3

Convex Hulls in Two Dimensions

The most ubiquitous structure in computational geometry is the convex hull (sometimes shortened to just "the hull"). It is useful in its own right and useful as a tool for constructing other structures in a wide variety of circumstances. Finally, it is an austerely beautiful object playing a central role in pure mathematics.

It also represents something of a success story in computational geometry. One of the first papers identifiably in the area of computational geometry concerned the computation of the convex hull, as will be discussed in Section 3.5. Since then there has been an amazing variety of research on hulls, ultimately leading to optimal algorithms for most natural problems. We will necessarily select a small thread through this work for this chapter, partially compensating with a series of exercises on related topics (Section 3.9).

Before plunging into the geometry, we briefly mention a few applications.

1. *Collision avoidance*. If the convex hull of a robot avoids collision with obstacles, so does the robot. Since the computation of paths that avoid collision is much easier with a convex robot than with a nonconvex one, this is often used to plan paths. This will be discussed in Chapter 8 (Section 8.4).
2. *Fitting ranges with a line*. Finding a straight line that fits between a collection of data ranges maps[1] to finding the convex region common to a collection of halfplanes (O'Rourke, 1981).
3. *Smallest box*. The smallest area rectangle that encloses a polygon has at least one side flush with the convex hull of the polygon and so the hull is computed at the first step of minimum rectangle algorithms (Toussaint, 1983*b*). Similarly finding the smallest three-dimensional box surrounding an object in space depends crucially on the convex hull of the object (O'Rourke, 1985*a*).
4. *Shape analysis*. Shapes may be classified for the purposes of matching by their "convex deficiency trees," structures which depend for their computation on convex hulls. This will be explored in Exercise 3.2.3[2].

The importance of the topic demands not only formal definition of a convex hull, but a thorough intuitive appreciation. The convex hull of a set of points in the plane is the shape taken by a rubber band stretched around nails pounded into the plane at each point. The boundary of the convex hull of points in three

[1]Maps via a duality relation to be studied in Chapter 6 (Section 6.5).

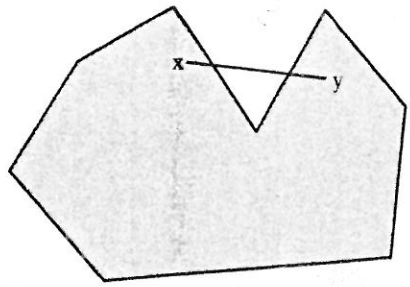


FIGURE 3.1 Any dent implies nonconvexity.

dimensions is the shape taken by plastic wrap stretched tightly around the points. We now examine a series of more formal definitions and approaches to convexity concepts. The remainder of the chapter is devoted to algorithms for constructing the hull.

3.1. DEFINITIONS OF CONVEXITY AND CONVEX HULLS

1. A set S is *convex* if $x \in S$ and $y \in S$ implies that the (closed) segment $xy \subseteq S$. This can be taken as the primary definition of convexity. Note that this definition does not specify any particular dimension for the points, whether S is connected, bounded or unbounded, closed or open. It should be clear from Figure 3.1. that any region with a "dent" is not convex, since two points straddling the dent can be found such that the segment they determine contains points exterior to the region. So, in particular any polygon with a reflex vertex[2] is not convex.
2. The *segment xy* is the set of all points of the form $\alpha x + \beta y$ with $\alpha \geq 0$, $\beta \geq 0$, and $\alpha + \beta = 1$.[3] For example, the midpoint $\frac{1}{2}(x + y)$ is realized with equal weights: $\alpha = \frac{1}{2}$ and $\beta = \frac{1}{2}$; the endpoints are achieved with one of the two weights zero. This algebraic view of a segment is quite useful both in mathematics and for computation. As an example of the latter, we will use it as the basis for finding the intersection point between two segments (Section 7.4.2).
3. A *convex combination* of points $x_1, \ldots, x_k$ is a sum of the form $\alpha_1 x_1 + \cdots + \alpha_k x_k$, with $\alpha_i \geq 0$ for all i and $\alpha_1 + \cdots + \alpha_k = 1$. So, a line segment consists of all convex combinations of its endpoints, and a triangle consists of all convex combinations of its three corners. In three dimensions, a tetrahedron is the convex combinations of its four corners. Note that all of these objects are themselves convex.

[2]Recall a vertex is reflex if its interior angle is $> \pi$.

[3]In the expression $\alpha x + \beta y$, α and β are scalar real numbers, while x and y are points or (equivalently) vectors.

4. The *convex hull* of a set of points S is the set of all convex combinations of points of S. In the mathematics literature, the convex hull of S is denoted by $\text{conv}\, S$. We will sometimes also use the notation $\mathscr{H}(S)$. Although it should be intuitively clear that the hull defined this way cannot have a dent, a proof is not immediate (Exercise 3.2.3[1]).
5. The convex hull of a set of points S in d dimensions is the set of all convex combinations of $d+1$ (or fewer points) of S. The distinction between this and the previous definition is that here only $d+1$ points need be used. Thus, the hull of a two-dimensional set is the convex combinations of its subsets of three points, which as we saw in (3) above, each determine a triangle. That the $(d+1)$-points definition is equivalent to the all-points definition (4) is known as Caratheodory's Theorem (Lay 1982, p. 17).
6. The convex hull of a set of points S is the intersection of all convex sets that contain S. This definition is perhaps clearer than the previous two because it does not depend on the notion of convex combination. However, the notion of "all convex sets" is not easily grasped.
7. The convex hull of a set of points S is the intersection of all halfspaces that contain S. This is perhaps the clearest definition, equivalent (although not trivially) to all the others. A halfspace in two dimensions is a *halfplane*: is the set of points on or to one side of a line. This notion generalizes to higher dimensions: a *halfspace* is the set of points on or to one side of a plane, and so on. Note that the convex hull of a set is a closed "solid" region, including all the points inside. Often the term is used more loosely in computational geometry to mean the boundary of this region, since it is the boundary we compute, and that implies the region. Usually confusion can be avoided.
8. The convex hull of a set of points S in the plane is the smallest convex polygon P that encloses S, smallest in the sense that there is no other polygon P' such that $P \supset P' \supseteq S$.
9. The convex hull of a set of points S in the plane is the enclosing convex polygon P with smallest area.
10. The convex hull of a set of points S in the plane is the enclosing convex polygon P with smallest perimeter. The equivalence of these last two definitions (9 and 10) with smallest in terms of subset nesting (8), is intuitively but not mathematically obvious (Exercise 3.2.3[6]). However, none of these three definitions of the boundary suggest an easy algorithm.
11. The convex hull of a set of points S in the plane is the union of all the triangles determined by points in S. This is just a restatement of 5 above, but in a form that hints at a method of computation.

The remainder of this chapter concentrates on algorithms for constructing the boundary of the convex hull of a finite set of points in two dimensions. We start

8.3.1. Minkowski Sum

Let A and B be two sets of points in the plane. If we establish a coordinate system, the points can be viewed as vectors in that coordinate system. Define the *sum* of A and B in the most natural manner possible: $A+B=\{x+y|x\in A, y\in B\}$, where $x+y$ is vector sum of the two points. This is known as the *Minkowski sum* of A and B. It will be a little easier to grasp the meaning of this abstract idea by considering the Minkowski sum of a point x and a set B: $x+B=\{x+y|y\in B\}$. This is just a copy of B translated by the vector x, for each point y of B is moved by x. So $A+B=\bigcup_{x\in A}(x+B)$ is the union of copies of B, one for each $x\in A$. Now, suppose A is a polygon P, and B is a disk R centered on the origin. Then $P+R$ can be viewed as many copies of R, translated by x for all $x\in P$. Since R is centered on the origin, $x+R$ will be centered on x. So $P+R$ amounts to placing a copy of R centered on top of every point of P. Now, it should be clear that $P+R$ is precisely the expanded region P'.

Let us examine the expansion of the triangle obstacle in Figure 8.4*b*. A copy of R is placed at each vertex of the triangle when x is a vertex, and when x lies on an edge, the tangents between these vertex disks are generated. For x interior to the triangle, $x+R$ lies inside P'.

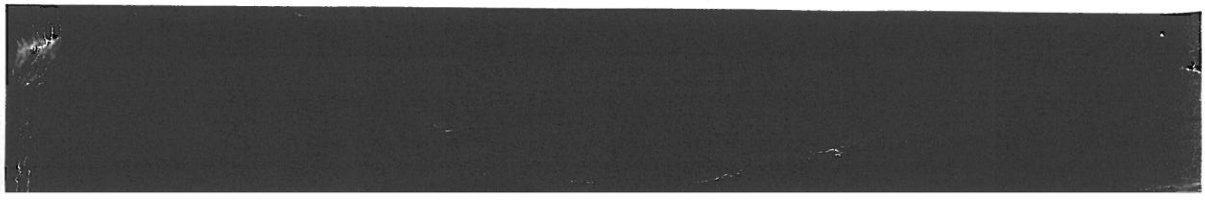
8.3.2. Conceptual Algorithm

We return now to the problem of moving a disk among polygonal obstacles. We will sketch a conceptual algorithm, but will not provide details. First, grow every obstacle by the disk R by constructing the Minkowski sum with R. We have not described how the grown obstacles can be constructed algorithmically; this issue we defer to the next section. Second, form the union of the grown obstacles. Again, we do not discuss how this can be done; it is not trivial. If t, the destination, is in a different "component" of the plane than is s, there is no free path from s to t. If they are in the same component, there is a path, and the shortest path can be found by modifying the visibility graph to include the appropriate arcs of the circles. We will not describe this algorithm any further, but it can be accomplished all in $O(n^2 \log n)$ time (Chew, 1985).

8.4. TRANSLATING A CONVEX POLYGON

When the robot is a convex polygon, we come to a serious complication: rotation of the robot might be necessary to move from one location to another. We will defer consideration of rotation until Section 8.5. Here, we restrict the motion to translations only.

The task is still more complicated than moving a disk, but fortunately the idea of growing obstacles by Minkowski sums still works. We explain the basic idea with an example before discussing algorithmic details.



5

Voronoi Diagrams

In this chapter, we study the Voronoi diagram, a geometric structure second in importance only to the convex hull. In a sense, a Voronoi diagram records everything one would ever want to know about proximity to a set of points (or more general objects). Often one does want to know detail about proximity: who is closest to whom, who is furthest, and so on. The concept is more than a century old, discussed in 1850 by Dirichlet and in a 1908 paper of Voronoi.[1]

We start with a series of examples to motivate the discussion, and then plunge into the details of the rich structure of the Voronoi diagram (in Sections 5.2–5.3). It is necessary to become intimately familiar with these details before algorithms can be appreciated (in Section 5.4). Finally, we reveal the beautiful connection between Voronoi diagrams and convex hulls in Section 5.7. This chapter includes only one short piece of code, in Section 5.7.4.

5.1. APPLICATIONS: PREVIEW

1. *Fire Observation Towers*
   Image a vast forest containing a number of fire observation towers. Each ranger is responsible for extinguishing any fire closer to her tower than to any other tower. The set of all trees for which a particular ranger is responsible constitutes the "Voronoi polygon" associated with her tower. The Voronoi diagram maps out the lines between these areas of responsibility: the spots in the forest that are equidistant from two or more towers. (A look ahead to Figure 5.5 may aid intuition.)
2. *Towers on Fire*
   Imagine now the perverse situation where all the rangers ignite their towers simultaneously, and the forest burns at a uniform rate. The fire will spread in circles centered on each tower. The points at which the fire quenches because it reaches previously consumed trees, are those points equidistant from two or more towers, which are exactly the points on the Voronoi diagram.
3. *Nearest Neighbor Clustering*
   A technique frequently employed in the field of pattern recognition is to map a set of target objects into a feature space by reducing the objects to points whose coordinates are feature measurements. The example of

[1]See Aurenhammer (1991) for a history.

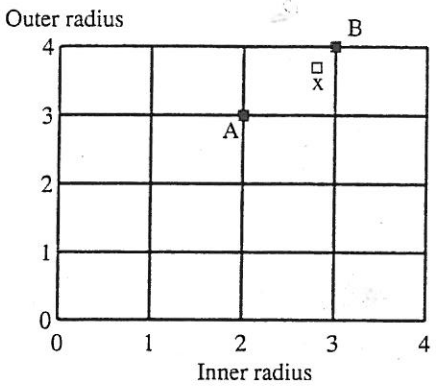


FIGURE 5.1 x is closer to B than to A.

five tailor's measurements from Section 4.5.1 can be viewed as defining such a feature space. The identity of an object of unknown affiliation then can be assigned the nearest target object in feature space.

An example will make this clearer. Suppose a parts bin includes two types of nuts A and B, A with inner and outer diameters of 2 and 3 cm respectively, and B with diameters 3 and 4 cm. Feature space is the positive quadrant of the two-dimensional Euclidean plane, positive because neither radius can be negative. A maps to the point $(2,3)$, and B to the point $(3,4)$.

Suppose a vision system focuses on a nut x in the bin, and measures its inner and outer radii to be 2.8 and 3.7 cm. Knowing that there are measurement inaccuracies, and that only nuts of type A and B are in the bin, which type of nut is x? It is most likely to be a B nut, because its distance to B in feature space is 0.36, whereas its distance to A is 1.06. See Figure 5.1. In other words, the nearest neighbor of x is B, because x is in B's Voronoi polygon.

If there are many types of nuts, the identification task is to locate the unknown nut x in the Voronoi diagram of the target nuts. How this can be done efficiently will be discussed in Section 5.5.1.

4. *Facility location*
Suppose you would like to locate a new grocery store in an area with several existing, competing grocery stores. Assuming uniform population density, where should the new store be located to optimize its sales? One natural method of satisfying this vague constraint is to locate the new store as far away from the old ones as possible. Even this is a bit vague; more precisely we could choose a location whose distance to the *nearest* store is as large as possible. This is equivalent to locating the new store at the center of the largest empty circle, the largest circle whose interior contains no other stores. The distance to the nearest store is then the radius of this circle.

We show in Section 5.5.3 that the center of the largest empty circle must lie on the Voronoi diagram.

5. *Path Planning*
   Imagine a cluttered environment through which a robot must plan a path. In order to minimize the risk of collision, the robot would like to stay as far away from all obstacles as possible. If we restrict the question to two dimensions, and if the robot is circular, then the robot should remain at all times on the Voronoi diagram of the obstacles. If the obstacles are points (say thin poles), then this is the conventional Voronoi diagram. If the obstacles are polygons or other shapes, then a generalized version of the point Voronoi diagram determines the appropriate path.
   We revisit this example in Chapter 8 (Section 8.5.2).
6. *Crystallography*
   Assume a number of crystal seeds grow at a uniform, constant rate. What will be the appearance of the crystal when growth is no longer possible? It should be clear now that this is analogous to the forest fire, and that each seed will grow to a Voronoi polygon, with adjacent seed regions meeting along the Voronoi diagram. Voronoi diagrams have long been used to simulate crystal growth.[2]

The list of applications could go on and on, and we will see others in Section 5.5, but it is time to define the diagram formally.

5.2. DEFINITIONS AND BASIC PROPERTIES

Let $P=\{p_1, p_2, \ldots, p_n\}$ be a set of points in the two-dimensional Euclidean plane. These are called the *sites*. Partition the plane by assigning every point in the plane to its nearest site. All those points assigned to p_i form the *Voronoi region* $V(p_i)$.[3] $V(p_i)$ consists of all the points at least as close to p_i as to any other site:

$$V(p_i) = \{x : |p_i - x| \leq |p_j - x|, \forall j \neq i\}. \tag{5.1}$$

Note, we have defined this set to be closed. Some points do not have a unique nearest site, or *nearest neighbor*. The set of all points that have more than one nearest neighbor form the *Voronoi diagram* $\mathcal{V}(P)$ for the set of sites.

Later we will define Voronoi diagrams for sets of objects more general than points. We first look at diagrams with just a few sites before detailing their properties for larger n.

[2] See Schaudt and Drysdale (1991) for recent work.

[3] This is also called a "Voronoi polygon," "Dirichlet domain," a "Thiessen polygon," or a "Wigner-Seitz region." The Voronoi region is not a polygon by our definition of "polygon," because it might be unbounded.

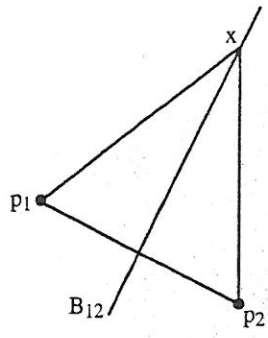


FIGURE 5.2 Two sites: $|p_1x| = |p_2x|$.

Two Sites

Consider just two sites, p_1 and p_2. Let $B(p_1, p_2) = B_{12}$ be the perpendicular bisector of the segment p_1p_2. Then every point x on B_{12} is equidistant from p_1 and p_2. This can be seen by drawing the triangle (p_1, p_2, x) as shown in Figure 5.2. By the side-angle-side theorem of Euclid,[4] $|p_1x| = |p_2x|$.

Three Sites

For three sites, it is clear that away from the triangle (p_1, p_2, p_3), the diagram contains the bisectors B_{12}, B_{23}, and B_{31}. What is not so clear is what happens in the vicinity of the triangle. Again from Euclid,[5] the perpendicular bisectors of the three sides of a triangle all pass through one point, the circumcenter, the center of the unique circle that passes through the triangle's vertices. Thus, the Voronoi diagram for three points must appear as in Figure 5.3. (However, the circumcenter of a triangle is not always inside the triangle as shown.)

5.2.1. Halfplanes

The generalization beyond three points is perhaps not yet clear, but it is certainly clear that the bisectors B_{ij} will play a role. Let $H(p_i, p_j)$ be the closed halfplane with boundary B_{ij} and containing p_i. Then $H(p_i, p_j)$ can be viewed as all the points that are closer to p_i than they are to p_j. Now recall that $V(p_i)$ is the set of all points closer to p_i than to any other site: in other words, the points closer to p_i than to p_1, *and* closer to p_i than to p_2, *and* closer to p_i than to p_3,

[4] Heath (1956, I.4).
[5] Heath (1956, IV.5).

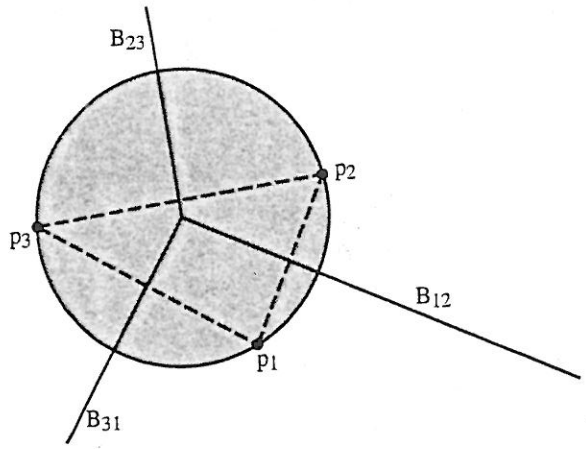


FIGURE 5.3 Three sites: bisectors meet at circumcenter.

and so on. This shows we can write this equation for $V(p_i)$:

$$V(p_i) = \bigcap_{i \neq j} H(p_i, p_j), \tag{5.2}$$

where the notation implies that the intersection is to be taken over all i and j such that $i \neq j$. Note the English conjunction "and" has been translated to set intersection.

Equation 5.2 immediately gives us an important property of Voronoi diagrams: the Voronoi regions are convex, for the intersection of any number of halfplanes is a convex set. When the regions are bounded, they are convex polygons. The edges of the Voronoi regions are called *Voronoi edges*, and the vertices are called *Voronoi vertices*. Note that a point on the interior of a Voronoi edge has two nearest sites, and a Voronoi vertex has at least three nearest sites.

Four Sites

The diagram of four points forming the corners of a rectangle is shown in Figure 5.4*a*.[6] Note the Voronoi vertex is of degree 4. Now suppose one site is moved slightly, as in Figure 5.4*b*. There is a sense in which this diagram is normal, and the one in Figure 5.4*a* is abnormal, or "degenerate." It is degenerate in that there are four cocircular points. We often will find it useful to exclude this type of degeneracy.

[6]This and several similar figures in this chapter were produced by the XYZ GeoBench software (Schorn 1991).

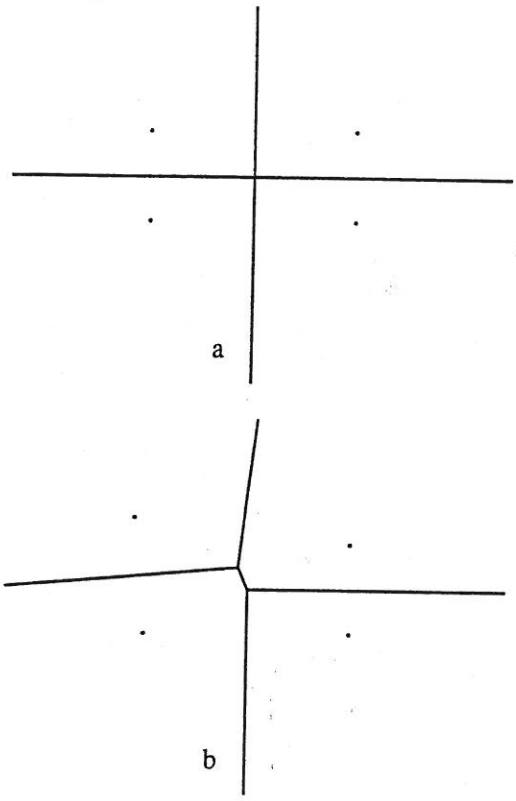


FIGURE 5.4 (*a*) Voronoi diagram of four cocircular points; (*b*) after moving upper left point.

Many Sites

A typical diagram with many sites is shown in Figure 5.5.[7] One Voronoi vertex is not shown in this figure: the two nearly horizontal rays leaving the diagram to the left are not quite parallel, and intersect at a Voronoi vertex about 50 cm left of the figure.

5.2.2. Size of Diagram

Although there are exactly n Voronoi regions for n sites, the total combinatorial size of the diagram conceivably could be quadratic in n, for any particular Voronoi region can have $\Omega(n)$ Voronoi edges (Exercise 5.3.3 [4]). We now show that this is in fact not the case, that the total size of the diagram is $O(n)$.

Let us assume for simplicity that no four points are cocircular, and therefore every Voronoi vertex is of degree three. Construct the *dual* graph G (Section

[7]This diagram was produced with code written by Susan Weller, John Kutcher, and Catherine Schevon.

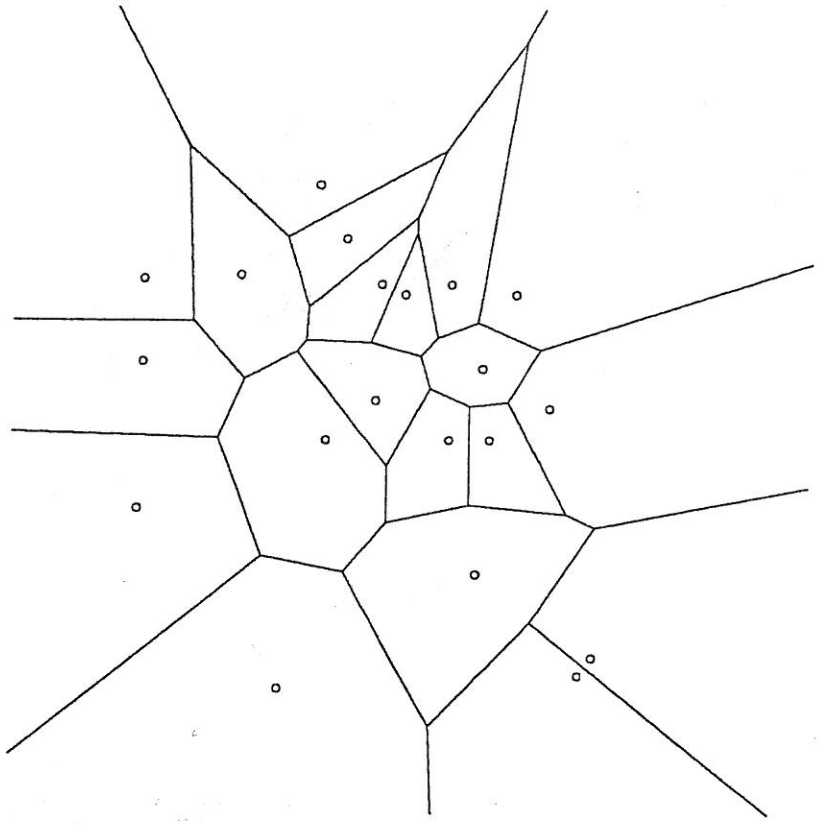


FIGURE 5.5 Voronoi diagram of $n = 20$ sites.

4.4) for a Voronoi Diagram $\mathscr{V}(P)$ as follows: the nodes of G are the sites of $\mathscr{V}(P)$, and two nodes are connected by an arc if their corresponding Voronoi polygons share a Voronoi edge (share a positive length edge).

Now observe that this is a planar graph: we can embed each node at its site, and all the arcs incident to the node can be angularly sorted the same as the polygon edges. Moreover, all the faces of G are triangles, corresponding to the degree-three Voronoi vertices. This claim will be made clearer below (Fig. 5.6).

We previously showed that Euler's formula implies that a triangulated planar graph with n vertices has $3n - 6$ edges and $2n - 4$ faces; see Section 4.1.5, Theorem 4.1.1. Since the faces of G correspond to Voronoi vertices, and the edges of G correspond to Voronoi edges (since each arc of G crosses a Voronoi edge), we have shown that the number of Voronoi vertices, edges, and faces are $O(n)$.

If we now remove the assumption that no four points are cocircular, the graph is still planar, but not necessarily triangulated. For example, the dual of the diagram shown in Figure 5.4*a* is a quadrilateral. However, such nontriangulated graphs have fewer edges and faces, so the $O(n)$ bounds continue to hold.

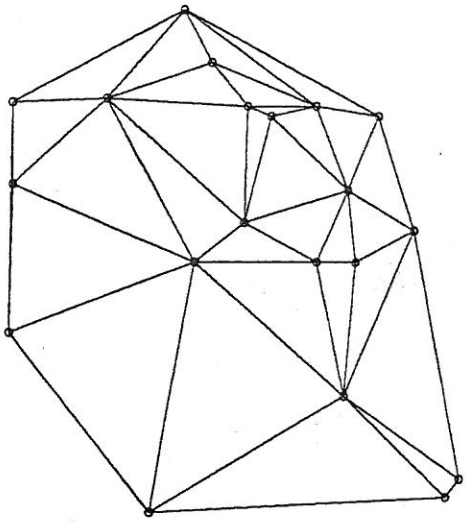


FIGURE 5.6 Delaunay triangulation for the sites in Figure 5.5.

One consequence of the $3n - 6$ edge bound is that the average number of edges of a Voronoi polygon is no more than six (Exercise 5.3.3[5]).

5.3. DELAUNAY TRIANGULATIONS

In 1934, Delaunay proved that when the dual graph is drawn with straight lines, it produces a planar triangulation of the Voronoi sites P (if no four sites are cocircular), now called the *Delaunay triangulation* $\mathscr{D}(P)$. Figure 5.6 shows the Delaunay triangulation for the Voronoi diagram in 5.5, and Figure 5.7 shows the Delaunay triangulation superimposed on the corresponding Voronoi diagram. Note that it is not immediately obvious that using straight lines in the dual would avoid crossings in the dual; the dual segment between two sites does not necessarily cross the Voronoi edge shared between their Voronoi regions, as is evident in Figure 5.7. We will not prove Delaunay's theorem now, but rather will wait until we have gathered more properties of Voronoi diagrams and Delaunay triangulations, when the proof will be easy.

5.3.1. Properties of Delaunay Triangulations

Because the Delaunay triangulation and Voronoi diagram are dual structures, each contains the same "information" in some sense, but represented in a rather different form. To gain a grasp on these complex structures, it is important to have a thorough understanding of the relationships between a Delaunay triangulation and its corresponding Voronoi diagram. We list without proof several Delaunay properties, and follow with a more substantive list of

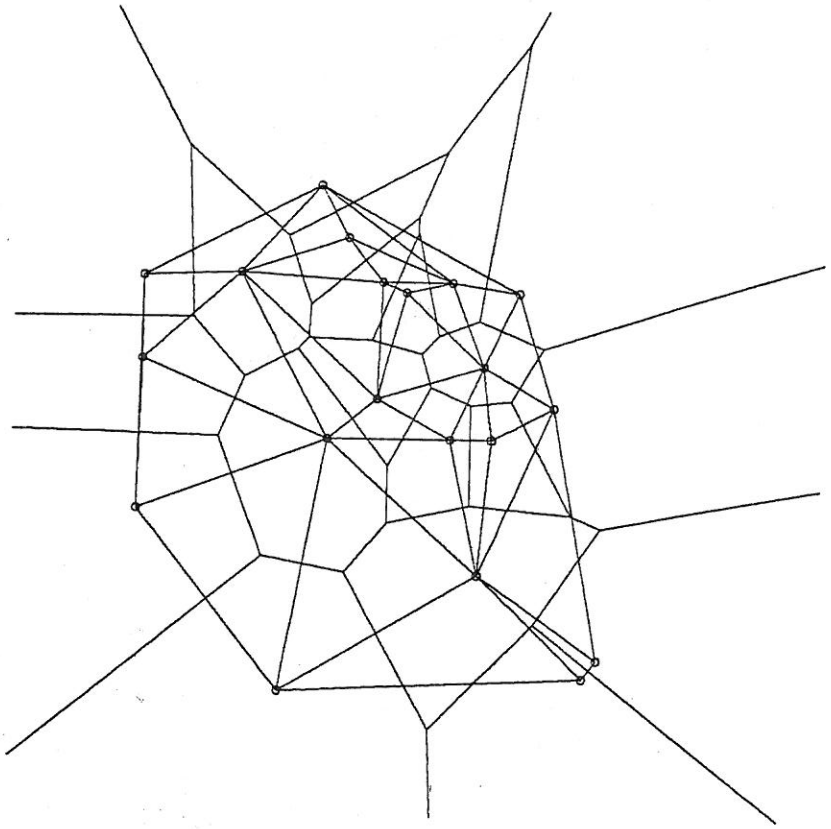


FIGURE 5.7 Delaunay triangulation and Voronoi diagram: Figures 5.5 and 5.6 together.

Voronoi properties.[8] Only the properties **D6** and **D7** have not been mentioned before. Fix a set of sites P.

D1. $\mathscr{D}(P)$ is the straight-line dual of $\mathscr{V}(P)$. This is by definition.

D2. $\mathscr{D}(P)$ is a triangulation if no four points of P are cocircular: every face is a triangle. This is Delaunay's theorem. The faces of $\mathscr{D}(P)$ are called *Delaunay triangles*.

D3. Each face (triangle) of $\mathscr{D}(P)$ corresponds to a vertex of $\mathscr{V}(P)$.

D4. Each edge of $\mathscr{D}(P)$ corresponds to an edge of $\mathscr{V}(P)$.

D5. Each node of $\mathscr{D}(P)$ corresponds to a region of $\mathscr{V}(P)$.

D6. The boundary of $\mathscr{D}(P)$ is the convex hull of the sites.

D7. The interior of each (triangle) face of $\mathscr{D}(P)$ contains no sites. (Compare **V5.**)

[8]Here I am following the pedagogic lead of Preparata and Shamos (1985, Section 5.5.1). In addition, some notation is borrowed from Okabe, Boots, and Sugihara (1992).

Properties **D6** and **D7** here the most interesting; they can be verified in Figures 5.6 and 5.7.

5.3.2. Properties of Voronoi Diagrams

V1. Each Voronoi region $V(p_i)$ is convex.

V2. $V(p_i)$ is unbounded iff p_i is on the convex hull of the point set. Compare property **D6**.

V3. If v is a Voronoi vertex at the junction of $V(p_1)$, $V(p_2)$, and $V(p_3)$, then v is the center of the circle $C(v)$ determined by p_1, p_2, and p_3. (This claim generalizes to Voronoi vertices of any degree.)

V4. $C(v)$ is the circumcircle for the Delaunay triangle corresponding to v.

V5. The interior of $C(v)$ contains no sites.

V6. If p_j is a nearest neighbor to p_i, then (p_i, p_j) is an edge of $\mathscr{D}(P)$.

V7. If there is some circle through p_i and p_j which contains no other sites, then (p_i, p_j) is an edge of $\mathscr{D}(P)$. The reverse also holds: for every Delaunay edge, there is some empty circle.

Property **V7** is the least intuitive, but is an important characterization of Delaunay edges, which is used in several proofs later on. This is the only property we will prove formally.

Theorem 5.3.1. ab $\in\mathscr{D}$(P) *iff* $\exists$ *an empty circle through* a *and* b: *the closed disk bounded by the circle contains no sites of* P *other than* a *and* b.

Proof. One direction is easy: if ab is a Delaunay edge, then $V(a)$ and $V(b)$ share a positive-length edge $e \in \mathscr{V}(P)$. Put a circle $C(x)$ with center x on the interior of e, with radius equal to the distance to a or b. This circle is obviously empty of other sites. For were it not, if, say, site c were on or in the circle, x would be in $V(c)$ as well, but we know that x is only in $V(a)$ and $V(b)$.

The reverse implication is more subtle. Suppose there is an empty circle $C(x)$ through a and b, with center x. We aim to prove that $ab \in \mathscr{D}(P)$. Because x is equidistant from a and b, x is in the Voronoi regions of both a and b as long as no other point interferes with "nearest-neighborliness." But none does, because the circle is empty. Therefore, $x \in V(a) \cap V(b)$ (recall we defined Voronoi regions to be closed sets). Because no points are on the boundary of $C(x)$ other than a and b (by hypothesis), there must be freedom to wiggle x a bit and maintain emptiness. In particular, we can move x along B_{ab}, the bisector between a and b, and maintain emptiness while keeping the circle through a and b. See Figure 5.8. Therefore x is on a positive-length Voronoi edge (a subset of B_{ab}) shared between $V(a)$ and $V(b)$, and therefore $ab \in \mathscr{D}(P)$. □

We leave the proof of the other properties to intuition, exercises, and to Section 5.7.2.

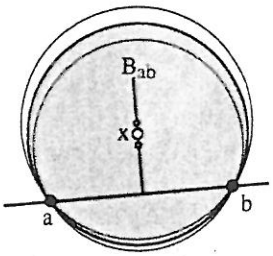


FIGURE 5.8 $C(x)$ is the shaded circle. Its center x can move along B_{ab} while remaining empty and still through a and b.

5.3.3. Exercises

1. *Regular polygon* [easy]. Describe the Voronoi diagram and Delaunay triangulation for the vertices of a regular polygon.
2. *Unbounded regions*. Prove property **V2**: $V(p_i)$ is unbounded iff p_i is on the convex hull of the point set. Do not assume the corresponding Delaunay property **D6**, but otherwise any Delaunay or Voronoi property may be employed in the proof.
3. *Nearest neighbors*. Prove property **V6**: if p_j is a nearest neighbor to p_i, then (p_i, p_j) is an edge of $\mathcal{D}(P)$. Any Delaunay or Voronoi property may be employed in the proof.
4. *High-degree Delaunay vertex*. Design a set of n points, no four cocircular, such that one vertex of the Delaunay triangulation has degree $n-1$.
5. *Average number of Voronoi polygon edges* [easy]. Prove that the number of edges in a Voronoi polygon, averaged over all polygons in any set of n points, does not exceed six (Preparata and Shamos, 1985, p. 211).
6. *Pitteway triangulations*. A triangulation of a set of points P is called a *Pitteway triangulation* (Okabe et al., 1992, p. 90) if, for each triangle $T = (a, b, c)$, every point in T has one of a, b, or c as its nearest neighbor among the points of P.
   a. Show by example that not every Delaunay triangulation is a Pitteway triangulation.
   b. Characterize those Delaunay triangulations that are Pitteway triangulations.

5.4. ALGORITHMS

The many applications of the Voronoi diagram and its inherent beauty have spurred researchers to invent a variety of algorithms to compute it. In this section, we will examine four algorithms, each rather superficially, for we will see in Section 5.7.2 that the Voronoi diagram can be computed using our convex hull code.

5.4.1. Intersection of Halfplanes

We could construct each Voronoi region separately, by intersecting $n-1$ halfplanes according to Equation 5.2. Although it is not immediately obvious, the intersection of n halfplanes may be constructed in $O(n \log n)$ time with a divide-and-conquer algorithm. Doing this for each site would cost $O(n^2 \log n)$.

5.4.2. Incremental Construction

Suppose the Voronoi diagram $\mathscr{V}$ for k points is already constructed, and now we would like to construct the diagram $\mathscr{V}'$ after adding one more point p. Suppose p falls inside the circles associated with several Voronoi vertices, say $C(v_1), \ldots, C(v_m)$. Then these vertices of $\mathscr{V}$ cannot be vertices of $\mathscr{V}'$, for they violate the condition that Voronoi vertex circles must be empty of sites (**V5**, Section 5.3.2). It turns out that these are the *only* vertices of $\mathscr{V}$ that are not carried over to $\mathscr{V}'$. It also turns out that these vertices are all localized to one area of the diagram. These vague observations can be made precise, and form one of the cleanest algorithms for constructing the Voronoi diagram (Green and Sibson, 1977). The algorithm spends $O(n)$ time per point insertion, for a total complexity of $O(n^2)$. In spite of its quadratic complexity, this has been the most popular method of constructing the diagram; see Field (1986) for implementation details. The incremental algorithm has been revitalized recently with randomization, which we will touch upon in Chapter 9.

5.4.3. Divide and Conquer

The Voronoi diagram can be constructed with a complex divide-and-conquer algorithm in $O(n \log n)$ time, first detailed by Shamos and Hoey (1975). It was this paper that introduced the Voronoi diagram to the computer science community. This time complexity is asymptotically optimal, but the algorithm rather difficult to implement. However, it can be done with careful attention to data structures; see Guibas and Stolfi (1985).

We pass over this historically-important algorithm to focus on some exciting recent developments.

5.4.4. Fortune's Algorithm

Until the mid-1980s, most implementations for computing the Voronoi diagram used the $O(n^2)$ incremental algorithm, accepting its slower performance to avoid the complexities of the divide-and-conquer coding. In 1985, Fortune (1987) invented a clever plane-sweep algorithm that is as simple as the incremental algorithms, but has worst-case complexity of $O(n \log n)$. We will now sketch the main idea behind this algorithm.

Plane-sweep algorithms (Section 2.2.1) pass a sweep line over the plane, leaving at any time the problem solved for the portion of the plane already swept, and unsolved for the portion not yet reached. A plane-sweep algorithm for construction the Voronoi diagram would have the diagram constructed behind the line. At first, this seems quite impossible, as Voronoi edges of a Voronoi region $V(p)$ would be encountered by the sweep line L *before* L encounters the site p responsible for the region. Fortune surmounted this seeming impossibility by an extraordinarily clever idea.[9]

[9] My exposition relies heavily on that of Guibas and Stolfi (1988), rather than on Fortune's original paper, which explained the algorithm in a rather different manner.

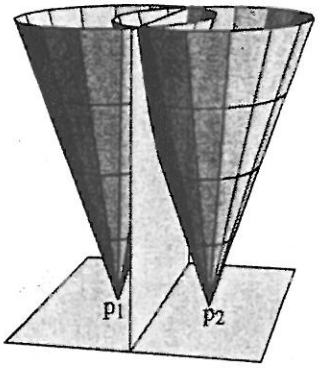


FIGURE 5.9 The curve of intersection of two cones projects to a line.

Cones

Imagine the sites in the xy-plane of a three-dimensional coordinate system. Erect over each site p a cone whose apex is at p, and whose sides slope at 45°. If the third dimension is viewed as time, the cone over p represents a circle expanding about p at unit velocity: after t units of time, its radius is t.

Now consider two nearby cones, over sites p_1 and p_2. They intersect in a curve in space. Recalling the expanding circles view of the Voronoi diagram, it should come as no surprise that this curve lies entirely in a vertical plane,[10] the plane orthogonal to the bisector of p_1 p_2. (See Figure 5.9.) Thus, although the intersection is curved in three dimensions, it projects to a straight line on the xy-plane.

It is but a small step from here to the claim that if the cones over all sites are opaque, and they are viewed from $z = -\infty$, what is seen is precisely the Voronoi diagram!

Cone Slicing

We are now prepared to describe Fortune's idea. His algorithm sweeps the cones with a slanted plane π, slanted at 45° to the xy-plane. The sweep line L is the intersection of π with the xy-plane. Let us assume that L is parallel to the y-axis, and that its x-coordinate is l. See Figure 5.10. Imagine that π, as well as all the cones, are opaque, and again consider the view from $z = -\infty$.

To the $x > l$ side of L, only π is visible from below: it cuts below the xy-plane and so obscures the sites and cones. This represents the portion of the plane yet to be swept. To the $x < l$ side of L, the Voronoi diagram is visible up to the intersection of π with the right (positive x) "frontier" of cones. The intersection of π with any one cone is a parabola (a basic property of conic sections), and so the intersection of π with this right frontier projects to the xy-plane (and so appears from $z = -\infty$) as a "parabolic front," a curve composed of pieces of parabolas. See Figure 5.11. Two parabolas join at a spot

[10] The curve is a branch of a hyperbola, the conic section formed by intersection with a plane parallel to the axis of the cone.

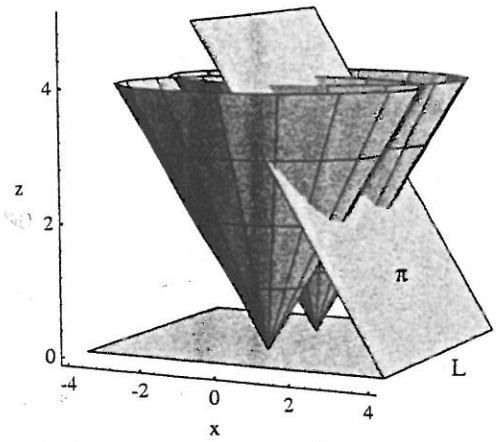


FIGURE 5.10 Cones cut by sweep plane. π and L are sweeping toward the right, $x \to \infty$

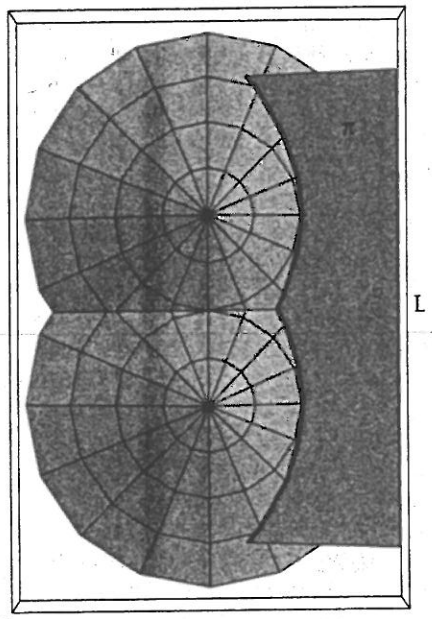


FIGURE 5.11 Figure 5.10 viewed from $z \approx -\infty$. The heavy curve is the parabolic front.

where π meets two cones. From our discussion of the intersection of two cones above, this must be at a Voronoi edge.

Parabolic Front

Now we finally can see how Fortune solved the problem of the sweep line encountering Voronoi edges prior to the generating sites: because his sweep plane π slopes at the same angle as the cone sides, L encounters a site p

exactly when π first hits the cone for p! So it is not the case that the Voronoi diagram is at all times constructed to the left of L, but it is at all times constructed *underneath* π, which means that it is constructed to the left of L up to the parabolic front, which lags L a bit.

What is maintained at all times by the algorithm is the parabolic front, whose joints trace out the Voronoi diagram over time, since these kinks all lie on Voronoi edges. Although we are by no means finished the algorithm description, we will make no attempt to detail it further here.

Finally, it should be clear that the algorithm only need store the parabolic front, which is of size $O(n)$, and is often $O(\sqrt{n})$. This is a significant advantage of Fortune's algorithm when n is large: the storage needed at any one time is often much smaller than the size of the diagram.

5.4.5. Exercises

1. $\mathscr{D}(P) \Rightarrow \mathscr{V}(P)$. Design an algorithm for computing the Voronoi diagram, given the Delaunay triangulation. Try to achieve $O(n)$ time complexity.
2. *One-dimensional Voronoi diagrams.* A one-dimensional Voronoi diagram for a set of points $P = \{p_1, \ldots, p_n\}$ on a line (say the x-axis) is a set of points $\mathscr{V}(P) = \{x_1, \ldots, x_{n-1}\}$ such that x_i is the midpoint of $p_i\, p_{i+1}$.

   Suppose you are given a set $X = \{x_1, \ldots, x_{n-1}\}$. Design criteria that will permit you to determine whether or not X is a one-dimensional Voronoi diagram of a set of points, and if so, determine P. How fast is the implied algorithm?
3. *Dynamic Voronoi diagrams.* Imagine a set of points moving on the plane, each with a fixed velocity and direction. Let $\mathscr{V}(t)$ be the Voronoi diagram of the points at time t. It is an unsolved problem to obtain tight bounds on the number of combinatorially distinct diagrams that can result over all time. Here I ask you to establish the best-known lower bound: $\Omega(n^2)$. In other words, find a set of n moving points such that $\mathscr{V}(t)$ changes its graphical structure cn^2 times for some constant c.

   No one has been able to find an example in which there are more than n^2 changes, but the best upper bound is about $O(n^3)$ (Fu and Lee, 1991) (Guibas, Mitchell, and Roos 1991).
4. *Arbitrary triangulation.* Design an algorithm to find an arbitrary triangulation of a point set P: a collection of diagonals incident to every point of P that partitions $\mathscr{H}(P)$ into triangles. The absence of the requirement that the triangulation be Delaunay permits considerable freedom in the design.
5. *Flipping algorithm.* Investigate the following proposed algorithm for constructing $\mathscr{D}(P)$. Start with an arbitrary triangulation of P. Then repeat the following procedure until $\mathscr{D}(P)$ is attained. Identify two adjacent triangles *abc* and *cbd* sharing diagonal *bc*, such that the quadrilateral *abcd* is convex. If d is inside the circumcircle of *abc*, then delete *cb* and add *ad*. Will this work?

5.5. APPLICATIONS IN DETAIL

We will now discuss five applications of the Voronoi diagram, in uneven detail: nearest neighbors, "fat" triangulations, largest empty circles, minimum spanning trees, and traveling salesperson paths.