Voronoi Diagrams Based on Convex Distance Functions

L. Paul Chew (*) and Robert L. (Scot) Drysdale, III
Dept. of Mathematics & Computer Science
Dartmouth College
Hanover, NH 03755

ABSTRACT

We present an "expanding waves" view of Voronoi diagrams that allows such diagrams to be defined for very general metrics and for distance measures that do not qualify as metrics. If a pebble is dropped into a still pond, circular waves move out from the point of impact. If n pebbles are dropped simultaneously, the places where wave fronts meet define the Voronoi diagram on the n points of impact.

The Voronoi diagram for any normed metric, including the L_p metrics, can be obtained by changing the shape of the wave front from a circle to the shape of the "circle" in that metric. (For example, the "circle" in the L_1 metric is diamond shaped.) For any convex wave shape there is a corresponding convex distance function. If the shape is not symmetric about its center (a triangle, for example) then the resulting distance function is not a metric, although it can still be used to define a Voronoi diagram.

Like Voronoi diagrams based on the Euclidean metric, the Voronoi diagrams based on other normed metrics can be used to solve various closest-point problems (all-nearest-neighbors, minimum spanning trees, etc.). Some of these problems also make sense under convex distance functions which are not metrics. In particular, the "largest empty circle" problem becomes the "largest empty convex shape" problem, and "motion planning for a disc" becomes "motion planning for a convex shape". These problems can both be solved quickly given the Voronoi diagram. We present an asymptotically optimal algorithm for computing Voronoi diagrams based on convex distance functions.

INTRODUCTION

One of the most useful data structures in computational geometry is the Voronoi diagram (see figure 1). Given n data points in the plane, the Voronoi diagram partitions the plane into n regions, one associated with each point. The region associated with data point p consists of all points in the plane that lie closer to p than to any of the other n-1 data points. Shamos and

(*) Supported in part by NSF grant MCS-8204821.

Hoey [SH75] presented an O(n log n) divide-and-conquer algorithm for computing the diagram in the Euclidean plane and showed that the all-nearest-neighbors problem, the Euclidean minimum spanning tree problem, Knuth's post-office problem [Kn73], the largest empty circle problem, and other seemingly unrelated problems could all be solved in asymptotically optimal time by procedures which compute the Voronoi diagram as their first step.

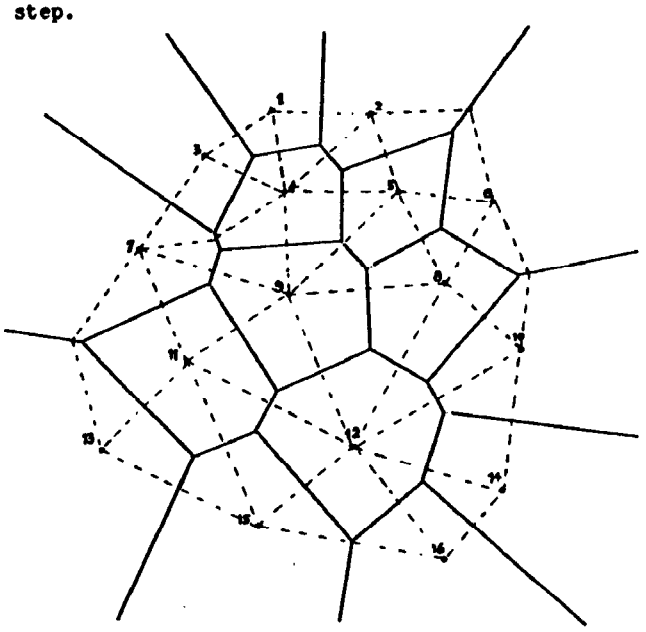

Figure 1. Voronoi diagram (solid) and its dual (dashed). From [Dr79].

A number of generalizations of the Voronoi diagram have been presented. The generalizations most relevant to this paper compute the Voronoi diagram for other distance measures. Hwang gives an algorithm for the L_1 metric, Lee and Wong give an algorithm for the L_1 and $L_{infinity}$ metrics, and Lee gives an algorithm for general L_p metrics [Hw79, LW80, Le80]. All three papers present applications for these diagrams. Unfortunately, the latter two papers have a substantive error which will be described later.

In our paper, we generalize these earlier results by using the "expanding wave" view of a Voronoi diagram described in the abstract. We present an asymptotically optimal algorithm for computing Voronoi diagrams where the distance function is based on any oriented convex shape.

VORONOI DIAGRAMS AND WAVES

Once we have the idea that a Voronoi diagram is just the set of points where circular waves collide, it is natural to ask what would happen if we use waves of a different shape. If C is our shape with center o in the interior of C then the distance from o to v is the amount that C must be expanded (or contracted) for the boundary of C to touch v. In other words, the distance d(v) from o to v is $|v|/|v_0|$, where v_0 is the point at which the ray from o to v intersects the boundary of C ($|v|$ represents the usual Euclidean distance from o to v). This idea of expanding "circles" provides some very helpful intuition for convex distance functions.

This "distance" based on shape C will be well-defined iff, in the process of expanding C, the boundary of C hits the point v just one time. It is easy to see that this condition is equivalent to requiring that the shape C is star-shaped.

This simple condition on C is not enough to ensure that a meaningful "Voronoi diagram" can be constructed based on the distance function defined by shape C. For most applications, it is important that each region be connected so that the numbers of faces, edges, and Voronoi points are all O(n). If C is not convex then Voronoi regions are not necessarily connected. Further, if p is one of the data points and if y is a point in the Voronoi

region of p, then we would expect any point between p and y to be closer to p than to any of the other data points. This is just another way of stating that the Voronoi region of p should be star-shaped as seen from p.

Lemma. For a given distance function d, all possible Voronoi regions are star-shaped iff the Triangle Inequality holds for d.

Proof: Assume there exists a Voronoi region, with corresponding data point p, that is not star-shaped. Then there exists a point y in p's region and a point x on segment py that is closer to some other data point q than to p. That is, $d(q,x) < d(p,x)$. But this implies that $d(q,x)+d(x,y) < d(p,x)+d(x,y) = d(p,y)$. By the Triangle Inequality, $d(q,y) \leq d(q,x)+d(x,y)$. From this, it follows that $d(q,y) < d(p,y)$, a contradiction, since y is supposed to be in the Voronoi region of p. Thus, the Triangle Inequality implies that Voronoi regions are star-shaped. To complete the proof, note that an example in which the Triangle Inequality fails can be used to construct a nonstar-shaped Voronoi region.

Lemma. For a given shape C and for the distance function d determined by C, the Triangle Inequality holds for d iff the shape C is convex.

Distance functions based on convex shapes are called convex distance functions and were first defined by Minkowski in 1911. Indeed, in some references it is called the Minkowski functional for C [KN63] or the Minkowski distance function [La72]. For a proof of the lemma see, for instance [La72, KN63, Ca59]. The Triangle Inequality is strict (for 3 noncollinear points) iff the convex shape has no flat edge.

Note that the usual Euclidean distance is the convex distance function of the unit circle, the L_1 metric ($d(v) = |v_x| + |v_y|$) is the convex distance function of a square with side square root of 2 centered at the origin and tipped 45 degrees, and the $L_{infinity}$ metric ($d(v) = \max\{v_x, v_y\}$) is the convex distance function of a square with side 2 centered at the origin. In fact, all of the L_p metrics can be shown to be convex distance functions of various squared off oval shapes.

A distance function defined using an asymmetric shape C will not be a metric because symmetry will not hold. The distance from p to q is not necessarily the same as the distance from q to p.

APPLICATIONS

1. Finding the largest empty "circle".

In the same way that a Euclidean Voronoi diagram can be used to quickly find the largest empty circle within a given region, a Voronoi diagram based on a convex distance function can be used to find the largest empty _____, where the blank can be filled in with any oriented convex shape. We know of no other results on this problem, although Chazelle, Drysdale, and Lee [CDL84] present an algorithm for the related problem of finding the largest empty oriented rectangle. Their method considers all rectangles while our method is restricted to shapes geometrically similar to an initial shape.

In the Euclidean case, if we are given n initial points and the center of the circle is constrained to lie within a given polygon P then the largest empty circle will be centered at either

(1) a Voronoi point (a point where 3 or more Voronoi regions touch), (2) a point of intersection of the boundary of polygon P with the Voronoi diagram, or (3) a vertex of the polygon P. Thus, to find the largest empty circle centered within a polygon P, we need only construct the Voronoi diagram on the n points, then check each of the 3 classes of points where the circle could be centered. If P is convex with O(n) edges then the entire process can be carried out in O(n log n) time.

The same process works for convex distance functions with one subtlety: if the desired empty shape C is not symmetric about its center then the convex distance function should be based on the reflection of C about its center. This is a result of the same problem that causes a distance function based on C to be a nonmetric. All the properties of a metric hold except for symmetry: the distance from p to q is not necessarily the same as the distance from q to p. A Voronoi point is equidistant from 3 (or more) of the initial points, but these distances are measured from an initial point to the Voronoi point. The distance from the Voronoi point to each of these initial points is likely to be different than the distance in the other direction. But a moment's reflection(!) will convince the reader that the reflection of the convex distance function will also cause the distances to reflect. In other words, the distance from p to q measured using the convex distance function is the same as the distance from q to p using the reflected convex distance function. Thus, to find the largest empty "circle" of shape C we construct a Voronoi diagram using the distance defined by C^R, the reflection of C about its center.

2. Motion planning for robots.

O'Dunlaing, Sharir, and Yap [OSY83] have shown how Voronoi diagrams can be used to plan motion for a disc-shaped robot. To find if a disc-shape can be moved from point p to point q in the presence of disjoint polygonal obstacles, one first computes the Voronoi diagram for the obstacles. This can be done using either of two O(n log n) methods for constructing a Voronoi diagram for a set of line segments, one due to Kirkpatrick [Ki79], the other due to Yap [Ya84]. (Yap's method will also handle obstacles made of certain simple curves.) Note that if there is a path from p to q then there is a path following the boundaries of the Voronoi regions since the Voronoi boundaries are as far from obstacles as possible. Thus, there is a path from p to q iff there is a path from p to q on the graph made up of the Voronoi boundaries that remain after we have eliminated boundaries too close to obstacles. Voronoi boundaries that are too close to obstacles can be eliminated in O(n) time by simply checking each boundary to see if it is too close to the obstacle that creates that boundary. Hence, given the Voronoi diagram we can determine if there exists a path from p to q using a simple depth first search of an O(n) size graph. Total time for construction of the Voronoi diagram and the depth first search is O(n log n).

We can use the same idea to detect if there is a path from p to q for a P-shaped robot where P is an arbitrary convex polygon. As above, one first constructs the Voronoi diagram for the set of obstacles, but instead of using the usual Euclidean Voronoi diagram, one uses the distance function defined by the reflection of the robot shape P. As in the largest empty "circle" problem we must use

the reflection when the robot is asymmetric about its center. The construction of a Voronoi diagram is straightforward if the obstacles are points (this algorithm is explained below). If the obstacles are disjoint polygons then we believe a construction similar to that of Yap [Ya84] can be used. Things are actually simplified a bit by the fact that for a distance function based on a convex polygon all Voronoi boundaries consist of straight line segments. (In the Euclidean case, the Voronoi boundary between a point and a line is a parabola.) The Voronoi boundaries form an O(n) size graph on which depth first search can be used. Thus, there is an O(n log n) algorithm for detecting if there is a path for a convex polygon shaped robot to move from p to q in the presence of disjoint polygonal obstacle. Note that the polygonal robot must have a fixed orientation. Turning the robot causes the distance measure (and, thus, the Voronoi diagram) to change.

THE ALGORITHM FOR EUCLIDEAN VORONOI DIAGRAMS

The divide-and-conquer algorithm for computing V(S), the Voronoi diagram for a set S containing n points, was developed by Shamos and Hoey and improved by Lee [SH75, Le82]. The first step is to sort the points lexicographically on their (x, y) coordinates. Split the points into a leftmost set L and a rightmost set R, each containing about half of the points. Recursively compute V(L) and V(R). Finally, merge the two diagrams into a single diagram V(S). If this merge step can be done in O(n) time, the entire algorithm will take O(n log n) time, which is optimal.

The tricky part is the merge step. The key is computing the bisector curve B(L,R) which separates that part of the plane closer to some point in L from that part of the plane closer to some point in R (see figure 2). V(S) consists of three parts: (1) B(L,R), (2) the part of V(L) lying to the left of B(L,R), and (3) the part of V(R) lying to the right of B(L,R).

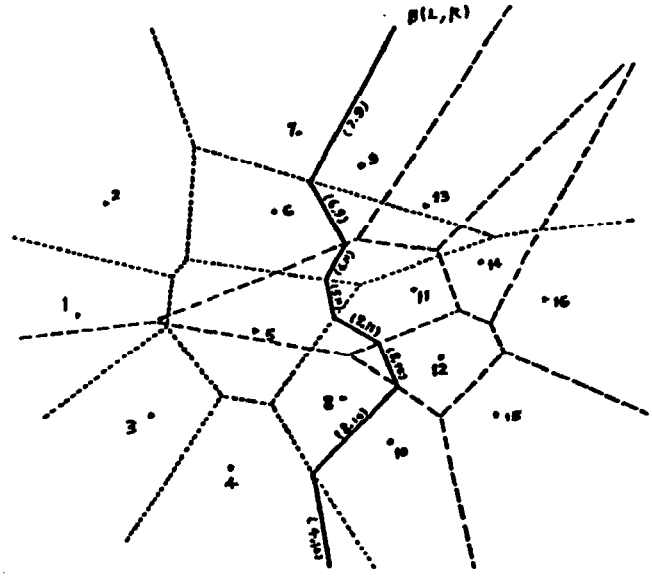

Figure 2. V(L), V(R), and the bisector curve B(L,R). From [Dr79].

The curve B(L,R) is a polygonal line with two infinite rays. Each segment is the perpendicular bisector between a point in L and a point in R. Computing B(L,R) is done in two parts. First, some starting segment on the curve must be found. Next, the curve must be traced through the diagram. Lee's clockwise-counterclockwise scan procedure traces the bisector and updates the diagram in O(n) time given a starting segment.

The starting segment is found by observing that unbounded regions in the diagram correspond to points on the convex hull of the data points. The convex hull of S consists of the left side of the

convex hull of L, the right side of the convex hull of R, a new edge at the top, and a new edge at the bottom (see figure 3). The infinite rays of B(L,R) correspond to the pairs of points joined by the new edges. These rays give the starting and ending points of B(L,R). If the computation of the convex hull for each half is carried out at the same time as the computation of the Voronoi diagram for each half then the two new edges can be computed and the hull updated in O(n) time. Therefore the merge step can be done in O(n) time.

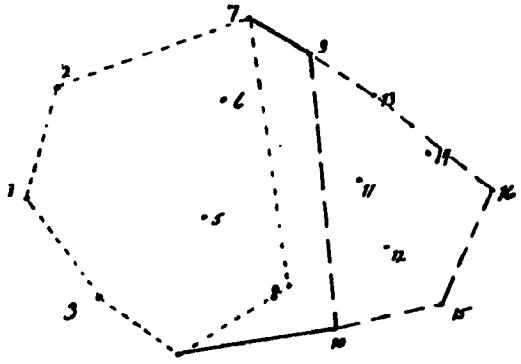

Figure 3. Finding the convex hull. From [Dr79].

POTENTIAL PROBLEMS WITH CONVEX DISTANCE FUNCTIONS

If we try this algorithm using an arbitrary convex distance function, several things can go wrong. First, the bisector B(L,R) can fail to be a curve (it can be a region). Second, it can break into several pieces, so that tracing a single piece is not enough. Finally, the method of finding a starting segment can fail.

Lee, Wong, and Hwang all note that for the L_1 metric, it is possible to get two-dimensional regions which are equidistant from two points [Le80, LW80, Hw79]. This happens whenever the points are at the ends of the diagonal of a square. In general, this can occur whenever the Triangle Inequality is not strict. They get around this difficulty by defining the clockwise-most boundary of such a region to be the bisector. We use the same method.

The bisector breaks into pieces whenever the region of a point p in one half "pushes between" two points in the other half and continues on to infinity. This can only occur when there are "corners" on the convex shape. (Corners are places where the tangent to the shape is not uniquely defined.)

Lee, and Lee and Wong claim that for arbitrary L_p metrics the bisector segment consists of a single piece, but do not prove it. This is incorrect for L_1. Figure 4 shows that the bisector in L_1 can break into n-1 pieces. This flaw causes their algorithm to sometimes fail. However, rotating the points 45 degrees before sorting them eliminates this problem. (Hwang effectively does this rotation.) We state conditions under which the bisector will remain a single piece below.

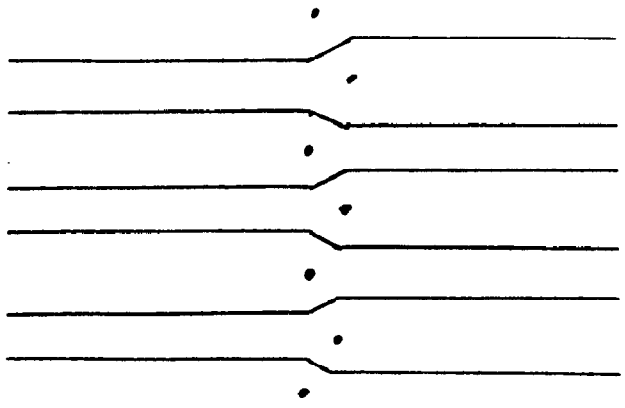

Figure 4. An example where the bisector curve breaks into pieces (L_1 metric).

Finally, we must consider finding a starting bisector. Lee proves that the convex hull method for finding a starting segment works for L_p metrics with $1 < p <$ infinity, but fails for L_1 and $L_{infinity}$. This is because points defining infinite regions need not be on the convex hull (see figure 5). [Le82, LW80, and Hw79] give three different ways of finding a starting segment. For arbitrary convex distance functions none of these methods work; so we need some new method for starting the boundary.

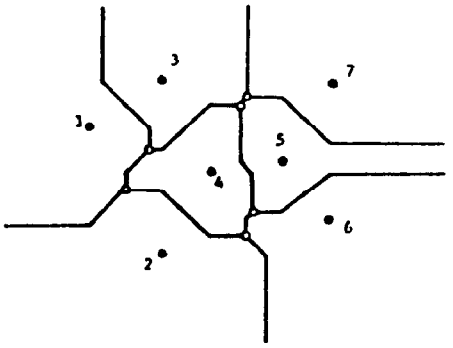

Figure 5. Voronoi diagram for the L_1 metric. Point 5 has an infinite region, but is not on the convex hull. From [Le80].

VORONOI ALGORITHM FOR CONVEX DISTANCE FUNCTIONS

The algorithm for computing the Voronoi diagram based on a convex distance function has the same basic steps as the Euclidean Voronoi algorithm. The only step that causes any new difficulties is finding the boundary B(L,R) between the two regions.

Note: in the following discussion we make a distinction between a piece of a boundary and a segment of a boundary. We use piece to represent a maximal connected subset of the boundary set B(L,R). For the Euclidean Voronoi algorithm the bourdary is always a single piece. We use segment to represent a part of a bisector between two points. Thus, a boundary piece is made of one or more boundary segments. For the Euclidean Voronoi diagram a segment is a straight line segment. For a Voronoi diagram based on a convex polygon a segment is a polygonal line (where the wave fronts from two points meet).

We can ignore the problem of the boundary being disconnected, provided we can find a way to start each disjoint piece of the boundary. To see this note that the boundary, regardless of how many disjoint pieces it contains, is made of O(n) boundary segments. (The whole Voronoi diagram is made of O(n) segments.) Thus, if we are given some starting point on each disjoint piece of the boundary then we can compute the entire boundary in time O(n).

Lemma. The boundary B(L,R) can be found in time O(n) using the clockwise-counterclockwise scan procedure, provided we are given at least one starting segment from each disjoint piece of the boundary.

Proof: The proof of this lemma for the standard Voronoi diagram counts operations and charges them to segments eliminated from V(L) or V(R) or to new Voronoi points created in V(S). The same arguments hold here.

The only remaining difficulty is how to start each disjoint piece of the boundary. The Euclidean Voronoi diagram algorithm starts the boundary by finding one of the two infinite boundary segments. We start each disjoint piece of the boundary in the same way using a simple algorithm (given below) to find infinite bourdary segments. But that still leaves us with a small problem: islands. An island

is a piece of the boundary that curves back on itself. Such a boundary piece would not include a segment that extends to infinity, and would, thus, be missed if we attempt to start each disjoint piece of the boundary by locating the infinite boundary segments. Fortunately, boundary islands do not exist.

Lemma. Each disjoint piece of B(L,R) contains a segment that extends to infinity (i.e., there are no islands).

Proof: Assume that the lemma is false. Then there exists a piece of B(L,R) that forms a closed loop encircling a set Q of points from either L or R. The cases are symmetric, so we will assume that the points of Q are from L. Let B' be the piece of boundary B(L,R) that encircles Q. (See figure 6.) Any point p on B' is equidistant from the nearest point in Q and the nearest point in R. Because Voronoi regions are star-shaped, the segment connecting p to its nearest neighbor in R cannot pass through the interior of the island. Consider what happens to the segment connecting p to the closest point in R as p is moved around the entire island along B'. (The closest point in R changes depending on where p is along B'.) Let q be any point of Q, the set of points encircled by B'. The segment connecting p to the nearest point in R will sometimes lie above q and will sometimes lie below q. As p passes around the left end of B' there will be some point at which the segment switches from above q to below q. Call the point where this switch occurs x. Let a and b be the two points in R which are equidistant from x when the switch occurs, with point a connected to x by a segment above q and point b connected to x by a segment below q. Let c be the nearest point in Q to x. Because x is on B(L,R), x is equidistant from a, b, and c. In other words, if we take the shape C defining the distance function, reflect it about its center, center it on x, and let it grow then it must touch a, b, and c at the same instant. But c lies strictly inside the triangle abx, because it is in L while a and b are in R. This contradicts the fact that C is convex. Therefore the assumption that an island exists is incorrect.

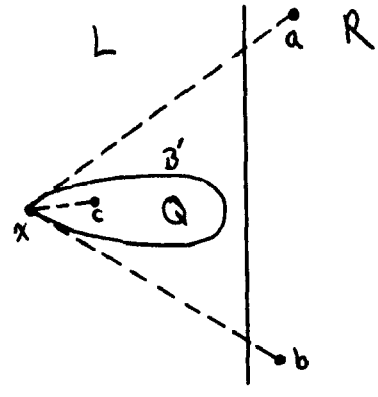

Figure 6. Proof that there are no islands.

To find the infinite boundary segments note that any segment of B(L,R) that extends to infinity must be closest to a single point in L and closest to a single point in R. After all, the segment is part of a bisector between two points, one from L and one from R. Thus, the segment must lie entirely within an infinite region of V(L) and entirely within an infinite region of V(R). The algorithm outlined below will check each possible pair of infinite regions.

1. Find the region pairs (A_L, A_R), where A_L is from V(L), and A_R is from V(R), and for which the intersection of A_L and A_R has infinite area.
2. For each such pair of regions (A_L, A_R);

3. Let p_L be the point of L corresponding to region A_L and let p_R be the point R corresponding to region A_R;

4. Compute the bisector of p_L and p_R;

5. If the bisector extends to infinity within the intersection of A_L and A_R then that part of the bisector is a segment of B(L,R).

Lemma. The above method will find every segment of B(L,R) that extends to infinity in O(n) time.

Proof: It is clear that the algorithm will find the infinite boundary segments. We show the algorithm outlined above can be done in O(n) time by using a method similar to merging. V(L) and V(R) provide an ordering for the infinite regions; all we need to do is march around the outer perimeter of each diagram. We can compare regions as we go, checking to see whether infinite boundaries intersect to determine those cases where the regions have an infinite-area intersection. This will allow us to find all the necessary region pairs in time O(n). Thus, the loop is executed O(n) times. Each step within the loop can be done in constant time so the total process takes time O(n).

By combining our method for finding starting segments for B(L,R) with the clockwise-counterclockwise scan procedure we can find B(L,R) in O(n) time. V(S) can then be constructed from B(L,R), V(L), and V(R) in time O(n), giving the following theorem.

Theorem. For any convex distance function d and set of points S, the Voronoi diagram for the set S and distance d, $V_d(S)$, can be found in O(n log n) steps and O(n) space.

IMPLEMENTATION CONSIDERATIONS

All the time bounds given above are based on the assumption that we can perform these operations in constant time:

1) Given two points, compute the bisector between them. That is, find the boundary where the two wavefronts meet.

2) Given two such bisectors, compute their intersection(s).

When we compute run times, we count the number of times that these operations are performed. For the case where the wave shape is a polygon the operations are easy: the bisector between two points is a polygonal line that can be computed with some simple vector operations.

A SIMPLER SPECIAL CASE

Theorem. Suppose it is possible to rotate C, the shape that defines the distance function, so that it has a uniquely defined vertical tangent on each side. If we first rotate the entire problem so that C has the above orientation then the boundary B(L,R) will always be a single piece.

Proof: Suppose B(L,R) consists of more than one piece. Since we know there are no islands, it must be the case that there is an infinite region "punching through" from either left-to-right or right-to-left. Without loss of generality, we can assume left-to-right. Let p be the point corresponding to this infinite region. Consider a ray from p that stays within the region and has a point a of R above the ray and a point b of R below the ray. (See figure 7.) This ray exists because the region is star-shaped and the region "punches through" R. C has a vertical tangent on its right

side (say at c). Choose a neighborhood size so that all points on the circle in the neighborhood about c are at slopes closer to vertical than either pa or pb. Now expand C until the neighborhood size is greater than twice the Euclidean distance from a to b. Let C' be the reflection of this enlarged version of C about its center. Place C' so that its center is on the ray from p and so that neither a nor b are within C', but at least one of a and b is on the boundary of C'. Without loss of generality we can assume that b is on the boundary of C'. Let x be the point on the ray from p that is the position of the center of C'. Note that because the section of C' between a and b corresponds to the neighborhood of c where the slope is nearly vertical, p must be outside of C'. Thus the reflected distance from x to p is greater than the reflected distance from x to b. In other words, d(p,x) is greater than d(b,x) where d is the distance function defined by C. This is a contradiction since x was supposed to be within the Voronoi region of p.

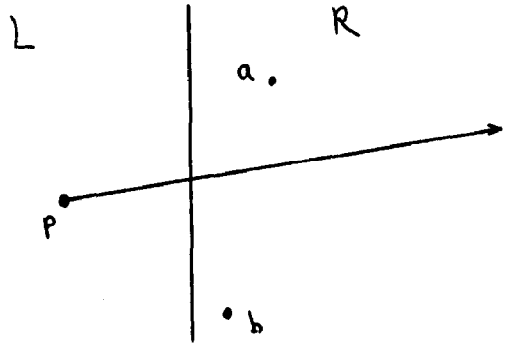

Figure 7. Proof that the boundary stays in one piece if C has vertical tangents.

For shapes that satisfy the theorem, we can rotate the points and apply the usual algorithm. However, to start the bisector we must use a method like the one in Lee and Wong [LW80] rather than the method that depends on the convex hull].

REFERENCES

[Ca59] J. W. S. Cassels, An Introduction to the Geometry of Numbers, Springer-Verlag, Berlin (1959).

[CDL84] B. Chazelle, R. L. Drysdale, and D. T. Lee, Computing the Largest Empty Rectangle, Proc. Symp. Theoretical Aspects of Comp. Sci. (1984), 43-54.

[Dr79] R. L. Drysdale, Generalized Voronoi Diagrams and Geometric Searching, Stanford PhD thesis, report STAN-CS-79-705, Stanford U., 1979.

[Hw79] F. K. Hwang, An O(n log n) algorithm for rectilinear minimal spanning trees, JACM 26(1979), 177-182.

[Ki79] D. G. Kirkpatrick, Efficient Computation of Continuous Skeletons, Proc. 20th IEEE Symposium on Foundations of Computer Science, 1979, 18-27.

[KN63] J. L. Kelley and I. Namioka, Linear Topological Spaces, Van Nostrand, Princeton (1963).

[Kn73] D. E. Knuth, The Art of Computer Programming, vol. 3, Addison-Wesley, Reading, MA (1973).

[La72] S. R. Lay, Convex Sets and their Applications, Wiley, New York (1972).

[Le80] D. T. Lee, Two-Dimensional Voronoi Diagrams in the L_p metric, JACM 27(1980), 604-618.

[Le82] D. T. Lee, On k-nearest neighbor Voronoi diagrams in the plane, IEEE TRANS. COMP. C-31(1982), 478-487.

[LW80] D. T. Lee and C. K. Wong, Voronoi diagrams in L_1 ($L_{infinity}$) metrics with 2-dimensional storage applications, SIAM J. COMPUT. 9(1980), 200-211

[OSY83] C. O'Dunlaing, M. Sharir, and C. K. Yap, Retraction: A new approach to motion planning, Proc. 15th Annual ACM Symposium on Theory of Computing, 1983, 207-220.

[SH75] M. I. Shamos and D. Hoey, Closest-Point Problems, Proc. 16th IEEE Symposium on Foundations of Computer Science, 1975, 151-162.

[Ya84] C. K. Yap, An O(n log n) algorithm for the Voronoi diagram of a set of simple curve segments, Technical Report, Courant Insttitute, New York University, Oct. 1984.