F. Dehne and R. Klein, "A sweepcircle algorithm for Voronoi diagrams," in Proc. *International Workshop on Graphtheoretic Concepts in Computer Science (WG'87)*, Erlangen (W.-Germany), 1987, Springer Verlag, Lecture Notes in Computer Science, Vol. 314, pp. 59-70.

A SWEEPCIRCLE ALGORITHM FOR VORONOI DIAGRAMS

Extended Abstract

Frank Dehne[1] , Rolf Klein[2]

Abstract
The Voronoi diagram of n sites on the surface of a cone has a combinatorial structure rather different from the planar one. We present a sweepcircle algorithm that enables its computation within optimal time O(n log n), using linear storage.

Keywords: computational geometry, cone, shortest path, sweepline algorithm, sweepcircle algorithm, Voronoi diagrams

1. INTRODUCTION

The Voronoi diagram of n sites p_i on a curved surface in space is a map whose regions $R(p_i)$ are in one-to-one correspondance to the sites; the region $R(p_i)$ contains all those points of the surface that are closer to p_i than to any other site p_j.
If the surface is a plane, the Voronoi regions are well known to be convex and simply-connected. Furthermore, the common boundary between two regions is always a connected set. However, these properties do not hold for general curved surfaces.

In the next section we determine the structure of Voronoi diagrams on the *surface of a cone*. We show that the bisector of two points can be a *closed curve* if the cone's slope is large enough. Thus, the Voronoi regions can encircle each other. This makes it hard to apply a divide-and-conquer algorithm, since for the merge step no method is known for efficiently finding starting points of the closed loops, the bisector of two sets possibly consists of (compare Chew, Drysdale [1]).

In section 3 we introduce a *sweepcircle algorithm* for the computation of Voronoi diagram. Instead of sweeping a line upward across the plane as suggested by Fortune [2] we use a circle emanating from the cone's peak for computing a certain transformation of the Voronoi diagram. We first consider the special case when the cone is flat, i.e., a plane. Here the sweepcircle technique yields an optimal O(n log n) algorithm, as was independently observed by Thurston [5].

---

1) School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6. Research supported by NSERC Grant No. A9173

2) Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Univ. Karlsruhe, Postfach 6980, D-7500 Karlsruhe, W.-Germany.

Then, in section 4, we show that the sweepcircle algorithm can be generalized to work on a "proper" cone, too. Again, the Voronoi diagram of n points can be computed within optimal O(n log n) time, using linear storage.

2. VORONOI DIAGRAMS ON THE SURFACE OF A CONE

The *distance* d(p,q) between two points, p and q, on the surface of a cone is the minimum length -as taken in 3-space - of all curves on the cone's surface that connect p with q.

For a point p on the cone (different from the peak 0) let *L(p)* denote the ray from 0 through p and let *L'(p)* be the intersection of the vertical plane through 0 and p with the surface of the cone minus L(p), i.e. the ray "opposite" to L(p). These rays divide the surface into two sectors, SL(p) and SR(p); see Figure 1.

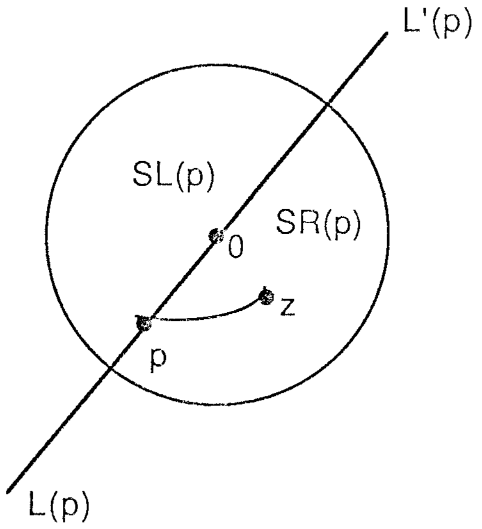


Figure 1

Clearly, each shortest path from p to a point z in SR(p) runs entirely in SR(p); only for points in L'(p) there is a symmetric shortest path running in SL(p).
If we are given a curve on a cone and a cut from the peak of the cone to infinity that doesn't intersect the curve, then slicing the cone along this cut and unfolding it doesn't change the length of the curve. Thus, a shortest path on the cone must become a straight line segment in the plane; see Figure 2. We call these shortest paths *straight curve segments*.

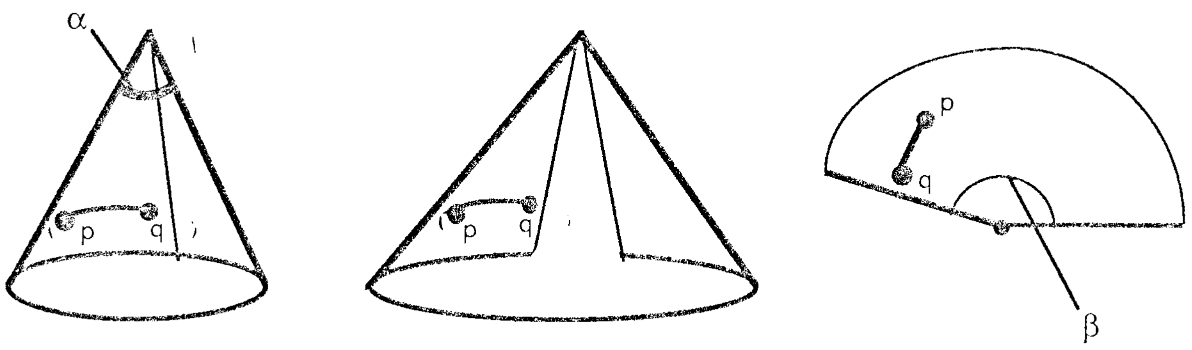


Figure 2

Now we have to investigate the *bisectors* $B(p,q)=\{z\in \text{cone's surface};\ d(p,z)=d(q,z)\}$ of two points, the constituent parts of the Voronoi diagram.

Lemma 2.1
A bisector between two points on a cone consists of one or two straight curve segments.

Proof: Assume that p and q are as shown in Figure 3. No shortest path from p or q to a point in $A = [SL(p)\cap SL(q)] \cup [SL(p)\cap SR(q)] \cup [SR(p)\cap SR(q)]$ intesects the shaded area $A' = SR(p)\cap SL(q)$. Hence, slicing the cone along a cut in A' and unfolding it doesn't affect these straight curves.

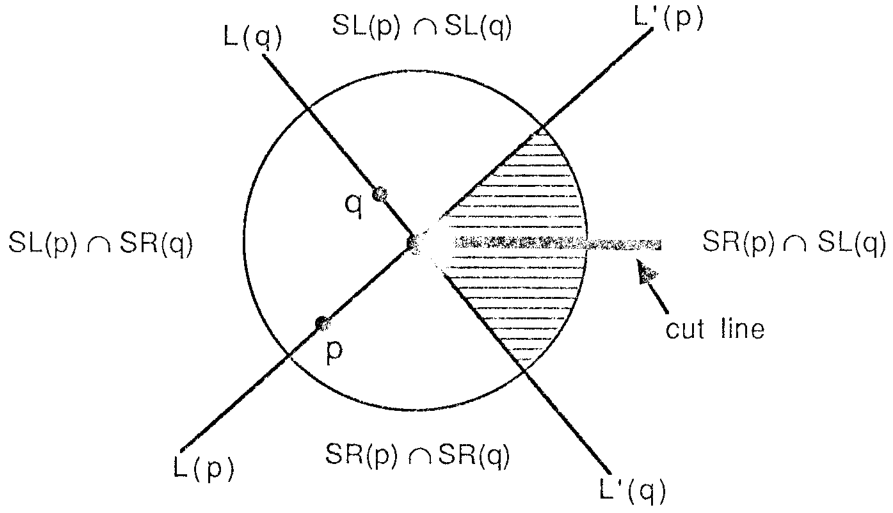


Figure 3

Thus, the piece of B(p,q) running in A becomes the usual Euclidean bisector after unfolding the cone. In Figure 4 the ray L'(p) was chosen as the cut line in A'.

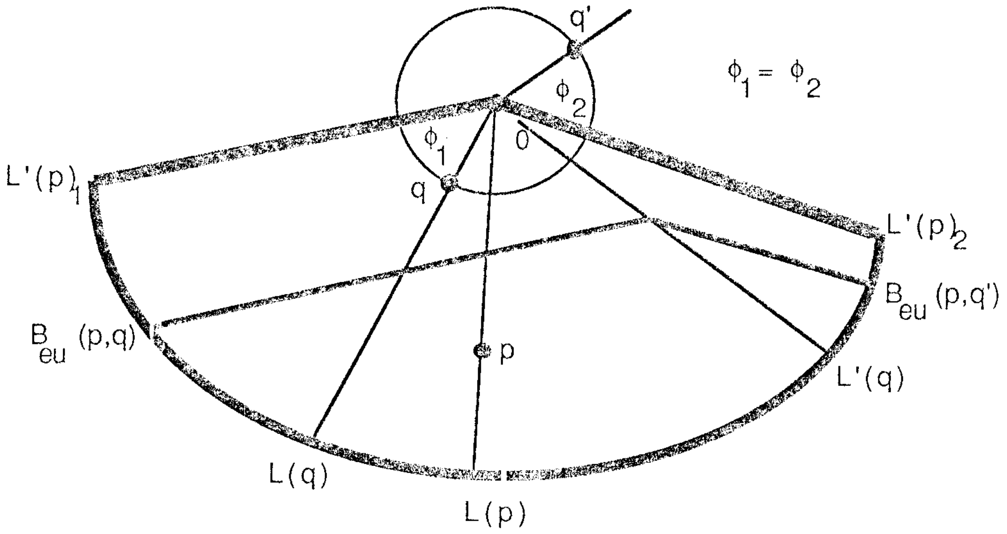


Figure 4

For dete[illegible]ining the part of the bisector B(p,q) in the area A' consider cutting and unfolding the cone with respect to L(q). In Figure 4 this is equivalent to rotating the sector between L(q) and $L'(p)_1$ clockwise such that $L'(p)_1$ and $L'(p)_2$ coincide. This rotation maps q into the point q'. Now, for every point in A' the shortest paths to p and q' do not cross the new cut line L(q); thus the unfolded bisector in this region is the Euclidean bisector betwen p and q'.
Note that one the intersections can be empty. In this case B(p,q) consists of one straight line segment only, see Figure 5. ◊

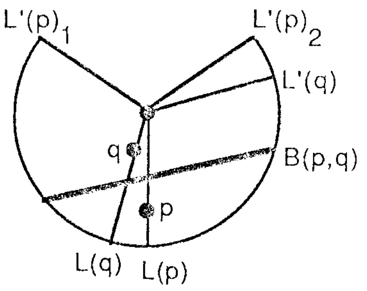


Figure 5

The next Lemma shows that the Voronoi diagram on a cone can have a structure different from Voronoi diagrams in the plane.

Lemma 2.2
A bisector between two points on a cone can be a closed curve if and only if the top angle of the cone is less than 60^o.

Sketch of Proof: Given a cone with top angle α, cutting and unfolding the cone creates a planar slice with angle β as depicted in Figure 6. It is easy to observe that α is less than 60^o if and only if β is less than 180^o. If β is smaller than 180^o than a closed bisector can exist, see Figure 6. On the other hand, Figure 5 indicates that a bisector consisting of one segment cannot intersect both boundaries created by the cut line if β is greater than 180^o. For a bisector which consist of two segments the proof is more involved and given in the full paper [2]. ◊

Each closed bisector circumscribes the peak of the cone and has a unique outermost point (the point v in figure 6). The existance of closed bisectors implies that Voronoi regions can encircle one another. Thus, they are not simply connected. The common boundary of two regions can consist of several disconnected pieces (see Figure 14). In general, the Voronoi regions $R(p_i)$ are not convex but they are *star-shaped* with respect to p_i, i.e. for each z in $R(p_i)$ all points on a shortest path connecting p_i and z are also contained in $R(p_i)$; see Klein/Wood [4] for a general theorem on this topic. Hence, the Voronoi regions are *connected*. Therefore, the Voronoi diagram V of n points on a cone is a *planar graph* with *n faces*. Besides the vertices of degree $d \geq 3$ in V (common boundary points of d regions) we call the outermost points of closed bisectors which are present in V *vertices of degree 2*.

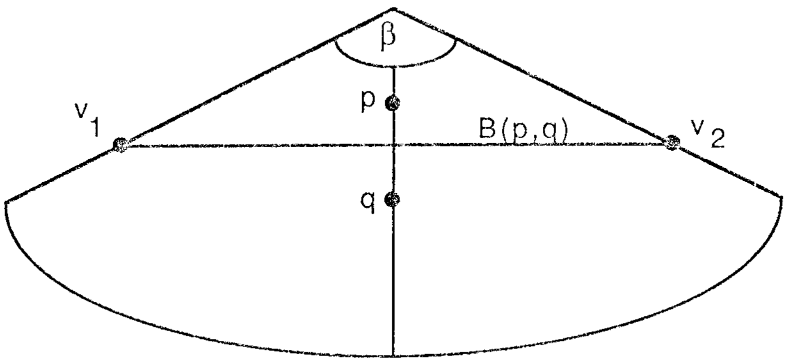


Figure 6

Theorem 2.3
The Voronoi diagram of n points on the surface of a cone is a planar graph with n faces and O(n) edges and vertices.

Proof : A vertex of degree 2 is either the origin of a cyclic edge or it has two outgoing edges leading to vertices of degree ≥3. Thus, the Theorem follows from Euler's formula. ◊

3. THE SWEEPCIRCLE ALGORITHM FOR POINTS IN THE PLANE

The problem in computing the Voronoi diagram by a swep technique is how to determine the halting points correctly.
In the situation displayed in Figure 7(a) the next halting point for the sweepline which is moving upward should be the vertex v. In this situation, however, we don't even know that vertex v exists because we haven't detected the site p, yet. This difficulty has been overcome in Fortune [3] by *bending* the Voronoi diagram in such a way that (property Δ):

- its combinatorial structure is preserved
- each site becomes the bottommost point of its region
- if a vertex isn't a site then all but one of its incident edges extend downward.

These properties enable the correct computation of the halting points. During the bottom-up line sweep a "list" (usually a balanced tree) containing the regions and bisectors currently intersected by the sweepline is maintained. Intersections of the adjacent bisectors are computed as possible halting points. This yields an O(n log n) algorithm because the swepline stops at most O(n) times and because at most O(n) objects are intersected at any time.

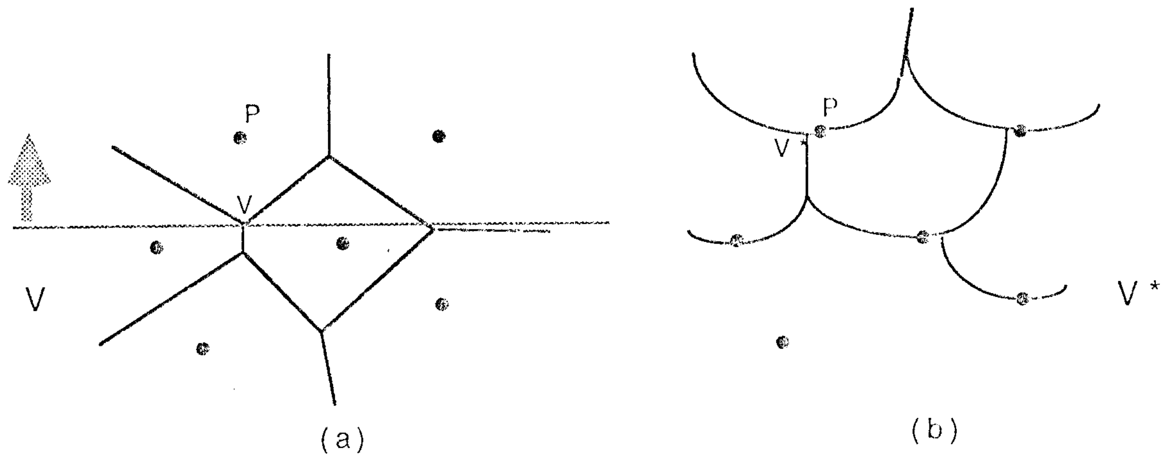


Figure 7

The bending transformation introduced by Fortune [3] is the following:
$z=(x,y) \to z^*=(x,y+d(z))$, where $d(z)=\min\{|z-p_i|;\ 1\le i\le n\}$ denotes the distance between z and its closest site. Figure 7(b) shows the transformed Voronoi diagram V^* (all sites are left fixed by *).
For computing the original Voronoi diagram V, the algorithm simply uses the untransformed bisectors and evaluates the *-values whenever necessary

In order to apply a *sweepcircle* technique, we have to bend the Voronoi diagram outward with respect to the peak 0 (of the flat cone). The transformation is:
$z=(\varphi,r) \to z^*=(\varphi,r+d(z))$, where (φ,r) are the *polar coordinates* of a point z; see Figure 8.

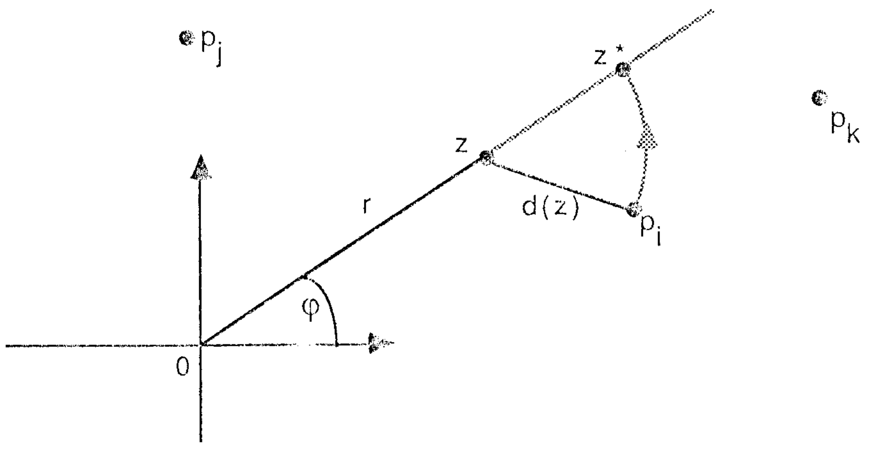


Figure 8

This mapping * is not defined for z=0. For the time being we assume that no bisector of two sites passes through 0, since otherwise the mapped bisector has a gap. We will discuss this case later on.
Obviously, $p_i^*=p_i$ for all sites p_i. Let $*_i$ be defined by $*_i(z)=(\varphi,r+|z-p_i|)$ where $z=(\varphi,r)$. Then $* = *_i$ on $R(p_i)$.

Lemma 3.1
Let $|p_i|>|p_j|$, then $*_i(B(p_i,p_j))$ passes through p_i and is strictly emanating outward in both directions from p_i.

Proof: In Figure 9 we have $C+A<D+B$ because it is easy to observe that y cannot be contained in the ellipse $E=\{z;\ |z|+|p_i-z|=C+A\}$.◊

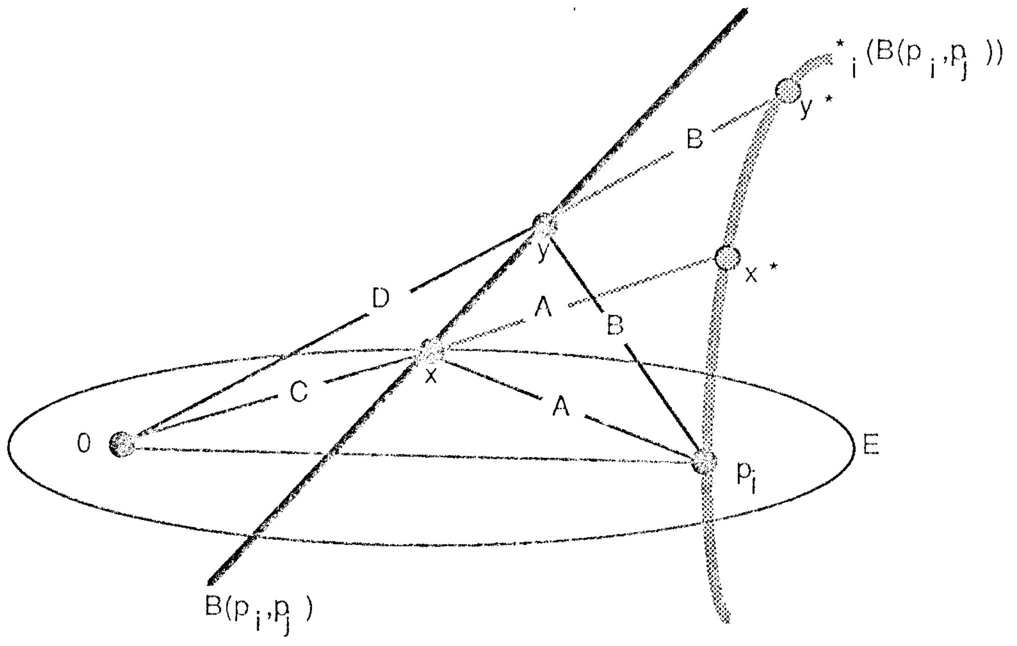


Figure 9

Again, properties equivalent to those in (Δ) hold.

Lemma 3.2
i) V* has the same combinatorial structure as V.
ii) Each site becomes the unique innermost point of its region.
iii) If a vertex of V* is not a site then all but one of its incident edges are emanating inward.

The proof is analogous to the sweepline case (see [3]). Figure 10 shows a Voronoi diagram and its bent image under *.
Now, a *sweepcircle* expanding from the origin can be used to compute V*, since sites are the first points of their regions to be hit and all vertices are detected due to Lemma 3.2. The list of objects intersected by the sweepcircle must be *cyclic* now but can still be implemented as a balanced tree.
If bisectors contain 0 then the respective sites are closest to 0 (and of equal distance); hence, they are detected first by the sweep circle and this special case can easily be solved by closing the gaps in the transformed bisectors.

Theorem 3.3
The sweepcircle algorithm computes the Voronoi diagram of n points in the plane within time O(n log n), using linear storage.

This observation was independently made by Thurston [4]. He also noted the following *advantage* over the sweepline algorithm. If the set of vertices is very large (or even infinite), then the sweepcircle technique allows to compute the Voronoi diagram *locally*, stopping as soon as the circle has become large enough.
This could be advantageous in an application where the computation of Voronoi diagrams has to be combined with a windowing mechanism, e.g. for large scale geographic systems.

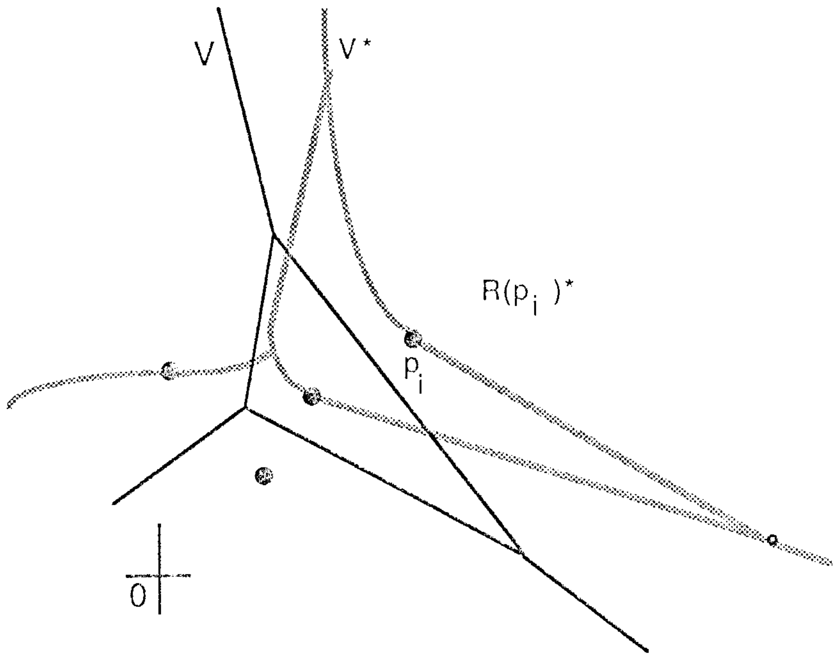


Figure 10

In the next section we show that the sweepline algorithm can be generalized to compute the Voronoi diagram on the surface of a cone.

4. THE SWEEPCIRCLE ALGORITHM FOR POINTS ON A CONE

Once a radial ray on the cone's surface is fixed we have a system of polar coordinates with the origin at the cone's peak, and the mapping * can be defined as in the planar case; see Figure 11.

Observation:
Let z be a point on the cone such that the shortest path from z to its nearest neighbor p_i is not intersected by some cut line l, then z* can be computed by cutting and unfolding the cone along l, applying the mapping * in the plane, and refolding the cone.

This observation allows to apply our previous results.

Lemma 4.1
Given two points p_i and p_j such that p_j is closer to 0 than p_i, then the image $*_i(B(p_i,p_j))$ of $B(p_i,p_j)$ contains p_i, and the distance of the points on $*_i(B(p_i,p_j))$ to 0 is monotonically increasing in both directions from p_i (up to the maximum, if $B(p_i,p_j)$ is closed).

Proof: Lemma 2.1, Lemma 3.1, and the above observation.

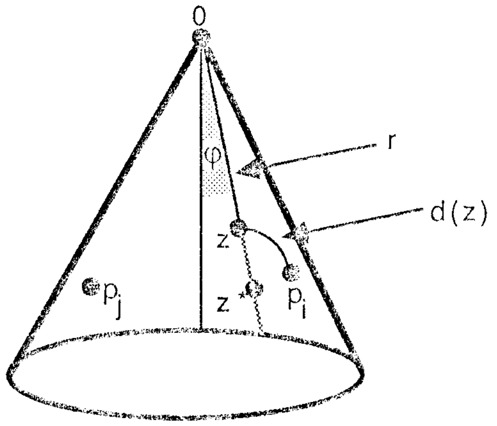


Figure 11

The counterpart to Lemma 3.2 can be reduced to the planar case, too. Only property iii) needs to be modified: A 2-vertex (outermost point of a closed bisector) has no edge emanating outward.

Theorem 4.2
The Voronoi diagram of n points on the surface of the cone can be computed in time O(n log n), using linear storage.

Sketch of proof: As long as the angle α at the cone's peak is greater than or equal to 60^O the situation is quite the same as in the plane, except that bisectors can consist of two straight curve segments (Lemma 2.1); this can be handled as part of the description of bisectors.
If α is less than 60^O we have to generalize the algorithm to handle closed bisectors (Lemma 2.2). This is accomplished by treating the two branches of $*_i(B(p_i,p_j))$ extending from p_i (depicted in Figure 12) as if they were adjacent pieces of different bisectors. Their intersection, the 2-vertex q^*, is computed as a halting point, but no new edge is created when q^* is encountered.
Figure 13 shows a snapshot of a sweepcircle algorithm stage, as seen from top of the cone. C is the cyclic list of the active (=intersected) regions (R_i) and bisector branches ($B_{i,j}$) in sorted φ-order. Q is the event queue of halting points in r-order containing sites and intersections of active bisector branches that are neighbors in C. Figure 14 displays the final Voronoi diagram.◊

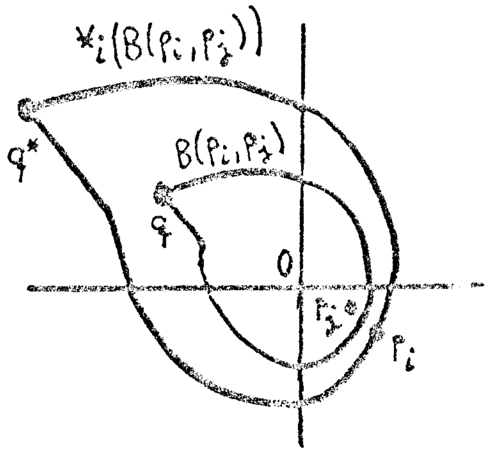


Figure 12
(Projection on the plane perpendicular to the main axis of the cone)

5. CONCLUDING REMARKS

We have discussed the structure of Voronoi diagrams of sites on a cone (=mountain) and shown how to compute them in optimal time. This is one step towards our goal of computing shortest paths and Voronoi diagrams for sets of sites on general curved surfaces (=natural landscapes).

To our knowledge the only other curved surface for which an algorithm for computing the Voronoi diagram has been considered is the surface of a sphere [5].

REFERENCES

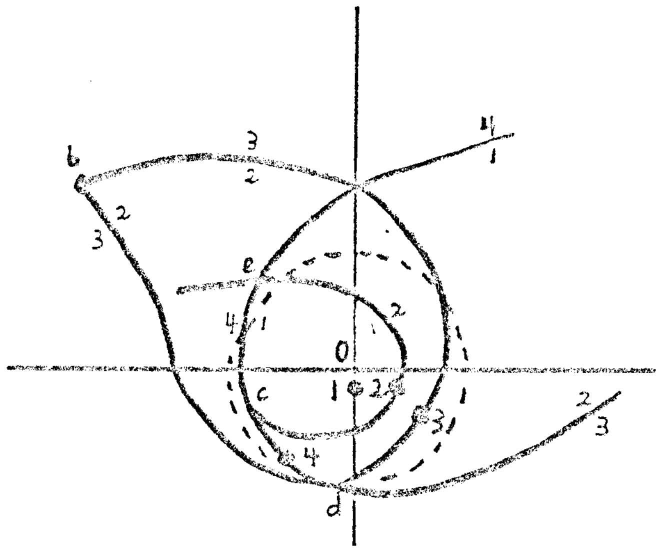
[1] L.P.Chew and R.L.Drysdale III, "Voronoi diagrams based on convex distance functions", Proc. 1st ACM Symposium on Computational Geometry, Baltimore, MD, 1985, pp. 235-244

[2] F.Dehne, R.Klein, "A sweepcircle algorithm for Voronoi diagrams on cones", Tech.Rep., School of Computer Science, Carleton University, Ottawa, Canada K1S5B6, 1987

[3] S.Fortune, "A sweeline algorithm for Voronoi diagrams", Algorithmica, vol. 2, No.2, 1987, pp.153-174

[4] R.Klein and D.Wood, "Voronoi diagram for general metrices in the plane", in preparation

[5] Thurston, "The geometry of circles: Voronoi diagrams, Moebius transformations, convex hulls, Fortune's algorithm, the cut locus and parametrization of shapes", unpublished notes, Princeton, 1986



C: $\ulcorner \rightarrow R_2^*, B_{2,3}^*, R_3^*, B_{3,2}^*, R_2^*, B_{2,4}^*, R_4^*, B_{4,1}^*, R_1^*, B_{1,2}^* \urcorner$

b d e

Q: d,e,b

Figure 13
(Projection on the plane perpendicular to the main axis of the cone)

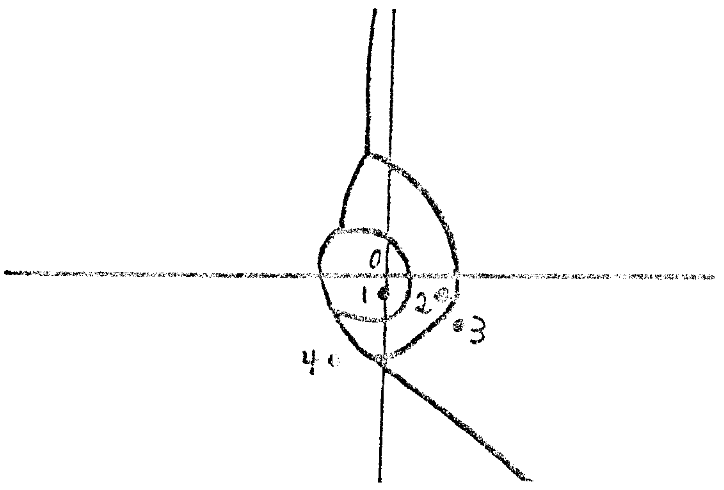


Figure 14
(Projection on the plane perpendicular to the main axis of the cone)