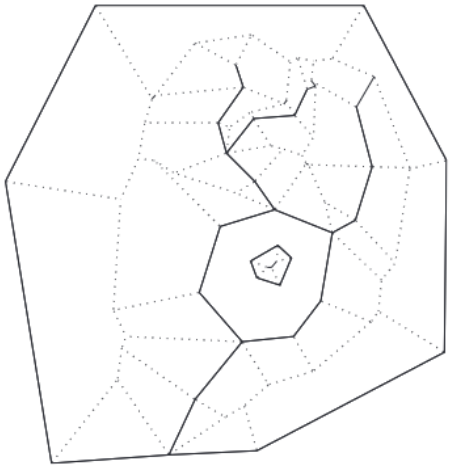


Figure 5.5. Coastline and rivers.

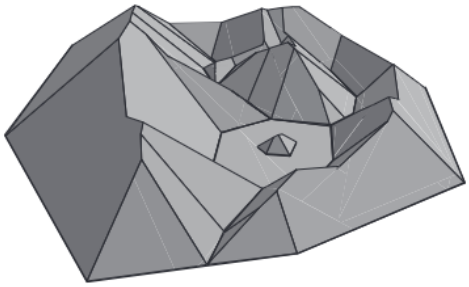


Figure 5.6. Reconstructed terrain.

may have applications in the study of rain water fall and its impact on the flooding caused by rivers in a given geographic area, where currently the Voronoi diagram of G is used; see, for example, Barett [116].

When restricted to the interior of a simple polygon P, the surface Σ_P can be used as a canonical construction of a *roof* of given slope above P. For *rectilinear* (and axis-aligned) *polygons* P, the medial axis of P in the L_∞*-metric* will do the job. $\mathrm{SK}(P)$ coincides with this medial axis for such polygons, and thus generalizes this roof construction technique to general shapes of P. Applications arise, e.g., in the automated generation of building models; see Brenner [164], Kelly and Wonka [451], and references therein.

Sometimes, the volatile behavior of the straight skeleton $\mathrm{SK}(P)$, which stems from the high propagation speed of sharp reflex vertices of the polygon P, is considered a disadvantage. A possible way out is to 'cut off' such vertices in a controlled way, an idea sketched in [29] and applied systematically in Tanase and Veltkamp [676]. The resulting (appropriately pruned) straight skeleton is called the *linear axis* of the polygon P in the latter paper, and is shown to approximate the medial axis of P in a certain sense. See also Bookstein [153] for an early study of the straight skeleton for polygons with large interior angles, who termed it the *line skeleton*.

The concept of straight skeleton $\mathrm{SK}(G)$ of a planar straight line graph G can be modified by tuning the propagation speed of the individual wavefront edges. That is, edges can be given a weight that expresses their velocity. For the 3D model of $\mathrm{SK}(G)$ described above, this means putting prescribed facet slopes for the corresponding surface. The maximal size of the skeleton, and its procedural definition, remain unaffected. (The exceptional case where

two parallel edges of different weights become adjacent after an edge event can be treated consistently, by considering the created wavefront vertex as a convex vertex.) The *monotonicity* of the skeleton faces, however, and if G is disconnected even their simple connectedness, may be lost.

In the particularly interesting case where G is the boundary of a *convex polygon* P, this so-called *weighted skeleton*, $\mathrm{SK}_W(P)$, is studied in Aurenhammer [94]. By adjusting the edge weights accordingly, $\mathrm{SK}_W(P)$ is capable of partitioning a given n-gon P into n convex parts, each part being based on a single edge of P and covering a predefined 'share' of P. The share may relate, for example, to the spanned area, to the number of contained points from a given point set, or to the total edge length covered in a given set of curves. (The partitioning results for power diagrams presented in Section 6.4 are related but not equivalent.) Possible applications of such *fixed-share decompositions* include priority-based or fair facility allocation, which may concern real estate or access to power lines, aquifers, or oil wells.

Area decomposition may be done recursively, yielding a new instance of a structure called *Voronoi tree map*, of use for visualizing attributed hierarchical data; see Andrews *et al.* [62], Granitzer *et al.* [377], and Balzer and Deussen [111].

Straight skeletons can be generalized to higher dimensions. For example, in 3D the input could be a nonconvex and boundary-connected polyhedron P with n vertices. (For convex polyhedra, the straight skeleton coincides with the medial axis.) The shrinking process now involves offsetting inwards the boundary facets of P in a self-parallel way and at unit speed. Thereby, facets change their shape and may disconnect (as do polygon edges in the 2D case), and the shrinking process continues with each part. The volumes traced out by the facets are polyhedra again. Hence, the structure retains its piecewise-linear shape and offers an alternative to the medial axis of P whose geometric description is rather complex; see Section 6.6.

However, and unlike the planar case, there is no unique (mitered) offsetting process for P, in general. This was observed in Demaine *et al.* [254] and Barequet *et al.* [115]. In fact, as there possibly exist several different 'skeleton-like' partitions of a nonconvex polygon by means of angular bisectors [28], even such having super-linear size, there may be different ways of locally offsetting a vertex v of P of high degree. In certain cases, the planar weighted straight skeleton can be used to compute a canonical offset structure [115]. In the general case (when the edges incident to v positively span 3-space), a generalization of the straight skeleton to the sphere will work; see Aurenhammer and Walzl [105].

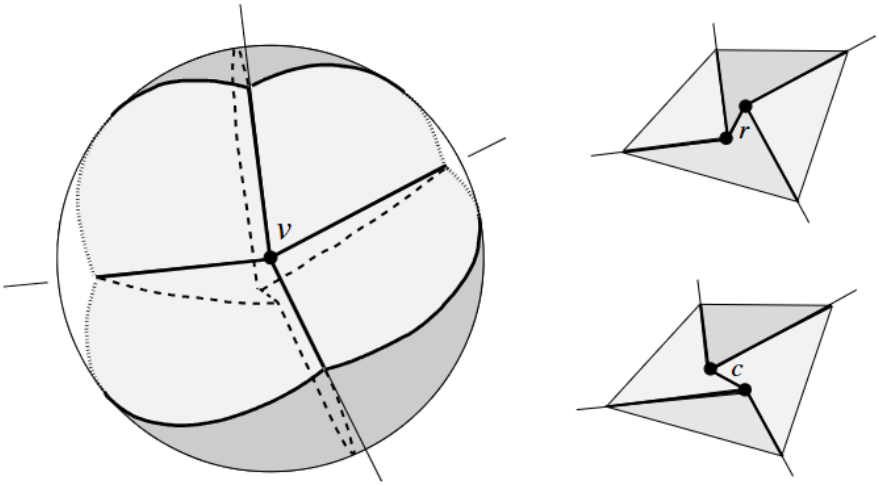


Figure 5.7. Two local offset surfaces for a saddle point (right). The spherical skeleton (left, dashed style) produces the lower solution. The short skeleton arc corresponds to a convex edge c in the offset polytope. It connects the two vertices which the saddle point is split into. In the upper solution, the offset edge r is reflex.

Figure 5.7 shows two possible offsets for a saddle point v of degree 4. Either choice leads to a different though valid straight skeleton for the underlying polyhedron P.

Concerning size, $O(n^2\alpha(n)^2)$ is a lower bound on the combinatorial complexity of a 3D skeleton [115]. (Here, $\alpha(n)$ denotes the inverse of the rapidly growing *Ackermann function*; see e.g. [642]. We have $\alpha(n) \leq 5$ even for 'astronomically large' arguments n, such that $\alpha(n)$ can be considered a constant for all practical purposes.) A trivial upper bound is $O(n^4)$, because each 4-tuple of facet planes for P can yield only one common point of intersection during the entire offsetting process.

If P is an *orthogonal polyhedron* (all edges of P are parallel to the coordinate axes) then the skeleton size reduces to $O(n^2)$, and an $O(n^2 \log n)$ construction algorithm exists [115]. A recent implementation using the plane sweep technique is given in Martinez *et al.* [516]. The straight skeleton is the L_∞*-medial axis* of P in this case, which also enables its efficient computation in *voxel representation*. Yet, by its non-interpretability in terms of distances in general [28], one cannot resort to powerful distance transforms (like the EDT for the medial axis; see Section 11.1), which makes the accurate computation of *pixel straight skeletons* an interesting task already in 2D; see Demuth *et al.* [256].

An efficient algorithm for the straight skeleton of general nonconvex 3D polyhedra (in either representation) is still missing, as are non-trivial upper bounds on its combinatorial complexity.

5.4. Constrained Delaunay and relatives

In certain situations, unconstrained proximity among a set of sites is not enough information to meet practical needs. There might be reasons for not considering two sites as neighbors although they are geometrically close to each other. For example, two cities that are geographically close but separated by high mountains might be far from each other from the point of view of a truck driver. The concepts described below have been designed to model constrained proximity among a set of sites.

Let S be a set of n point sites in the plane, and let L be a set of non-crossing line segments spanned by S. Note that $|L| \leq 3n-6$. The segments in L are viewed as obstacles: We define the *bounded distance* between two points x and y in the plane as

$$b(x,y) = \begin{cases} d(x,y) & \text{if } \overline{xy} \cap L = \emptyset, \\ \infty & \text{otherwise,} \end{cases}$$

where d stands for the Euclidean distance. In the resulting *bounded Voronoi diagram* $V(S,L)$, regions of sites that are close but not visible from each other are clipped by segments in L. Regions of sites being segment endpoints are *nonconvex* near the corresponding segment; see Figure 5.8 (left).

The dual of $V(S,L)$ is not a full triangulation of S, even if the segments in L are included. However, $V(S,L)$ can be modified to dualize into a triangulation which includes L and, under this restriction, is as much 'Delaunay' as possible.

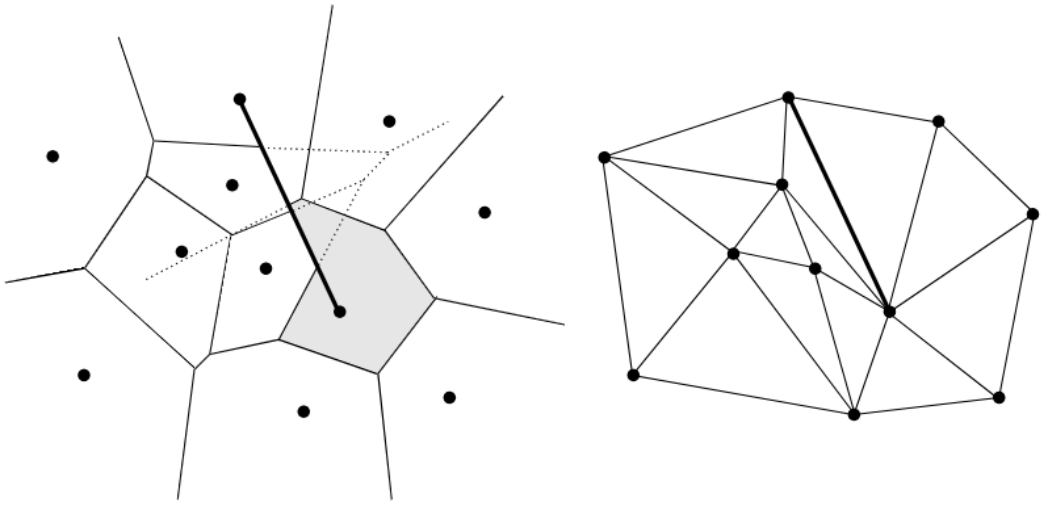


Figure 5.8. Bounded Voronoi diagram extended, and its dual, the constrained Delaunay triangulation. The single constraining segment is drawn in bold style.

The modification is simple but nice. For each segment $\ell \in L$, the regions clipped by ℓ from the right are extended to the left of ℓ, as if only the sites of these regions were present. The regions clipped by ℓ from the left are extended similarly. See Figure 5.8 again, where these modifications are shown in dotted lines. Of course, extended regions may overlap now, so they fail to define a partition of the plane. If we dualize now by connecting sites of regions that share an edge, a full triangulation that includes L is obtained: the *constrained Delaunay triangulation*, $\mathrm{CDT}(S, L)$. It is clear that the number of edges of $\mathrm{CDT}(S, L)$ is at most $3n - 6$, and that, in general, the number of edges of $V(S, L)$ is even less. Hence both structures have a linear size.

The original definition of $\mathrm{CDT}(S, L)$ in Lee and Lin [497] is based on a modification of the *empty-circle property*: $\mathrm{CDT}(S, L)$ contains L and, in addition, all edges between sites $p, q \in S$ that have $b(p, q) < \infty$ and that lie on some circle enclosing only sites $r \in S$ with at least one of $b(r, p), b(r, q) = \infty$.

Algorithms for computing $V(S, L)$ and $\mathrm{CDT}(S, L)$ have been proposed in Lee and Lin [497], Chew [211], Wang and Schubert [697], Wang [696], Seidel [629], and Kao and Mount [440]. The last two methods seem best suited to implementation. For an application of $\mathrm{CDT}(S, L)$ to quality *mesh generation* see Chew [213] and Shewchuk [647]; the latter paper carefully discusses the implementation details.

We sketch the $O(n \log n)$ time *plane sweep* approach in [629]. If only $V(S, L)$ is required then the plane sweep algorithm described in Section 3.4 can be applied without much modification. If $\mathrm{CDT}(S, L)$ is desired then the extensions of $V(S, L)$ as described above are computed in addition. To this end, an additional sweep is carried out for each segment $\ell \in L$. The sweep starts from the line through ℓ in both directions. It constructs, on the left side of this line, the (usual) Voronoi diagram of the sites whose regions in $V(S, L)$ are clipped by ℓ from the right, and vice versa.

The special case where S and L are the sets of vertices, and sides, of a *simple polygon* has received special attention, mainly because of its applications to visibility problems in polygons. The bounded Voronoi diagram $V(S, L)$ is constructible in $O(n)$ randomized time in this case; see Klein and Lingas [463]. If the L_1*-metric* instead of the Euclidean metric is used to measure distances, the same authors [465] give a deterministic linear time algorithm. Both algorithms, as well as the linear time medial axis algorithms in [464] and in [218], first decompose the polygon into smaller parts called *histograms*. These are polygons whose vertices, when considered in cyclic order, appear sorted in some direction.

An alternative concept that forces a set L of line segments spanned by S into $\mathrm{DT}(S)$ is the *conforming Delaunay triangulation*. For each segment $\ell \in L$ that does not appear in $\mathrm{DT}(S)$, new sites on ℓ are added such that ℓ becomes expressible as the union of Delaunay edges in $\mathrm{DT}(S \cup C)$, where C is the total set of added sites. For several site adding algorithms, $|C|$ depends on the size as well as on the geometry of L. See, e.g., the survey article by Bern and Eppstein [126] and references therein. Edelsbrunner and Tan [316] show that $|C| = O(k^2 n)$ is always sufficient, and construct a set of sites with this size in time $O(k^2 n + n^2)$, for $k = |L|$.

Conforming triangulations should not be confused with the concept of *compatible triangulations*; see Aichholzer *et al.* [37]. These are triangulations T_1 and T_2 of two planar point sets S_1 and S_2, respectively, such that there is a triangle-preserving bijection $S_1 \to S_2$. Whether a compatible triangulation T_2 always exists, given S_1, S_2, and T_1 (under the obvious necessary size and convex hull restrictions for the point set S_2), is still an unsettled question.

Things get remarkably different if the constraining line segments (obstacles) in L do *not* have their endpoints included in the set S of sites. In this setting, the obstacles will just block visibility, without exerting proximity influence — a scenario useful in visibility, motion planning, and guarding problems. The resulting *visibility-constrained Voronoi diagram* is a piecewise linear structure of superlinear combinatorial complexity in $n = |S|$. Its regions are disconnected in general, being the union of polygons bounded by bisectors and visibility rays. Already for $|L| = 2$, when visibility is constrained to a 'window' between two segments, and the so-called *peeper's Voronoi diagram* is obtained, the size may increase to $\Theta(n^2)$; see Aurenhammer and Stöckl [103]. This bound remains valid if general *visibility angles* (of less than π) are attached to the sites, as in Figure 5.9; see Fan *et al.* [335]. $O(n^2)$ and $O(n^2 \log n)$ time algorithms, respectively, exist for these scenarios.

For the general case of n sites among m line segment obstacles, where the combinatorial complexity may be as large as $\Theta(n^2 m^2)$, Wang and Tsin [698] gave an $O(n^4 + n^2 m^2)$ algorithm, worst-case optimal for $m = \Omega(n)$.

A different, and geometrically even more complicated, type of constrained Voronoi diagram is the *geodesic Voronoi diagram*, also called the *Voronoi diagram for obstacles*. Here, the *geodesic distance* between a site p and a point x in the plane is the length of the *shortest obstacle-avoiding path* between p and x. It leads to a diagram with (path-)connected regions and linear size; see Aronov [66]. Once the geodesic Voronoi diagram is available, several proximity questions that respect the obstacles can be

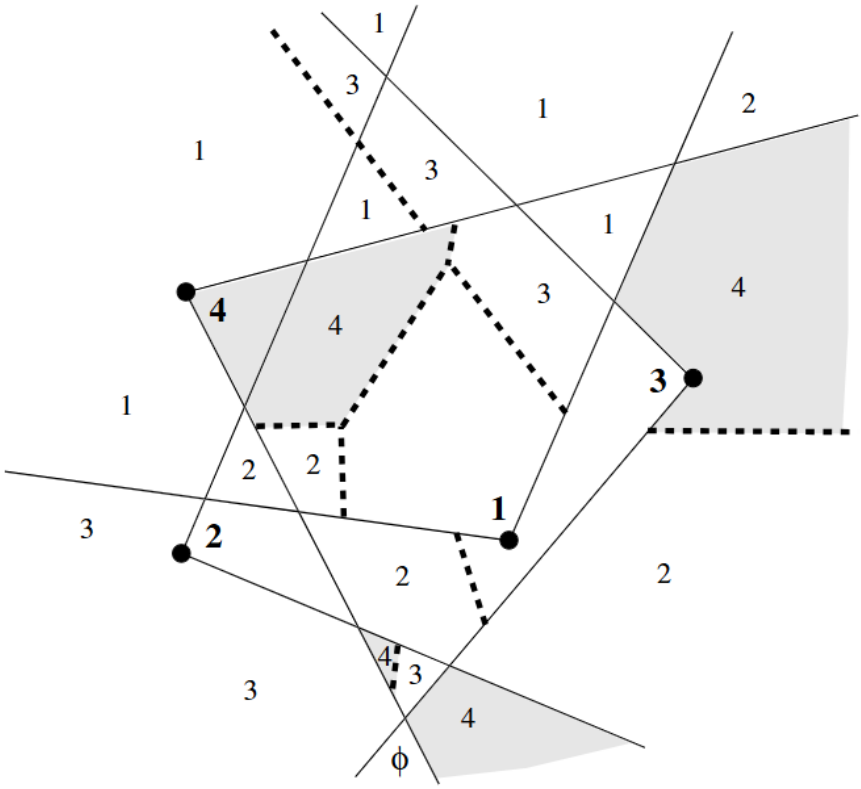


Figure 5.9. Voronoi diagram of four sites, constrained by angular visibility. The plane is dissected by visibility rays into cells $A(T)$, exclusively seen by subsets $T \subseteq S$, and partitioned by their (classical) Voronoi diagram $V(T)$. The region of a site p can be expressed as $\bigcup_{T \ni p} (A(T) \cap \mathrm{VR}(p, T))$, where $\mathrm{VR}(p, T)$ is the region of p in $V(T)$. The shaded areas above constitute the region of site 4. Cells not visible from S, like the bottommost cell $A(\emptyset)$, belong to no site.

answered efficiently, like nearest neighbor queries or the construction of a geodesic minimum spanning tree; cf. Chapter 8. The obstacles are usually modeled by a set L of non-crossing line segments. If all their endpoints are sites then the *bounded Voronoi diagram* is obtained.

Computing geodesic distances is not a constant-time operation, which complicates matters. Bisectors may consist of various linear and hyperbolic pieces. A subquadratic construction algorithm is given by Mitchell [546], and Hershberger and Suri [404] speeded up his *wavefront propagation* method (based on the *continuous Dijkstra technique*) to $O((n+m)\log(n+m))$ time and space, for n sites and m segments. The storage is reduced to optimal, $O(n+m)$, for the geodesic Voronoi diagram inside a simple m-gon, in Papadopoulou and Lee [582].

Voronoi diagrams induced by geodesic distances (shortest paths) on *surfaces in 3-space* are discussed in Section 7.1.

There are many other ways of measuring distances in the plane in the presence of line segments, or more generally, of polygonal objects or straight-line graphs. Let us briefly mention two of them at this place. Let A and B be two planar straight-line graphs. The *Hausdorff distance* between A and B is defined as the maximum of the minimum distances from A to B, and from B to A, respectively. An instance of the Voronoi diagram under this distance (the so-called *cluster Voronoi diagram*) will be discussed in Section 6.5.

The (continuous) *Fréchet distance* between A and B is based on parametrizing A and B; for simplicity, assume that A and B are two polygonal paths. As an intuitive interpretation, imagine a person walking all along A and his/her dog walking all along B. Then the Fréchet distance between the paths A and B corresponds to the minimum possible length of a leash that enables both, the person and the dog, to complete their walk. Computing Fréchet distances is quite elaborate. See Alt and Godau [51] for an efficient algorithm, which is based on the *parametric search* paradigm introduced in Megiddo [530].

Using a discrete version of this distance, Bereg *et al.* [125] recently analyzed the combinatorial complexity of the respective Voronoi diagram for a set of polygonal paths in two (and higher) dimensions.

5.5. Voronoi diagrams for curved objects

Some of the algorithms mentioned in Section 5.1 that compute the Voronoi diagram for line segments will also work for curved objects.

The *plane sweep algorithm* in Fortune [344] elegantly handles arbitrary sets of *circles* (i.e., the additively weighted Voronoi diagram, or Johnson–Mehl model; see Section 7.4) without modification from the point site case. Circular arcs, however, cannot be treated easily with that approach. Yap's *divide & conquer algorithm* [712] allows sets of disjoint segments of arbitrary *degree-two curves*. A *randomized incremental algorithm* for general *curved objects* is given by Alt *et al.* in [50]. They show that complicated curved objects can be partitioned into 'harmless' ones by introducing new points. All these algorithms achieve an optimal running time, $O(n \log n)$, but are not easy to implement.

The topic of this section is a simple and practical algorithm for computing the Voronoi diagram of a set of general (not necessarily polygonal) sites, developed in Aichholzer *et al.* [25]. In fact, such diagrams may have all kinds of artifacts. Their edge graph may be disconnected, and their bisectors may be closed curves. In particular, the abstract Voronoi

diagram setting in Klein [461] and Klein *et al.* [466] (see Section 7.5) does not apply. Moreover, divide & conquer is usually involved when emphasis is on the *bottom-up phase* (i.e. the merge step; cf. Section 3.3), even if the sites are of relatively simple shape. This is due to the missing separability condition for the sites, which would prevent the merge chain from cycling and breaking into several components.

The idea is to put emphasis on the *top-down phase*, namely, the divide step. The edge graph of the Voronoi diagram can be divided into a certain *tree* that corresponds to the *medial axis* of a (generalized) planar domain, as will be described in Subsection 5.5.1. (Usually, it is the other way round: Voronoi diagram algorithms are applied to compute the medial axis; cf. Section 6.6.) Division into base cases is then possible, which, in the bottom-up phase, can be merged by trivial concatenation. The resulting construction algorithm, in Subsection 5.5.2, is *not bisector-based* and merely computes dual links between the sites — similar to Delaunay triangulation methods. This guarantees computational simplicity and numerical stability. No part of the Voronoi diagram, once constructed, has to be discarded again.

5.5.1. *Splitting the Voronoi edge graph*

In the Voronoi diagram for general objects to be considered now, sites are allowed to be two-dimensional objects (any topological disks), one-dimensional objects (topological line segments), or simply points.

Let S denote the given set of sites. The distance of a point x to a site $s \in S$ is given by $\min_{y \in s} d(x, y)$, where d denotes the Euclidean distance function. As done e.g. in [50, 712], we define the Voronoi diagram, $V(S)$, of S via its *edge graph*, G_S, which is the set of all points having more than one closest point on the union of all sites. Our aim is to relate the graph G_S to the medial axis of a generalized planar domain. In this way, we will be able to construct the Voronoi diagram $V(S)$ by means of the medial axis algorithm presented in Subsection 5.5.2.

An edge of G_S containing points equidistant from two or more different points on the same site s is called a *self-edge* for s. The *regions* of $V(S)$ are the maximal connected subsets of the complement (of the closure) of G_S in the plane. The differences to a bisector-based definition of the Voronoi diagram should be noticed. Self-edges are ignored in such a definition unless the sites are split into suitable pieces. Such pieces, however, share boundaries — a fact that, if not treated with care, may give rise to unpleasant phenomena like two-dimensional bisectors.

To get rid of the unbounded components of the diagram, we include a surrounding circle, Γ, into the set S of sites, in a way such that each vertex of $V(S\backslash\{\Gamma\})$ is also a vertex of $V(S)$.

Removal of certain points from the edge graph G_S of $V(S)$ will now break all its cycles and make it a *tree*. Finding such points is non-trivial, in view of the possible presence of self-edges. For a site $s \neq \Gamma$, let $p(s)$ be a point on s with smallest y-coordinate, and denote with $q(s)$ the closest point on G_S vertically below $p(s)$. Then the geometric graph

$$T_S = G_S\backslash\{q(s) \mid s \in S\backslash\{\Gamma\}\}$$

can be shown to be a tree.

Our next goal is to interpret the tree T_S as the medial axis of a generalized planar domain. We first consider $V(S)$ to be the medial axis of a planar shape B, by taking the surrounding circle Γ as part of the shape boundary, and considering each remaining site $s \in S$ as a (possibly degenerate) 'hole'. That is, we define

$$B = B_0\backslash\{s \in S \mid s \neq \Gamma\},$$

where B_0 denotes the disk bounded by Γ. The medial axis $M(B)$ is just the (closure of the) edge graph G_S of $V(S)$. We now want to combinatorially disconnect the shape B at appropriate positions, such that the medial axis of the resulting domain corresponds to the tree decomposition T_S of $V(S)$ above. Observe from Lemma 5.4 (in the next subsection, 5.5.2) that any *maximal inscribed disk* can be used to split the medial axis of a shape into two (or more) components which share a point at the disk's center. In order to adapt this fact to our situation, the notion of a so-called augmented domain is introduced. Its definition is recursive, as follows.

An *augmented domain* is a set A together with a projection $\pi_A : A \to \mathbf{R}^2$. Initially, A is the original shape B, and the associated projection π_B is the identity. Now, consider a maximal inscribed disk D of an augmented domain A, which touches the boundary ∂A of A at exactly two points u and v. Denote by $\overset{\frown}{uv}$ and $\overset{\frown}{vu}$ the two circular arcs which the boundary of D is split into. The new augmented shape, A', which is obtained from A by splitting it with D, is defined as

$$A' = A^0 \cup D^1 \cup D^2,$$

where $A^0 = \{(x, 0) \mid x \in A\backslash D\}$, $D^1 = \{(x, 1) \mid x \in D\}$, and $D^2 = \{(x, 2) \mid x \in D\}$. The associated projection is

$$\pi_{A'} : A' \to \mathbf{R}^2, \quad (x, i) \mapsto \pi_A(x).$$

We say that the line segment in A between points (x,i) and (y,j) is *contained* in A' if one of the following conditions is satisfied:

(1) $i = j$ and the line segment $\overline{xy}$ avoids ∂D,
(2) $\{i,j\} = \{0,1\}$ and $\overline{xy}$ intersects the arc $\overset{\frown}{uv}$, or
(3) $\{i,j\} = \{0,2\}$ and $\overline{xy}$ intersects the arc $\overset{\frown}{vu}$.

For any two points (x,i) and (y,j) in A', their *distance* now can be defined. It equals the distance of $\pi_A(x)$ and $\pi_A(y)$ in the plane, provided the connecting line segment is contained in A', and is ∞, otherwise. An (open) disk in A' with center (m,i) and radius ϱ is the set of all points in A' whose distance to (m,i) is less than ϱ. Such a disk is said to be *inscribed* in A' if its projection into the plane is again an open disk.

Having specified inscribed disks for A', the boundary of A' and the medial axis of A' can be defined as in the case of (usual) planar shapes. In particular, $\partial A'$ derives from ∂A by disconnecting the latter boundary at the contact points u and v of the splitting disk D, and reconnecting it with the circular arcs $\overset{\frown}{uv}$ and $\overset{\frown}{vu}$. See Figure 5.10, which shows the boundary of a domain augmented with two disks, one for each of its two holes (i.e., the sites s_1 and s_2).

Concerning the medial axis, every maximal inscribed disk in A different from D corresponds to exactly one maximal inscribed disk in A'. The medial axis of A' therefore is the same geometric graph as $M(A)$, except that the edge of $M(A)$ containing the center of D is split into two disconnected edges which both have the center of D as one of their endpoints. These two points are two leaves of $M(A')$.

To draw the connection to the edge graph G_S of $V(S)$, the initial shape B from above is augmented with $|S|-1$ maximal inscribed disks,

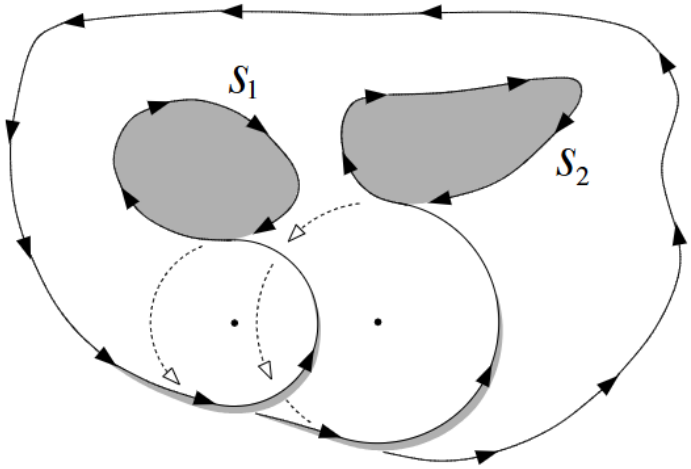


Figure 5.10. Oriented boundary of an augmented domain (from [25]).

namely, the ones centered at the points $q(s) \in G_S$, where $q(s)$ was the vertical projection onto G_S of a point with smallest y-coordinate on the site s. Denote by A_S the resulting domain after these $|S| - 1$ augmentation steps. We may conclude the main finding of this subsection as follows.

Lemma 5.3. *The tree* $T_S = G_S \backslash \{q(s) \mid s \in S \backslash \{\Gamma\}\}$ *is the medial axis of the augmented domain* A_S.

From the algorithmic point of view, augmenting B amounts to connecting its boundary ∂B to a *single* cyclic sequence, ∂A_S, that consists of pieces from ∂B and from circles bounding the splitting disks. (One-dimensional sites contribute to ∂B with two curves, one for either orientation, and the special case of point sites can be handled consistently.) Each such boundary piece is used exactly once on ∂A_S, and traversing ∂A_S corresponds to tracing the medial axis tree $M(A_S)$ in *preorder* (i.e., root first, then the subtrees for its children recursively, in radial order).

Of course, the construction of ∂A_S is trivial once the *splitting disks* are available. The non-trivial task is to find these disks D_i, one for each site $s_i \in S \backslash \{\Gamma\}$. Recall that D_i is horizontally tangent to s_i at a lowest point $p(s_i)$ of s_i. The center $q(s_i)$ of D_i lies on the edge graph G_S of $V(S)$ but, of course, D_i needs to be found without knowledge of G_S. Indeed, a simple and efficient *plane sweep* can be applied in $O(n \log n)$ time if the sites in S are described by a total of n objects, each being manageable in constant time.

Note that ∂A_S then consists of $\Theta(n)$ pieces, that either bound splitting disks, or pieces that stem from site boundaries. The former pieces are used only to link the site segments in the correct cyclic order. They do not play any geometric role, and can be ignored when the medial axis algorithm of Subsection 5.5.2 is applied to A_S in order to compute the desired Voronoi diagram $V(S)$.

5.5.2. *Medial axis algorithm*

We now describe the medial axis algorithm for *circular arc shapes* in [27, 34] that is suited to our needs. This algorithm works without modifications for the augmented domains introduced in Subsection 5.5.1.

Let A be some planar shape bounded by n circular arcs. We restrict attention to circular curves, because they are well suited to approximate more general objects accurately (for example, objects bounded by *spline curves*), and the case of line segments is trivially covered. Moreover, if the approximation by circular arcs (in particular, *biarcs*, i.e., smooth concatenations of two circular arcs) is done carefully, stability of the medial

axis can be guaranteed [34]. For a recent survey on medial axes for general objects, the interested reader is referred to Attali *et al.* [83].

For the shape A, call a disk $D \subseteq A$ *maximal* if there exists no disk D' different from D such that $D' \supset D$ and $D' \subseteq A$ holds. Then the *medial axis*, $M(A)$, of A can be defined as the (infinite) set of all centers of maximal disks for A. Observe that this definition is equivalent to the distance-based definition given in Subsection 5.5.1. The corresponding infinite set of disks (its union gives A) is commonly called the *medial axis transform* of A. Sometimes also the term *symmetric axis* is used to denote the medial axis (transform) of a shape.

$M(A)$ is connected and cycle-free and thus forms a tree. It consists of $O(n)$ *edges*, which are maximal pieces of straight lines and (possibly all four types of) conics. Endpoints of edges will be called *vertices* of $M(A)$. Compared to polygonal shapes, the medial axis for circular arc shapes is not more complicated, as both structures contain edges of algebraic degree 2 in general; see Figure 5.11.

We give a simple and practical *divide & conquer randomized* algorithm for computing $M(A)$. The costly part is delegated to the divide step, which basically will consist of *inclusion tests* for arcs in circles. The merge step is trivial; it just concatenates two partial medial axes. The expected runtime is bounded by $O(n^{3/2})$, and can be proven to be $O(n \text{ polylog } n)$ for several types of shape.

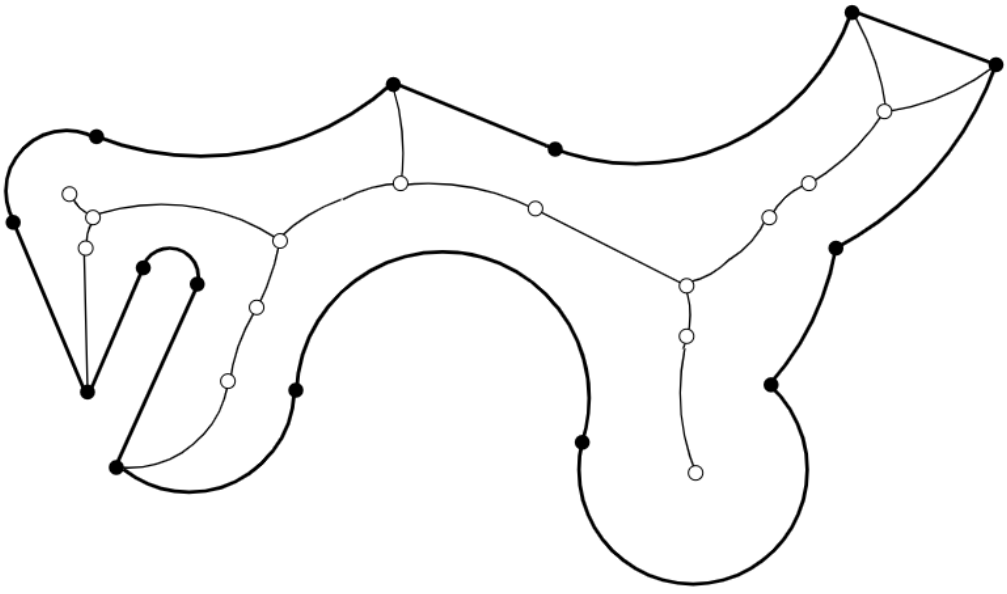


Figure 5.11. Medial axis of a shape bounded by circular arcs.

A qualitative difference to existing medial axis algorithms is that a *combinatorial* description of $M(A)$ is extracted first, which can then be directly (and robustly) converted into a geometric representation. The algorithm is based on the following simple though elegant *decomposition lemma* in Choi *et al.* [219].

Lemma 5.4. *Consider any maximal disk D for A. Let $A_1, \ldots, A_t$ be the connected components of $A \backslash D$, and denote by p the center of D.*

$$(1) \quad M(A) = \bigcup_{i=1}^{t} M(A_i \cup D),$$

$$(2) \quad \{p\} = \bigcap_{i=1}^{t} M(A_i \cup D).$$

That is, using any maximal disk one can compute the medial axes for the resulting components recursively, and then glue them together at the disk's center. By a moving argument, a *balanced* decomposition always exists.

Lemma 5.5. *There exists a maximal disk D for A such that at most $\frac{n}{2}$ arcs from ∂A are (completely) contained in each component of $A \backslash D$.*

For an edge e of $M(A)$, define $\mathrm{Walk}(e)$ as the path length in $M(A)$ from e to p^*, the center of a balanced disk as in Lemma 5.5. Further, define $\mathrm{Cut}(e)$ as the size of the smaller one between the two subtrees which constitute $M(A) \backslash \{e\}$. Any tree with small 'cuts' tends to have short 'walks', in the following respect.

Lemma 5.6. *If an edge e of $M(A)$ is chosen uniformly at random, then we have $E[\mathrm{Walk}(e)] = \Theta(E[\mathrm{Cut}(e)])$.*

Lemma 5.6 motivates the disk finding algorithm below, which combines random cutting with local walking. Its main subroutine, $\mathrm{MAX}(b)$, selects for an arc $b \subset \partial A$ its midpoint x and returns the unique maximal disk for A with x on its boundary. Let $c \geq 3$ be a (small) integer constant.

Procedure CUT(A)
  Put $A' = A$
  Repeat
    Choose a random arc b of $\partial A'$
    Compute D=MAX(b) and let A_0 be the larger
      component of A induced by D
    Assign $A' = A' \cap A_0$
  Until A_0 contains less than $n - \frac{n}{c}$ arcs
  Report D

Procedure WALK(A)
Choose a random arc b of ∂A
Compute D=MAX(b)
Let A_0 be the larger component induced by D
While A_0 contains more than $n - \frac{n}{c}$ arcs do
Let b_1 (b_2) be the first (last) complete arc of ∂A in A_0
Compute D_1=MAX(b_1) and D_2=MAX(b_2)
Assign to A_0 the smaller one of the respective larger components of A for D_1 and D_2
Memorize the corresponding disk $D \in \{D_1, D_2\}$
Report D

The disk finding algorithm now combines the CUT procedure and the WALK procedure as follows. The repeat loop of CUT and the while loop of WALK are executed by turns. Whenever CUT is closer to the goal (i.e., yields a smaller largest component than does WALK), we readjust the current disk for WALK to be that of CUT. Termination takes place in either WALK or CUT. Using Lemma 5.6, a bound of $O(\sqrt{n})$ on the expected number of total loop executions in CUT and WALK can be proven.

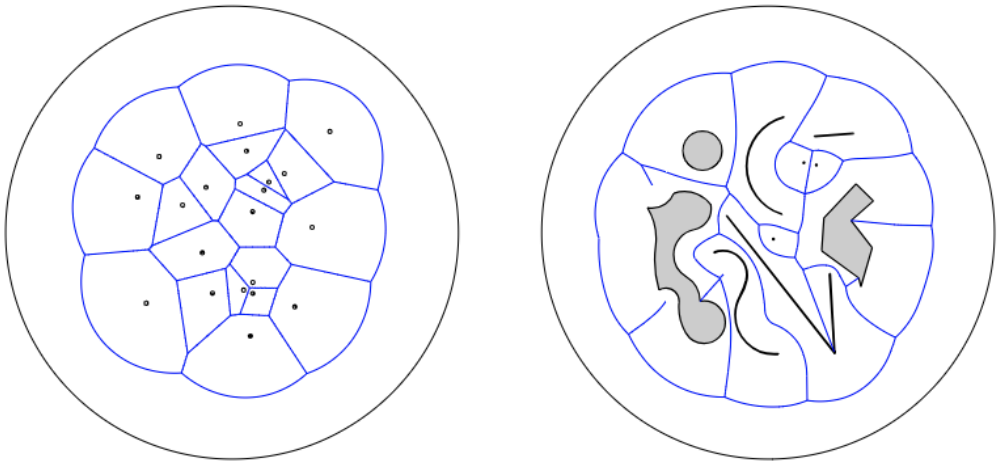
The costly part in both procedures is their subroutine MAX, whose expected number of calls obeys the same bound, $O(\sqrt{n})$. Computing $D = \text{MAX}(b)$ has a trivial implementation which runs in $O(n)$ time: We initialize the disk D as the (appropriately oriented) halfplane that supports b at its midpoint x. Then, for all remaining arcs $b_i \subset \partial A$ that intersect D, we shrink D so as to touch b_i while still being tangent to b at x. The most complex operation for shrinking D is computing the intersection of two circles. In particular, and unlike previous medial axis algorithms, *no conics* take part in geometric operations.

The randomized complexity for computing the medial axis is thus given by the recurrence relation

$$T(n) = T\left(\frac{1}{c} \cdot n\right) + T\left(\left(1 - \frac{1}{c}\right) \cdot n\right) + O(n^{3/2}) = O(n^{3/2}).$$

In many cases, however, the algorithm will perform substantially better. Let Δ be the *graph diameter* of $M(A)$ (i.e., the maximal number of edges on a path in this tree). If $\Delta = \Theta(\log n)$, which is the smallest value possible, then an overall runtime of $O(n \log^2 n)$ is met. For the other extreme case, $\Delta = \Theta(n)$, our strategy is even faster, $O(n \log n)$. The latter situation is quite relevant in practice, because an input shape, even if not branching much, is typically approximated by a large number of circular arcs.

Notice that the algorithm works exclusively on the *boundary* ∂A of A, except for a final step, where the conic edges of $M(A)$ are explicitly



(a) Point sites (b) Mixed sites

Figure 5.12. Voronoi diagrams for different kinds of sites, computed by the generalized medial axis algorithm (from [25]).

calculated and reassembled. This gives rise to increased numeric stability in comparison to other existing approaches. The necessary implementation details, including a treatment of degenerate situations, and a discussion of the base cases occurring in this divide & conquer algorithm, are given in the paper [27]. Figures 5.12(a) and 5.12(b) give two examples of the output.

We remark at this place that approximating *general objects* by circular arcs — instead of line segments — drastically reduces the input volume, namely from N line segments to $O(N^{\frac{2}{3}}) = n$ circular arcs, for fixed accuracy ε; see e.g. [34]. This is particularly desirable in applications concerning *motion planning* and *offset calculation* (a standard operation in image processing).

The δ-offset of a shape A is the *Minkowski sum* of A and a disk with radius δ. Whereas the class of polygons is not stable under taking offsets (because the δ-offset of a polygon will also contain circular arcs in its boundary), the class of circular arc shapes does have this property. This is useful, for example, in computer-aided manufactoring; see Held *et al.* [400] and Seong *et al.* [635]. In fact, the medial axis algorithm presented in this subsection delivers a link structure for ∂A that allows for offset computations without constructing $M(A)$ explicitly; see [25].

A different possibility for (polygon) offsetting, the so-called *mitered offset*, results from the straight skeleton of a polygon, discussed in Section 5.3.