A Sweepline Algorithm for Generalized Delaunay Triangulations*

Sven Skyum

November 1991

Abstract

We give a deterministic O(n log n) sweepline algorithm to construct the generalized Voronoi diagram for n points in the plane or rather its dual the generalized Delaunay triangulation. The algorithm uses no transformations and it is developed solely from the sweepline paradigm together with greediness. A generalized Delaunay triangulation can be based on an arbitrary strictly convex Minkowski distance function (including all L_p distance functions for $1 < p < \infty$) in contrast to ordinary Delaunay triangulations which are based on the Euclidean distance function.

1 Introduction

The Voronoi diagram for a set of points, called sites, in the plane, divides the plane into Voronoi regions, one for each site. The Voronoi region for a site is the set of points in the plane which are as close to the site as to any other site in the set. It is well known how to construct a Voronoi diagram in time O(n log n). See [1] for an overview. Applications such as solving various

*This research was (partially) supported by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM).

proximity problems are most effectively done by using Voronoi diagrams. Also in the area of motion planning, Voronoi diagrams can be applied [12].

Many generalizations of Voronoi diagrams have occurred in the literature. One way of generalizing is to allow sites to be other objects than points — eg disks, line segments etc. A natural generalization is to base the construction on other distance functions than the Euclidean one. Time $O(n \log n)$ methods for general Voronoi diagrams for L_p norm $1 \leq p \leq \infty$ and more general distance functions also exist [2, 8, 9].

A recent generalization has been proposed by Klein [5] and followed up by others [11, 6]. Here the concept of distance has been exchanged with the notion of bisecting curves for pairs of sites. A bisecting curve $J(p, q)$ for sites p and q divides the plane into a p-region and a q-region. The abstract Voronoi region for a site p is then the intersection of the various p-regions w.r.t. other sites q. Given a suitable set of conditions on the family of separating curves randomized $O(n \log n)$ algorithms for the construction of abstract Voronoi diagrams are given in [11, 6].

The dual graph to a Voronoi diagram, ie the Delaunay triangulation is defined to be the graph where the sites are the nodes and an edge between site p and q exists if and only if the boundaries of the Voronoi regions for p and q intersect. It is well known that apart from the abstract Voronoi diagrams mentioned above this is equivalent to saying that there is a circle through p and q for which the corresponding open disk does not contain any sites. Thus generalized Delaunay triangulations can be characterized by empty disks without referring to Voronoi diagrams and only the shape of the disks is of importance. This has the advantage that there are no problems involved in dealing with bisecting curves and their intersections.

Most algorithms for construction of Voronoi diagram and Delaunay triangulations are either incremental or use the divide and conquer paradigm. Fortune has presented [4] a sweepline algorithm for the Euclidean distance function. He uses a clever but rather artificial transformation * of the plane to delay events for new points to be included. For the L_1 and L_∞ norms a sweepline algorithm for the generalization of a Delaunay triangulation has been given in [14]. It has been open whether there exist sweepline algorithms for other distance functions and whether there exists one for the Euclidean case without introducing a transformation like the one in Fortune's paper.

We present an algorithm which constructs generalized Delaunay triangulation (and generalized Voronoi diagrams) w.r.t. arbitrary Minkowski distance functions (see [7] for an introduction to distance functions) defined by families of pseudo disks. We follow the line taken in [2, 3] and define pseudo disks from a given convex unit disk. This is slightly different from the notion used in [10]. We define our notion of pseudo disks in the next section and some of their fundamental properties are proven. In Section 3 we prove a number of Lemmas on which the correctness and the time analysis of the algorithm, to be presented in Section 4, are based.

2 Pseudo disks and their properties

In this section we define our notion of pseudo disks and prove a number of their fundamental properties. First some basic notation.

$(x(p), y(p))$ denotes the coordinates of a point $p \in \Re^2$.

For three points $p, q, r \in \Re^2$, $LT(p, q, r), (RT(p, q, r))$is true if p, q and r form a left (right) turn. That is

$$LT(p,q,r) \equiv (\begin{vmatrix} x(p) & y(p) & 1 \\ x(q) & y(q) & 1 \\ x(r) & y(r) & 1 \end{vmatrix} > 0)$$

and

$$RT((p,q,r) \equiv (\begin{vmatrix} x(p) & y(p) & 1 \\ x(q) & y(q) & 1 \\ x(r) & y(r) & 1 \end{vmatrix} > 0)$$

For two distinct points $p, q \in \Re^2$, $H_l(p, q)$ and $H_r(p, q)$ denote the half planes defined by p and q:

$$H_l(p, q) = \{r \in \Re^2 \mid LT(p, q, r)\}$$

and

$$H_r(p, q) = \{r \in \Re^2 \mid RT(p, q, r)\}$$

For a set M in $\Re^2$, ∂M denotes the boundary of M.

Finally $\overline{pq}$ denotes the line segment between two points p and q in $\Re^2$.

In [10] a compact (closed and bounded) set in $\Re^2$ with smooth boundary of positive curvature is called an *oval*. A family $\mathcal{D}$ of ovals is called a family of *pseudo disks* if and only if for every three non-colinear points there is a unique oval $D \in \mathcal{D}$ which has these three points on its boundary.

We define a family of pseudo disks slightly differently, namely by defining a pseudo unit disk U with centre in $(0,0)$ and then the family of pseudo disks to consist of all sets which are scaled translations of U. A pseudo disk in our notation is also a pseudo disk in the sense of [10], but not necessarily vice versa.

Definition 2.1 (pseudo disks) *A pseudo unit is with centre* $(0,0)$ *is a compact strictly convex set* U *with smooth boundary such that* $(0,0)$ *is an internal point of* U. *The* family of pseudo disks *given by a pseudo unit disk* U *is* $\{c + R \cdot U \mid c \in \Re^2, R \in \Re\}$. *The pseudo disk* $c + R \cdot U$ *is said to have centre* c *and radius* R.

The Minkowski distance *between* p *and* q *(*$d(p,q)$*) is the radius of the pseudo disk with centre* p *and* q *on its boundary. That is* $q \in \partial(p + d(p,q)U)$. *Note that* $d(\cdot,\cdot)$ *is not necessarily symmetric.*

Remarks
The set of L_p disks is for each $1 < p < \infty$, a family of pseudu disks while the set of L_1 or L_∞ disks are not, since they are neither strictly convex nor smooth.

Lemma 2.1 *For two different disks* D_1 *and* D_2 *with centres* c_1 *and* c_2, $|\partial D_1 \cap \partial D_2| \leq 2$.

Proof Let $D_1 = c_1 + R_1 U$ and $D_2 = c_2 + R_2 U$ be two disks.
If $c_1 = c_2$, then either D_1 and D_2 are identical or the boundaries of D_1 and D_2 do not intersect.
If $c_1 \neq c_2$, then let for $i \in \{1,2\}$ and $t > 0$, $D_i^{(t)} = c_i + tR_iU$. We will show that for all $t > 0$, $|\partial D_1^{(t)} \cap \partial D_2^{(t)}| \leq 2$. For sufficiently small ϵ, $D_1^{(\epsilon)} \cap D_2^{(\epsilon)} = \emptyset$. Now let $a > 0$ be minimal such that $D_1^{(a)} \cap D_2^{(a)} \neq \emptyset$. Then, if p and q are two different points of intersection, $\overline{pq} \subset \partial D_1^{(a)} \cap \partial D_2^{(a)}$ in contraction with the disks being strictly convex. Thus $|\partial D_1^{(a)} \cap \partial D_2^{(a)}| = 1$ and $\partial D_1^{(a)}$ and $\partial D_2^{(a)}$ have a common tangent l_a in the point of intersection. Consequently for

small ϵ, $\partial D_1^{(a+\epsilon)}$ and $\partial D_2^{(a+\epsilon)}$ will intersect in two points, since the common tangent l_a above does not separate the disks any more.
If for some $t > a$, $|\partial D_1^{(t)} \cap \partial D_2^{(t)}| \geq 2$, let $b > a$ be the minimal such that $|\partial D_1^{(b)} \cap \partial D_2^{(b)}| \geq 2$. Let l_b be the common tangent of a new point p of intersection. l_b must separate c_1 from c_2 because otherwise either the line through c_1 and p would intersect $\partial D_1^{(b)}$ in more than two points or the line through c_2 and p would intersect $\partial D_2^{(b)}$ in more than two points. Finally, since $\partial D_1^{(b)}$ and $\partial D_2^{(b)}$ intersect in at least two more points, l_b cannot separate $\partial D_1^{(a)}$ from $\partial D_2^{(a)}$. Hence l_b must intersect either $\partial D_1^{(a)}$ or $\partial D_2^{(a)}$ in more than two points. □

Existence of a disk D with three given non-colinear points p, q and r on its boundary follows from smoothness, which together with Lemma 2.1 implies the following fundamental properties of pseudo disks on which all Lemmas of Section 3 and the algorithm in Section 4 are based.

Property 2.1

1. *For a pseudo disk D und a line l, l intersects ∂D in zero, one or two points.*

2. *For three non-colineur points p, q and r there is a unique pseudo disk D_{pqr} with p, q und r on its boundary ∂D_{pqr}.*

3. *For two pseudo disks D_1 and D_2 with $p, q \in \partial D_1 \cap \partial D_2$ $(p \neq q)$ the following two statements are equivulent*

$$\begin{aligned} D_1 \cap H_l(p,q) &\subseteq D_2 \cap H_l(p,q) \\ D_1 \cap H_r(p,q) &\supseteq D_2 \cap H_r(p,q) \end{aligned}$$

4. *For a pseudo disk D, $p, q \in \partial D$ $(p \neq q)$ and $r \in H_l(p,q)$ the following statements are equivulent*

$$\begin{aligned} & r \in D \\ & D \cap H_l(p,q) \supseteq D_{pqr} \cap H_l(p,q) \\ & D \cap H_r(p,q) \subseteq D_{pqr} \cap H_r(p,q) \end{aligned}$$

3 Analysis of the problem

Let a family of pseudo disks be fixed in the following. When we refer to *distance, disk, cocircular*, they will among other things, always refer to the given set of pseudo disks. For simplicity all figures will be for the Euclidean case.

More notation is needed before we state the Lemmas.

Notation

For $s \in \Re$, l_s denotes the line $\{(x, y) \mid y = s\}$ and $Below_s$ denotes the closed half plane $\{(x, y) \mid y \leq s\}$.

For three different non-colinear points $p, q, r \in \Re^2$, c_{pqr} denotes the centre of D_{pqr}, R_{pqr} denotes the radius, and t_{pqr} denotes the unique point in D_{pqr} with maximal y-coordinate.

For a set of points S in $\Re^2$, M a subset of S and D a disk, $Empty_S(D, M, s)$ is stating that M is a subset of ∂D, $D \setminus \partial D$ contains no points from S, and D is below l_s. Formally

$$Empty_S(D, M, s) \equiv (M \subset \partial D) \wedge ((D \setminus \partial D) \cap S = \emptyset) \wedge (D \subset Below_s).$$

Usually it is clear from the context what the set of points S is, so the subscript S will be omitted.

All graphs involved in this paper will be planar. Hence the notion *graph* will be used to mean both an undirected graph in the normal sense and a *straight line embedding* of it.

As usual we will make some assumptions on the set of points S. They are all easy to overcome and the algorithm to be presented is not sensitive to them, but the Lemmas in this section will be simpler to state.

Assumption 1 *No four points in S are cocircular.*

A generalized Delaunay triangulation is defined as follows:

Definition 3.1 *For a set of points $S = \{p_1, p_2, \cdots, p_n\}$ in $\Re^2$, the* generalized Delaunay triangulation *of S is the graph $G = (S, E)$, where $E = \{\{p_i, p_j\} \mid$ there is a disk D where $Empty(D, \{p_i, p_j\}, \infty)\}$.*

As for the ordinary Delaunay triangulation of S, the generalized Delaunay triangulation is indeed a triangulation of the convex hull of S. It is the triangulation for which it holds that for all triangles $\triangle_{pqr}$ in the triangulation, $Empty(D_{pqr}, \{p, q, r\}, \infty)$ (Figure 1). Edges in the Delaunay triangulation will be called Delaunay edges.

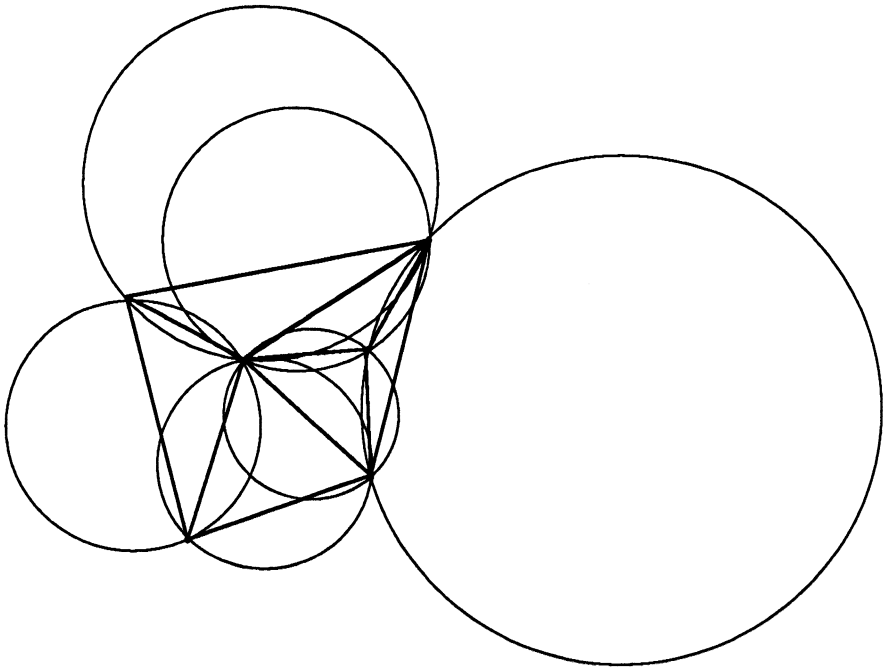

Figure 1: Delaunay triangulation and empty disks.

The algorithm to be presented in Section 4 is based on Definition 3.1. The paradigm on which it is based is, apart from the sweepline paradigm, that the algorithm is greedy.

The sweepline will be horizontal and will be moved upwards. The goal is for a specific sweepline l_s to have found all the edges between pairs of points below l_s which are known to be a Delaunay edge. In other words, all pairs of points $\{p, q\}$ in $S \cap Below_s$ for which there is a disk D such that $Empty(D, \{p, q\}, s)$, should have been identified. The sweepline status will then naturally contain information on pairs of points $\{p, q\}$ from $S \cap Below_s$ which are not yet known to be a Delaunay edge, but which might be. That is equivalent to saying that there is a corresponding disk D_{pq} where $Empty(D_{pq}, \{p, q\}, \infty)$

but ∂D_{pq} intersects the sweepline l_s in two points.

Thus an algorithm could be like the following (the sites are supposed to be sorted according to increasing y-coordinates):

add$\{p_1, p_2\}$ to E;
for $i := 3$ to n do
    addedges $\{p_j, p_k\}$, $j, k \leq 2$ for which there exists a disk D
        where $Empty(D, \{p_j, p_k\}, y(p_i))$;
od ;
add edges $\{p_j, p_k\}$ for which there is a disk D where
    $Empty(D, \{p_j, p_k\}\infty)$

The rest of this section contains of a number of Lemmas which enable us to efficiently implement the above algorithm. The main results are that for each new point p_i to be added to the structure, we can find in time O(log n) a point q in $S \cap Below_{y(p_i)}$ such that there is a disk D where $Empty(D, \{p_i, q\}y(p_i))$ (Lemmas 3.1 through 3.4) and that the only disks D_{pq} with $Empty(D_{pq}, \{p, q\}, \infty)$ (see above) we have to deal with, are among those given by three consecutive points on the outer region of the planar graph which has been constructed (Lemma 3.5).

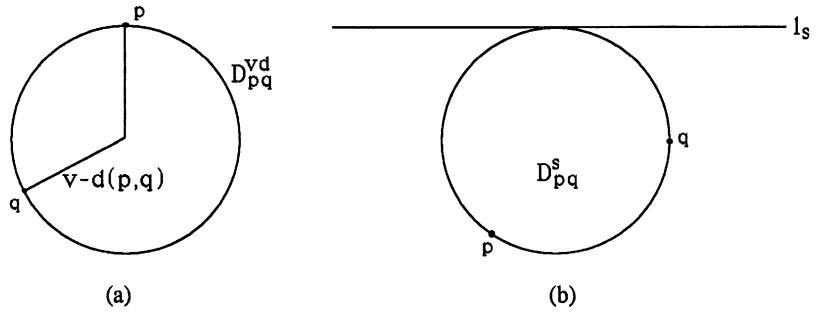

Figure 2: The disks D^{vd}_{pq}, D^{s}_{pq} and the v-distance.

First some more useful notations.

Assumption 2 *No two points in S have the same y-coordinate.*

Definition 3.2 *Let p and q be two distinct points in $\Re^2$. The disk D, with $p, q \in \partial D$ and horizontal tangent in p if $y(p) > y(q)$ and q otherwise, is denoted D^{vd}_{qp} (Figure 2).*

The v-distance *between two points* $p, q \in \Re^2$ *is the radius of* D^{vd}_{pq}.

If $s > \max\{y(p), y(q)\}$, D^s_{pq} *denotes the disk where* $p, q \in \partial D^s_{pq}$, l_s *is tangent to* ∂D^s_{pq} *in* t *and* $t \notin H_r(p, q)$.

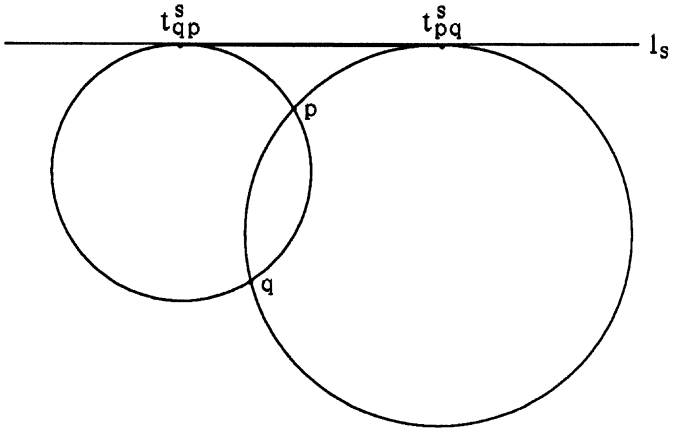

Figure 3: The line segment on l_s closer to p than to q w.r.t the v-distance.

Definition 3.3 *Let* $p, q \in \Re^2$ *and* $s > y(p) > y(q)$. *The left end right end points of the line segment* $\{r \in l_s \mid v - d(r, p) \leq v - d(r, q)\}$ *(Figure 3) are denoted* t^s_{qp} *and* t^s_{pq}.

Remark In general, if $s > \max\{y(p), y(q)\}$, t^s_{pq} is the point on ∂D^s_{pq} with maximal y-coordinate and $LT(p, q, t^s_{pq})$.

For the rest of this section, we analyse a static situation where $s \in \Re$, $S = \{p_1, p_2, \cdots, p_m\}$, and $y(p_1) < y(p_2) < \cdots < y(p_m) < s$.

Lemma 3.1 *The graph* $G_s = (S, E_s)$ *where* $E_s = \{\{p_i, p_j\} \mid$ *there is a disk* D *such that* $Empty(D, \{p_1, p_2\}, s)\}$, *is connected and planar.*

Proof See Figure 4. For p_i, $1 < i \leq m$, let q_i be the point among $\{p_1, p_2, \cdots, p_{i-1}\}$ of minimal v-distance to p_i. Then $Empty(D^{vd}_{p_i q_i}, \{p_i, q_i\}, s)$ so from all points p_i in S, $1 < i \leq m$, there is an edge between p_i and a point p_j with $j < i$. Therefore, $G_s = (S, E_s)$ is connected.

For planarity we have to prove that if $\{p, p'\}$, $\{q, q'\} \in E_s$ then they can have only end points in common.

Let $D_{pp'}$ be a disk such that $Empty(D_{pp'}, \{p, p'\}, s)$ and let $D_{qq'}$ be defined

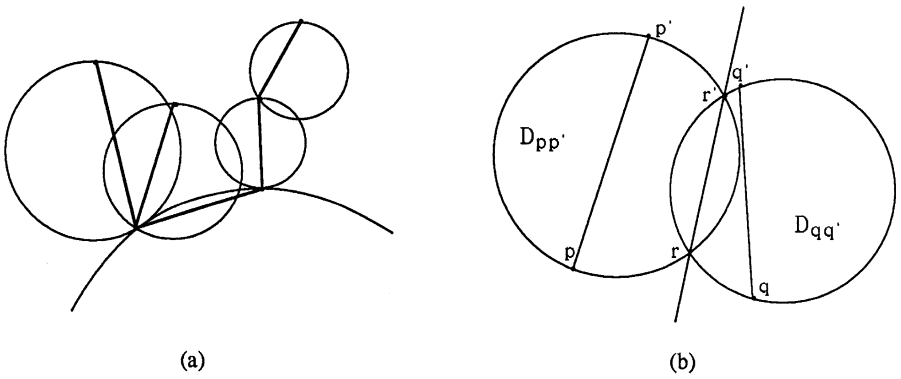

Figure 4: The connectedness and planarity of G_s (Lemma 3.1).

similarly. If $D_{pp'} \cap D_{qq'} = \emptyset$ then the statement holds since $\overline{rr'} \subset D_{rr'}$, for $r \in \{p, q\}$. Assume now that $\partial D_{pp'} \cap \partial D_{qq'}$ consists of two points r and r'. Then p and p' are situated on $\partial D_{pp'}$ outside $D_{qq'} \setminus \partial D_{qq'}$ and q and q' situated on $\partial D_{qq'}$ outside $D_{pp'} \setminus \partial D_{pp'}$. Hence the open line segments from p to p' and q to q' are separated by the line through r and r'. □

Assumption 3 *For no points $a, b, c \in S, y(t_{abc}) = s$.*

Lemma 3.2 *For $a, b \in S$, $Empty(D^s_{ab}, \{a, b\}, s)$ if and only if a and b are consecutive points on the outer region of G_s in clockwise order. (See Figure 6.)*

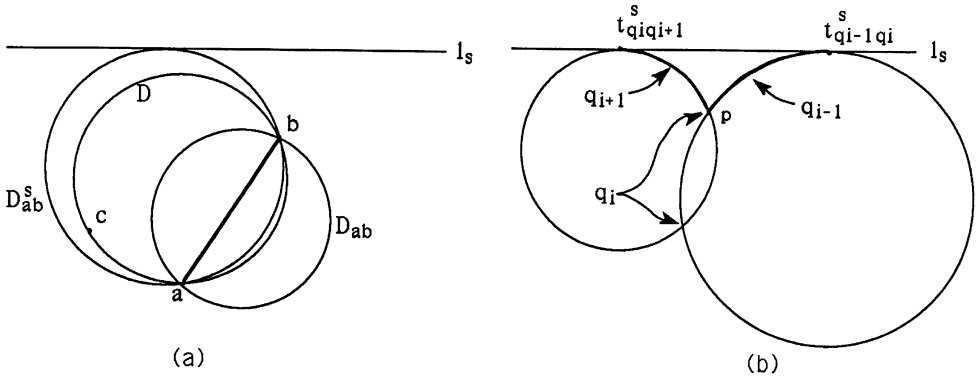

Figure 5: Disks involved in the proof of Lemmas 3.2 and 3.3.

Proof See Figure 5(a). For the *if* part, let a and b be consecutive points

on the outer region of G_s in clockwise order and D_{ab} a disk such that $Empty(D_{ab}, \{a, b\}, s)$. Such a disk exists since $\{a, b\}$ is in E_s. Now if D^s_{ab} contains other points from S than a and b, let D be a disk such that it contains a, b and a one more point c from S. Then since $c \in (D^s_{ab} \cap H_l(a, b)), (D \cap H_l(a, b)) \subset D^s_{ab}$ and $(D \cap H_r(a, b)) \subset D_{ab}$. Hence $Empty(D, \{a, b, c\}, s)$ and $\{a, c\}$ and $\{b, c\}$ are in E_s in contradiction with a and b being consecutive points on the outer region of G_s in clockwise order.

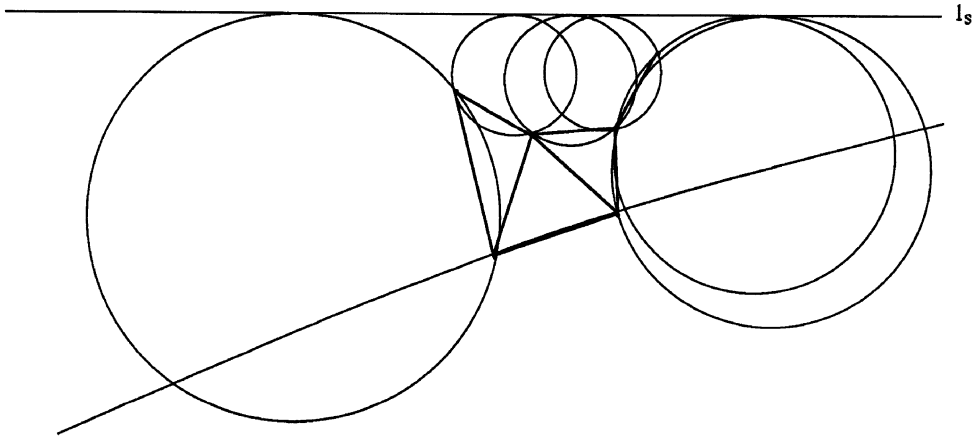

Figure 6: The disks D^s_{ab} (Lemma 3.2).

With Assumption 3 the *only if* part is obvious. □

Let $p_1 = q_1, q_2, \cdots, q_k = p_1$ be the points on the boundary of the outer region in clockwise order. Note that there can be several instances of points from S in the sequence.

Lemma 3.3 $x(t^s_{q_{i-1}q_i}) < x(t^s_{q_iq_{i+1}})$ *for* $1 < i < k$.

Proof By Assumption 3, $x(t^s_{q_{i-1}q_i}) \neq x(t^s_{q_iq_{i+1}})$. Assume that $x(t^s_{q_iq_{i+1}}) < x(t^s_{q_{i-1}q_i})$. Let p be the point on $\partial D^s_{q_iq_{i+1}} \cap \partial D^s_{q_{i-1}q_i}$ with maximal y-coordinate. Such a p exists since $q_i \in \partial D^s_{q_iq_{i+1}} \cap \partial D^s_{q_{i-1}q_i}$. (See Figure 5(b))

By the remark following Definition 3.3, we have that $t^s_{q_iq_{i+1}}$ is the top point of $\partial D^s_{q_iq_{i+1}}$ and $LT(q_i, q_{i+1}, t^s_{q_iq_{i+1}})$. By Lemma 3.2 we have that $Empty(D^s_{q_{i-1}q_i}, \{q_{i-1}, q_i\}, s)$. Therefore q_{i+1} must be on the part of $\partial D^s_{q_iq_{i+1}}$ between p and $t^s_{q_iq_{i+1}}$ (in anti clockwise order). See Figure 5. Similarly it follows that q_{i-1}

must be on the part of $\partial D^s_{q_{i-1}q_i}$ between p and $t^s_{q_{i-1}q_i}$ (in clockwise order). It follows that $x(q_{i+1}) < x(q_{i-1}), y(q_{i+1}) > y(q_i)$. Since $i \notin \{1, k\}, q_{i-1}, q_i$ and q_{i+1} cannot occur as consecutive points on the boundary of the outer region of G_s in clockwise order. □

Lemma 3.4 *Let $p, q \in \Re^2$ and $s > \max\{y(p), y(q)\}$. $r \in l_s$ is then to the left of t^s_{pq} on l_s ($x(r) < x(t^s_{pq})$) if and only if $LO(p, q, r)$, where*
$LO(p, q, r) \equiv$

$$[(y(p) < y(q)) \wedge LT(p, q, r) \wedge (v - d(r, p) < v - d(r, q))] \vee$$
$$[(y(p) > y(q)) \wedge (RT(p, q, r) \vee (v - d(r, p) < v - d(r, q)))]$$

Proof Straightforward observation (Figure 3).

The preceding Lemmas demonstrate that if the sequence $q_1, q_2, \cdots, q_k$ on the boundary of the outer region of G_s is also organized in a balanced tree scheme, then for a point r on l_s we can find in time $O(\log n)$ the instance q_i of a point such that $x(t^s_{q_{i-1}q_i}) < x(r) < x(t^s_{q_iq_{i+1}})$ which implies that $v - d(r, q_i)$ is minimal over $\{q_1, q_2, \cdots, q_k\}$.

We now turn to the problem of identifying pairs of points $\{p, q\}$ in S for with there is a disk D such that $empty(D, \{p, q\}, \infty)$ but for no disk D, $Empty(D, \{p, q\}, s)$. The reason why $Empty(D, \{p, q\}, s)$ does not hold for disks with p and q on the boundary is the presence of the other points from S. Therefore we shall be looking for triples of points $\{p, q, r\}$ in S such that $Empty(D_{pqr}, \{p, q, r\}, \infty)$ bit not $Empty(D_{pqr}, \{p, q, r\}, s)$.
The set of those triples is denoted $Triples_s$ below. It turns out that the triple of points from S where the corresponding disk has minimal y-coordinate for the top point is to be found among triples of consecutive points on the outer region ($BTRIPLES_s$ below). This is the subject of Lemma 3.5.

$$Triples_s = \{(a, b, c) \in S^3 \mid Empty(D_{abc}, \{a, b, c\}, \infty) \; and |\partial D_{abc} \cap l_s| = 2\}$$

$$BTRIPLES_s = \{(q_{i-1}, q_i, q_{i+1}) | 1 < i < k \text{ and } LT(q_{i-1}, q_i, q_{i+1})\}$$

Lemma 3.5 *If (a, b, c) minimizes $y(t_{abc})$ over $Triples_s$ and (q_{i-1}, q_i, q_{i+1}) minimizes $y(t_{q_{i-1}q_iq_{i+1}})$ over $BTRIPLES_s$ then $\{a, b, c\} = \{q_{i-1}, q_i, q_{i+1}\}$.*

Proof Assume first that $\{a, b, c\}$ minimizes $y(t_{abc})$ over $Triples_s$. Assume wlog that a, b and c occur in anti clockwise order on ∂D_{abc} (see Figure 7(a)).

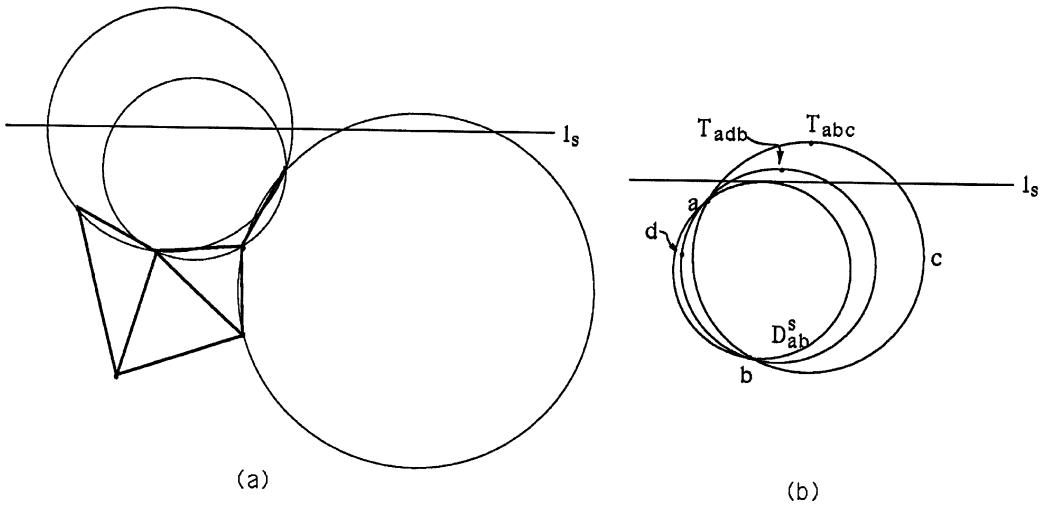

Figure 7: (a) Disks corresponding to $BTRIPLES_s$. (b) Disks involved in the proof of Lemma 3.5.

We show that $Empty(D^s_{ab}, \{a, b\}, s)$. This follows since $S \cap (D_{abc} \setminus \partial D_{abc}) = \emptyset$ and if $D^s_{ab} \cap (S \setminus D_{abc}) \neq \emptyset$ then for a $d \in D^s_{ab} \cap (S \setminus D_{abc})$, $\{a, d, b\} \in Triples_s$ and $y(t_{abd}) < y(t_{abc})$ contradicting the minimality of $y(t_{abc})$. Thus $\{a, b\}$ is in E_s. Similarly for $\{b, c\}$. Therefore it follows by Lemma 3.2 that a, b and c are consecutive points on the outer region of G_s. $LT(a, b, c)$ since a, b and c occur on ∂D_{abc} in anti clockwise order so we conclude that $(a, b, c) \in BTRIPLES_s$. If (q_{i-1}, q_i, q_{i+1}) minimizes $y(t_{q_{i-1}q_iq_{i+1}})$ over $BTRIPLES_s$ then by a completely analogous argument we get that $Empty(t_{q_{i-1}q_iq_{i+1}}, \{q_{i-1}, q_i, q_{i+1}\}, \infty)$ such that $(q_{i-1}, q_i, q_{i+1}) \in Triples_s$ from which the Lemma follows. □

4 The Algorithm

The time requirements in what follows are dependent on the time for computing various quantities listed below. The numerical computations involved are not dealt with here.

Assumption 4

1. *For three distinct non-linear points $p, q, r \in \Re^2$, $y(t_{pqr})$ can be computed*

in constant time.

2. *For two points $p, q \in \Re^2$, $v - d(p, q)$ can be computed in constant time.*

The algorithm uses three data structures (and pointers between them) supporting various operations:

For $s \in \Re$, let $S_s = \{p \in S \mid y(p) < s\}$ and $G_s = (S_s, E_s)$.

- *GRAPH* contains the straight line embedding of the graph G_s. AddEdge(p, q) adds edge $\{p, q\}$ to *GRAPH* in constant time. Note, that only edges on the the boundary are added. p might be a new point.
- *BOUNDARY* is a structure over the points on the outer region of G_s. It supports the following operations: (note that there can be several instances of the same point from S in *BOUnDARY*)
  - before[q] and next[q], which for an instance q on the outer region give in constant time the instances before and after q in clockwise order.
  - ClosestPointTo(p_i), which for a new point p_i in time O(log n) finds the instance q of the closest point among S_s w.r.t. the v-distance, such that $x(t^{y(p_i}_{\text{before}[q]q}) < x(p_i) < x(t^{y(p_i}_{q\ \text{next}[q]q})$. This is possible because of Lemmas 3.1 through 3.4
  - InsertNewOnBoundary(p, q), which in time O(log n) inserts a new point p by replacing instance q with q, p, q (adding the edge $\{p, q\}$).
  - UpdateOnBoundary(q), which in time O(log n) removes q (adding the edge {before[q], next[q]}).
- *TRIPLES* is a structure over points q on the outer region of G_s for which LT(before[q], q, next[q]), supporting the operations: (like above there can be several instances of the same point from S in *TRIPLES*)
  - MinTop, which in constant time finds the minimal y-coordinate of any top point $t_{\text{before}[q]\ q\ \text{next}[q]}$ of instances q in *TRIPLES*. If there are no points in *TRIPLES*, then MinTop $= \infty$.
  - GetPointCorrToMinTop, which in constant time finds the instance q in *TRIPLES* corresponding to MinTop.

– DeleteFromTriples(q), which in time O($\log n$) deletes q from *TRIPLES* (if it is there).
– InsertInTriples(q), which in time O($\log n$) inserts q in *TRIPLES*.

There are two kinds of event points.
The first is *point events*, $\{y(p) \mid p \in S\}$. We assume that the points in $S = \{p_1, p_2, \cdots, p_n\}$ are sorted in increasing y-coordinates.
The second is *top point events*, y-coordinates for top points of the disks corresponding to points in *TRIPLES*. These event points are exactly points where Assumption 3 is violated so this assumption is not supposed to hold. If a point event and a top point event coincide then the top point event is handled first.

Apart from the above listed operations three procedures are used:

Initialize(p) sets up the structures for one point p.

Add(p)To(q) adds a new point p to the structures by adding the edge $\{p, q\}$ (q is on the boundary). See Figure 8.

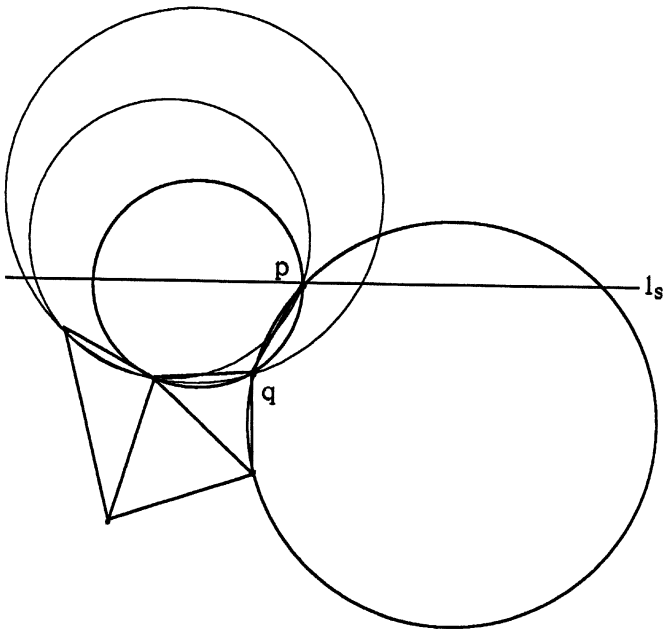

Figure 8: Adding a new point to the structure.

```
Add(p)To(q):
    AddEdge(p,q);
    InsertNewOnBoundary(p,q);
    DeleteFromTriples(q);
    if LT(before[before[p]], before[p], p) then
        InsertInTriples(before[p]);
    if LT(p,next [p], next [next [p]]) then
        InsertInTriples(next[p]);
```

Note, that before[p] and next[p] above are different instances of q.

Update(q). q is on the boundary and the edge {before[q], next[q]} is added to the structure because $y(t_{\text{before[q]qnext[q]}}) = s$. See Figure 9.

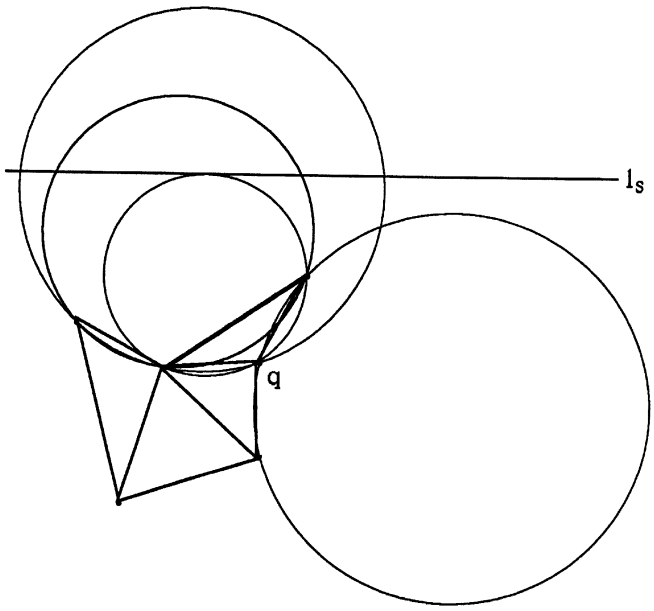

Figure 9: Adding a new edge on the boundary.

```
Update(q);
bq :=before[qj; nq :=next[q];
AddEdge(bq, nq);
UpdateOnBoundary(q);
DeleteFromTriples(bq);
```

```
DeleteFromTriples(q);
DeleteFromTriples(nq);
if LT(before[bq], bq, nq) then
    InsertInTriples(bq);
if LT(bq, nq, next[nq]) then
    Insert InTriples(nq);
```

The algorithm is not explicitly presented as event driven, but in lines 9 and 10 point events are handled while top point events are handled in lines 5, 6, 13 and 14.:

Algorithm Delaunay

```
 0:  Initialize(p_1);
 1:  Adq(p_2)To(p_1);
 2:  for i := 3 to n do
 3:      if MinTop ≤ y(p_i) then
 4:        repeat
 5:            q := GetPointCorrToMinTop;
 6:            Update(q)
 7:        until MinTop > y(p_i);
 8:      fi;
 9:      q := ClosestPointTo(p_i);
10:      Add(p_i)To(q);
11:  od;
12:  repeat
13:      q := GetPointCorrToMinTop;
14:      update(q)
15:  until MinTop = ∞
```

The complexity of the algorithm, is with Assumption 4, easily seen to be $O(n \log n)$.
Correctness is an easy exercise using invariant techniques. The appropriate invariants between lines l and $l+1$ are for a small enough ϵ:

$0, 1$: $G_{y(p_1)+\epsilon}$ is constructed.
$1, 2$: $G_{y(p_2)+\epsilon}$ is constructed.
$2, 3$: $G_{y(p_{i-1})+\epsilon}$ is constructed.
$3, 4$: $G_{MinTop-\epsilon}$ is constructed.
$4, 5$: $G_{MinTop-\epsilon}$ is constructed.
$5, 6$: $G_{y(t_{before[q]qnext[q]})-\epsilon}$ is constructed.
$6, 7$: $G_{MinTop-\epsilon}$ is constructed.
$7, 8$: $G_{y(p_i)-\epsilon}$ is constructed.
$8, 9$: $G_{y(p_i)-\epsilon}$ is constructed.
$9, 10$: $G_{y(p_i)-\epsilon}$ is constructed.
and $Empty(D^{vd}_{p_iq}, \{p_i, q\}, y(p_i))$.
$10, 11$: $G_{y(p_i)+\epsilon}$ is constructed.
$11, 12$: $G_{y(p_n)+\epsilon}$ is constructed.
$12, 13$: $G_{MinTop-\epsilon}$ is constructed.
$13, 14$: $G_{y(t_{before[q]qnext[q]})-\epsilon}$ is constructed.
$14, 15$: $G_{MinTop-\epsilon}$ is constructed.
15 : G_∞ is constructed.

To construct the generalized Voronoi diagram for S we only have to change the procedures Add($\cdot$)Edge($\cdot$) and Update($\cdot$) slightly.
For the construction of Delaunay triangulations, only the shape of the disks is of importance, not where the centres are situated. To be able to construct the Voronoi diagram we need to know the centres and the bisecting curves derived from the distance function. (See Figure 10.)

Observe that $c_{\text{before}[q]q\text{next}[q]}$ where q is an argument to Update in lines 6 and 14 is a Voronoi node and that all Voronoi nodes are met that way. If we just represent bisecting curves by giving their end points and the two sites the bisect, then we only have to be able to compute c_{pqr} in constant time, given sites p, q and r, to turn algorithm *Delaunay* into an algorithm constructing a generalized Voronoi diagram in time O(n log n).

Conclusions

We have presented a very general sweepline algorithm for construction of generalized Delaunay triangulations and generalized Voronoi diagrams without

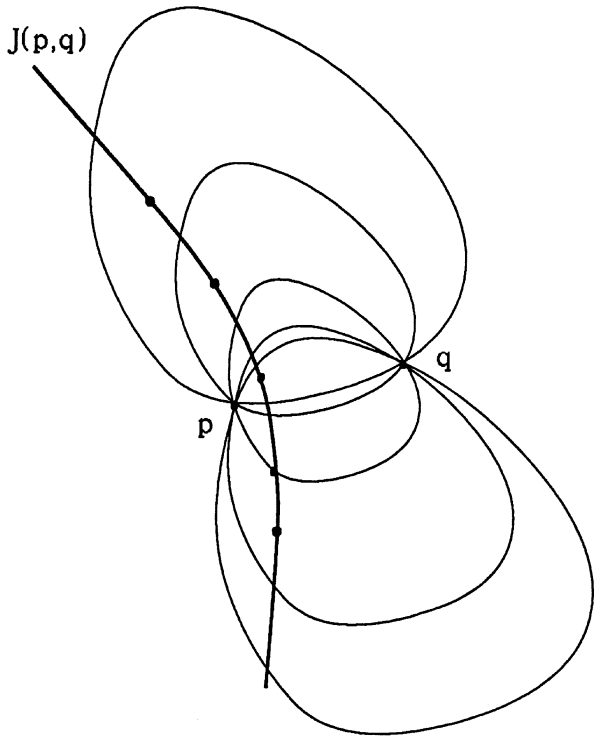

Figure 10: An example of a bisecting curve.

using any transformations. Although the analysis might not seem simple, the algorithm is so and is derived in a natural way from the sweepline paradigm together with greediness.

The reason for introducing the transformation * in Fortune's paper [4] was to prevent updating of the structure to take place below the sweepline. There is no need for that as long as we can handle the events and operations in the right order as demonstrated in this paper. If we apply our algorithm to Euclidean disks and move the centres to the top points then the constructed Voronoi diagram is exactly the transformed diagram in Fortune's paper. Moving the centre to the top point of the disk precisely prevents new Voronoi nodes to be added below the sweepline.
If wanted, the centre could be moved even outside the disks and we could still construct a corresponding Voronoi diagram.

We can also drop the requirement that disks should be strictly convex and

smooth. Convexity is enough. Uniqueness and existence of circles through three non-colinear points are the problem. The problem with uniqueness can be overcome by approximating the disks by strictly convex ones. The problem with non-existence implies that the Delaunay triangulation might not be a triangulation of the convex hull of S. For instance L_∞ disks can be approximated by L_p disks for $p \to \infty$. This enables us to define a canonical L_∞ disk through three given points if it exists. This suffices for the algorithm to be applicable.

Finally the method can be generalized to higher dimensions. This will be the topic of a forthcoming paper.

References

[1] Aurenhammer, F. *Voronoi Diagrams - A survey* Tech. Rep. 263, Institute for Information Processing, Graz Technical University (1988)

[2] Chew, L.P. & Drysdale III, R.L.(Scot) *Voronoi Diagrams based on Convex Distance Functions* Proc. of Computational Geometry (1985) 1, 235–244

[3] Drysdale III, R.L.(Scot) *A Practical Algorithm for Computing the Delaunay Triangulation for Convex Distance Functions* Proc. of Discrete Algorithms (1990) 1, 159–168

[4] Fortune, S. *A Sweepline Algorithm for Voronoi Diagrams* Algoritmica (1987) 2, 153–174

[5] Klein, R. *Concrete and Abstract Voronoi Diagrams* Lecture Notes in Computer Science, Vol 400, Springer Verlag, Berlin

[6] Klein, R. & Mehlhorn, K. & Meiser, St *On the Construction of Abstract Voronoi Diagrams, II* Proc. SIGAL Symp. on Algorithms, Tokyo (1990). Lecture Notes in Computer Science, Vol. 450, Springer Verlag, Berlin

[7] Lay, S. R. *Convex Sets and their Applications* Wiley, New York (1972)

[8] Lee, D.T. *Two-Dimensional Voronoi Diagrams in the L_p-Metric* JACM (1980) 27, 604–618

[9] Lee, D.T. & Drysdale R.L. *Generalizations of Voronoi diagrams in the plane* Siam J. Comput., (1981) 10, 73–87

[10] Matoušek, J. & Seidel, R. & Welzl, E. *How to Net a Lot with Little: Small ϵ-Nets for Disks and Halfspaces* Proc. of Computational Geometry (1990) 6, 16–22

[11] Mehlhorn, K. & Meiser, St. & Ó'Dúnlaing *On the Construction of Abstract Voronoi Diagrams* Discrete and Computational Geometry (1991) 6, 211–224

[12] Ó'Dúnlaing, C. & Sharir, M. & Yap, C.K. *Retraction: A new approach to Motion-Planning* STOC (1983) 15, 207–220

[13] Sharir, M. *Intersection and closest-pair problems* Siam J. Comput., (1985) 14, 448–468

[14] Shute, G.M. & Deneen, L.L. & Thomborson, C.D. *An $O(n \log n)$ Plane-Sweep Algorithm for L_1 and L_∞ Delaunay Triangulation* Algoritmica (1991) 6, 207–221