Computational Geometry: Theory and Applications 1 (1992) 171–187
Elsevier

A convex hull algorithm for discs, and applications

David Rappaport*

Department of Computing and Information Science, Queen's University, Kingston, Ont., Canada K7L 3N6

Communicated by Hiroshi Imai
Submitted 15 October 1990
Accepted 30 August 1991

Abstract

Rappaport D., A convex hull algorithm for discs, and applications, Computational Geometry: Theory and Applications 1 (1992) 171–187.

We show that the convex hull of a set of discs can be determined in $\Theta(n \log n)$ time. The algorithm is straightforward and simple to implement. We then show that the convex hull can be used to efficiently solve various problems for a set of discs. This includes $O(n \log n)$ algorithms for computing the diameter and the minimum spanning circle, computing the furthest site Voronoi diagram, computing the stabbing region, and establishing the region of intersection of the discs.

Keywords. Discs; convex hull; Voronoi diagram; transversals; minimum spanning circle.

1. Introduction

In many applications using geometric data, a set of sites on a plane is modelled as a set of points. Much work has been done in solving problems efficiently for planar point sites. In this paper we consider a collection of geometric problems by modelling sites as a set of discs of various sizes.

In Section 2 we begin by presenting an algorithm to compute the convex hull of a set of discs. We then show how the convex hull algorithm can be used to solve a collection of other problems. The key to many of the results is to consider the set of discs as a cross section of a set of axis parallel cones.

An algorithm to compute the so called stabbing region for a set of discs is presented in Section 3. A *stabbing line,* or a *line transversal,* of a set of objects is a straight line that intersects every object in the set. In [7] an $O(n \log n)$

* This research was supported in part by the Natural Sciences and Engineering Research Council of Canada grant A9204.

algorithm is introduced to determine whether a stabbing line exists for a set of line segments. The *stabbing region,* a concise description of all possible stabbing lines is also obtained in $O(n \log n)$ time. A simple object is an object that can be described in O(1) space and pairwise tangents and pairwise intersections can be computed in O(1) time. Examples of simple objects are *k-gons* (k fixed), ellipses, etc. Atallah and Bajaj [2] obtain stabbing regions of a set of simple object in $O(\lambda_t(n)\log n)$ time. The function $\lambda_t(n)$ grows very slowly, i.e., $\lambda_t(n) = O(n)$, $t < 3$, $\lambda_t(n) = O(n \log^* n)$, $t \geqslant 3$. The parameter t represents the maximum number of common support lines admissible by a pair of simple objects. For example for discs $t = 2$ and for triangles $t = 6$. The function $\lambda_t(n)$ can also be considered as the length of a *Davenport–Schinzel sequence* of parameter t. We show how the convex hull algorithm of Section 2 can be used to implement the ideas in [2] for discs. We also introduce a novel approach to the representation of the stabbing region for a set of objects.

In Section 4 we show how to compute the diameter of a set of discs by using the convex hull algorithm. In Section 5, our algorithm to compute the convex hull of discs is used to compute the intersection of a set of three dimensional right circular cones. The results of Section 5 are applied in Section 6 to obtain the intersection of a set of discs.

In Section 7, we present an algorithm to compute the furthest site Voronoi diagram for a set of discs. Given a set of point sites, S, in E^2, the Voronoi diagram is a partition of E^2 into disjoint *Voronoi regions* such that for each site $s \in S$, the Voronoi region $V(S)$ is the locus of points that are closest to the site s than to any other site [12–13]. A generalization of the Voronoi diagram, the *furthest site Voronoi diagram,* is a partition of E_2 such that for each site $s \in S$ the furthest site Voronoi region $FV(s)$ is the locus of points further from the site s than from any other site. As was shown in [12], the region $FV(s)$ is non-empty if and only if s is on the convex hull of S. Shamos and Hoey [13] introduced on $O(n \log n)$ divide and conquer algorithm for computing the Voronoi diagram for a set of point sites. Briefly, Shamos and Hoey's algorithm splits a set of points by a vertical separating line into two roughly equal parts, and recursively computes their Voronoi diagrams. A linear time algorithm is then used to merge the two diagrams. The merge step uses the geometry of Voronoi regions and requires careful transversal of fairly complex data structures. In subsequent years, similar methods have been used to find Voronoi diagrams for different types of sites. Kirkpatrick [9] examines the geometric structure of the Voronoi diagram of a set of line segments to tailor a linear time merge step for its efficient computation. Sharir [14] has analyzed the geometry of the Voronoi diagram for sites that are discs, and has developed an $O(n \log n)$ merge step, leading to an $O(n \log^2 n)$ algorithm. Yap [19] examines the case where sites are straight or curved line segments and gives an $O(n \log n)$ algorithm. Fortune [8] has presented a novel way of constructing Voronoi diagrams. Fortune's result relies on the property that vertices in the Voronoi diagram are intersection points of a suitably constructed

set of axis parallel cones. He then uses a sweeping plane to detect these intersections. The same method is modified slightly to compute the Voronoi diagram for a set of discs. A more complicated version is also presented to compute the Voronoi diagram for a set of line segments. Keeping in spirit with the method of [8], we use the algorithm for computing the intersection of a set of right circular cones, presented in Section 5, to compute the furthest site Voronoi diagram for a set of discs in $O(n \log n)$ time.

In the final section of the paper we discuss the expected time complexity of the algorithms that have been presented. We show that under certain assumptions the algorithms have an $O(n)$ expected time complexity.

2. A convex hull algorithm for a set of discs

Let S be a set of closed planar discs. To simplify the presentation we assume that the discs are in general position. The convex hull of S, Hull(S), is the smallest convex region containing S. Let $\partial(R)$ denote the boundary of a simple closed set R. Then $\partial(\text{Hull}(S))$ denotes the boundary of the convex hull of S, and consists of straight line segments or edges and arcs of circles. To be more precise, define L as a directed line, and let $H(L)$ denote the closed right half-plane of L. We say that a half plane is a *supporting half plane* of the set S if it contains S in its interior. We say a line L is a *support line* of a set of discs S if $H(L)$ is a supporting half plane of S and no subset of $H(L)$ is a supporting half plane of S. Let A and B be two sets of discs. The common support line of A and B, if such a line exists, directed from A to B, is defined as $L(A, B)$. let $t(A, B)$ denote a closed subset of $L(A, B)$ with one endpoint on $\partial(A)$ and one endpoint on $\partial(B)$. Let a and b denote two different discs in S. We define an edge of Hull(S) as $t(a, b)$ such that $L(a, b)$ is a supporting line of Hull(S). We represent $\partial(\text{Hull}(S))$ by a list of discs $s \in S$, that is, $CH(S) = (s_0, s_1, \ldots, s_h)$, such that $t(s_i, s_{i+1})$, is an edge of Hull(S) for $i = 0, \ldots, h-1$. Note that a disc may appear more than once on $\partial(S)$, so the list $CH(S)$ may contain elements s_i and s_j where $i \neq j$ but $s_i = s_j$. See Fig. 1. $CH(S)$ is viewed as a cyclic sequence. We overcome the complication of using a circular list by setting $s_0 = s_h$.

Lemma 2.1. $|CH(S)| \leq 2n - 1$.

Proof. Let u, v denote any two distinct discs in S. We show that $u, \ldots, v, \ldots, u, \ldots, v$ is a forbidden subsequence of the list $CH(S)$. Let S_0, S_1, S_2, S_3, S_4 denote subsequences of $CH(S)$ so we can write $CH(S)$ as S_0, u, S_1, v, S_2, u, S_3, v, S_4. First consider the case where S_1 and S_3 are of length 0. (Or S_0, S_2 and S_4 are of length 0). This implies that $t(u, v)$ appears twice as an edge of $CH(S)$, which is absurd. (Or $t(u, v)$ appears twice as an edge of $CH(S)$). Suppose on the other hand the S_1 is not of length 0. This implies that $L(u, v)$ is a line

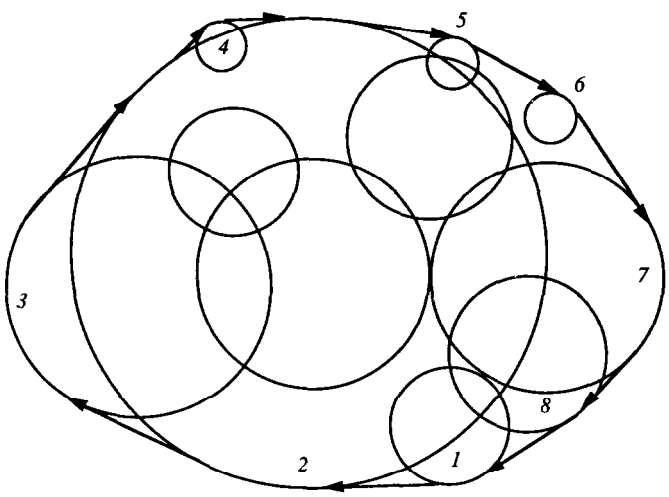

Fig. 1. A set of discs S. We have numbered only those discs with arcs on $CH(S)$. Thus $CH(S) = (1, 2, 3, 2, 4, 2, 5, 6, 7, 8, 1)$.

intersecting the interior of Hull(S). Let ψ denote the intersection of $L(u, v)$ and Hull(S). The second occurrence of the subsequency $u, \ldots, v$ in $CH(S)$ implies that $L(u, v)$ has an intersection with Hull(S) disjoint from ψ, a contradiction. Sequences with the property that identical elements are not adjacent and occurrences of the form $u, \ldots, v, \ldots, u, \ldots, v$ are forbidden are known as Davenport–Schinzel sequences with the parameter 2 [6, 15–16]. An inductive proof shows that the maximum length of these sequences in $2n - 1$. The bound is tight, as can be seen by an input S of $n - 1$ small discs adjacent to one large disc to create a list $CH(S)$ of the form $(1, n, 2, n, 3, n \cdots)$ □

We proceed by describing an algorithm to compute the convex hull of S. The algorithm uses a divide and conquer approach. The set of discs is split into two roughtly equally sized subsets called P and Q. We recursively compute the convex hulls of the subsets, and then merge. Thus an efficient algorithm to merge two (possibly intersecting) convex hulls of discs is germane to an effective solution. The merge algorithm rotates a parallel pair of supporting lines L_p and L_q one for the set P and one for the set Q. At each iteration a decision is made regarding whether the current arc is extreme to both hulls, that is, we decide if the current arc belongs to $CH(P \cup Q)$.

We make use of the following functions.

- Given the directed lines L_1 and L_2, the function $\alpha(L_1, L_2)$ returns the positive angle swept clockwise from L_1 to L_2.
- Given a disc s in $CH(P)$, the function succ(s) denotes the successor of s in $CH(P)$. If Hull(P) is a single disc p then $CH(P) = p$, and succ(p) returns p always.

• Given parallel supporting lines of P and Q, respectively denoted by L_p and L_q, the function dom(L_p, L_q) returns true if $H(L_q)$ is a proper subset of $H(L_p)$.
• Given a list ϑ and an element φ, the function Add(ϑ, φ) returns ϑ if φ is currently the last item in ϑ, otherwise it inserts φ at the end of ϑ and then returns ϑ.

Algorithm Hull

1. Split S arbitrarily into two disjoint subsets of discs, P and Q, such that $|P|$ and $|Q|$ differ by at most one.
2. Recursively find $CH(P)$ and $CH(Q)$.
3. Use algorithm merge to merge $CH(P)$ and $CH(Q)$ resulting in $CH(S)$.

Algorithm Merge

Input: $CH(P)$ and $CH(Q)$.
Output: $CH(S) = CH(P \cup Q)$
{Initialization}
Let L_p and L_q denote lines supporting P and Q respectively, and tangent to the sites $p \in P$ and $q \in Q$ such that both L_p and L_q are parallel to a given line L^*. Initialize $CH(S)$ empty.

{We make use of a procedure Advance to advance to the next arc in either $CH(P)$ or $CH(Q)$.}

repeat
  if dom(L_p, L_q) **then** Add($CH(S), p$); Advance(L^*, p, q);
    else {dom(L_q, L_p)} Add($CH(S), q$); Advance(L^*, q, p);
  $L_p \leftarrow$line parallel to L^* and tangent to P at p;
  $L_q \leftarrow$line parallel to L^* and tangent to Q at q;
until every arc in $CH(P)$ and $CH(Q)$ has been visited.

procedure Advance (L^*, x, y);
{Find common lines of support and advance on the minimum angle.}
{We test for edges that bridge the two hulls, that is, edges of the form $t(x, y)$ and $t(y, x)$.}
{If $L(x, y)$ does not exist then $\alpha(L^*, L(x, y))$ is undefined.}
  $a_1 \leftarrow \alpha(L^*, L(x, y)); a_2 \leftarrow \alpha(L^*, L(x, \text{succ}(x)))$;
  $a_3 \leftarrow \alpha(L^*, L(y, \text{succ}(y))); a_4 \leftarrow \alpha(L^*, L(y, x))$;
  if $a_1 = \min(a_1, a_2, a_3)$ **then** Add($CH(S), y$); {$t(x, y)$ is a bridge}
    if $a_4 = \min(a_4, a_2, a_3)$ **then** Add($CH(S), x$); {and $t(y, x)$ is a bridge too}
  if $a_2 < a_3$ **then** $L^* \leftarrow L(x, \text{succ}(x)); x \leftarrow \text{succ}(x)$;
    else $\{a_3 < a_2\}$ $L^* \leftarrow L(y, \text{succ}(y)); y \leftarrow \text{succ}(y)$;

We proceed by proving correctness for algorithm Merge.

Lemma 2.2. *At every iteration of algorithm Merge L_p and L_q are parallel, and L_p supports P and L_q supports Q.*

Proof. We initialize L_p and L_q so that the invariant is true. Assume that the invariant is true at the start of an arbitrary iteration. We must show that after updating L_p and L_q the invariant is maintained. Clearly L_p and L_q are always parallel. Consider consecutive edges of $CH(P)$ (similarly for $CH(Q)$), for example, $t(\text{pred}(p), p)$ and $t(p, \text{succ}(p))$. Every line tangent to p lying between $\alpha(L(\text{pred}(p), p)), L(p, \text{succ}(p)))$ is a support line of P. To show that L_p and L_q are support lines we examine procedure Advance. Our sweeping method can be described as sweeping L_p and L_q until a subsequent edge on $CH(P)$ or $CH(Q)$ is encountered. In either case, one line supports an edge of a convex hull and the other line lies between consecutive edges of the other hull. Therefore the assertion is true at every iteration. □

An edge $t(x, y)$ of $CH(P \cup Q)$ is defined to be a *bridge* if x and y are not both in P nor both in Q.

Lemma 2.3. *Let* $\sup(p, q)$ *denote a continuous interval of* $[0, 2\pi]$ *such that for every value* θ *in* $\sup(p, q)$ *there exist parallel support lines* L_p *and* L_q *with angle* θ. *Referring to procedure Advance, we can assume without loss of generality that* $x = p$ *and* $y = q$ *that is,* $\text{dom}(L_p, L_q)$ *is true. A line* $L(p, q)$ *with angle* θ *in* $\sup(p, q)$ *supports a bridge if and only if* $a_1 = \min(a_1, a_2, a_3)$. *Furthermore, a line* $L(q, p)$ *with angle* θ *in* $\sup(p, q)$ *is a bridge if and only if* $t(p, q)$ *is a bridge and* $a_4 = \min(a_4, a_2, a_3)$.

Proof. We first show that if the conditions are satisfied then $t(p, q)$ is a bridge. We know that P is supported by L_p and $L(p, \text{succ}(p))$ and Q is supported by L_q and $L(q, \text{succ}(q))$. Since a_1 makes the smaller internal angles with L_p and L_q, $L(p, q)$ supports both P and Q, and $t(p, q)$ is therefore a bridge. Suppose on the other hand that $t(p, q)$ is a bridge but a_1 is not the minimum angle. This implies that there exists at least one point of P or of Q that is above the line $L(p, q)$. Thus $L(p, q)$ does not support $P \cup Q$, and $t(p, q)$ is not a bridge. A symmetric argument can be used to show that $t(q, p)$ is a bridge. □

Lemma 2.4. *Algorithm Merge correctly computes* $CH(S) = CH(P \cup Q)$.

Proof. The edges of $CH(S)$ are either hull edges of $CH(P)$ or of $CH(Q)$ or bridges. In every iteration of algorithm Merge we advance in either $CH(P)$ or $CH(Q)$ until every hull edge is encountered, and thus we encounter all edges that are potentially hull edges of $CH(S)$. It remains to show that all potential bridge edges are considered. Algorithm Merge advances through $CH(P)$ and $CH(Q)$ so that every pair of discs p in P and q in Q that admit parallel support lines are

considered. For each pair of discs that admit parallel support lines there can be at most two transitions in $\sup(p, q)$, that is, $t(p, q)$ and/or $t(q, p)$ are bridges. The previous lemma shows how we correctly test for each of these occurrences. □

The computational complexity of the convex hull algorithm is characterized by the recurrence relation $T(n) = 2T(n/2) + \text{Cost(Merge)}$. Algorithm Merge is an $O(n)$ algorithm and thus the complexity of the convex hull algorithm is $O(n \log n)$. The lower bound for computing the convex hull of a set of points is $\Omega(n \log n)$, and this lower bound applies to the convex hull for discs problem as well. Thus the algorithm is optimal.

Theorem. *The convex hull of a set of n discs can be computed in* $\Theta(n \log n)$ *time.*

3. Computing partial support lines and stabbing regions

Define a half plane that intersects every disc in S as a *stabbing half plane* of S. We define a *partial support line* as a line L such that $H(L)$ is a stabbing half plane of S, and no half plane contained in $H(L)$ is a stabbing half plane of S. Let $K(a, b)$ denote a line tangent to the discs a and b, directed from a to b and containing neither a nor b in its right half plane $H(K(a, b))$. Observe that $L(b, a) = K(a, b)$. We show that there exists a sequence of discs $KCH(S) = (s_0, s_1, \ldots, s_k)$, $s_0 = s_k$, such that $K(s_i, s_{i+1})$ is a partial support line of S for all $i = 0, \ldots, k-1$. Let M denote the maximum radius over all discs in S. Consider the transformation $\tau: (x, y, r) \rightarrow (x, y, M - r)$, and $\tau(S) = \{\tau(s) \mid s \in S\}$. The transformation τ takes a disc with centre (x, y) and radius r, to a disc with centre (x, y) and radius $M - r$.

Lemma 3.1. *$K(a, b)$, $a, b \in S$ is a partial support line of S if and only if $L(\tau(a), \tau(b))$ is a support line of $\tau(S)$.*

Proof. We show that if $L(\tau(a), \tau(b))$ is a support line of $\tau(S)$ then $K(a, b)$, $a, b \in S$ is a partial support line of S. Without loss of generality assume that $L(\tau(a), \tau(b))$ is vertical and the left extreme of S. Let x_a and r_a denote respectively the x coordinate of the centre and radius of the disc a, and let $M - r_a$ denote the radius of the disc $\tau(a)$. Thus we can write:

$$x_a - M + r_a = x_b - M + r_b, \text{ and}$$
$$x_a - M + r_a \leq x_c - M + r_c \quad \text{for all } \tau(c) \in \tau(S).$$

This implies:

$$x_a + r_a = x_b + r_b, \text{ and}$$
$$x_a + r_a \leq x_c + r_c \quad \text{for all } c \in S.$$

Thus $H(K(a, b))$ is a stabbing half plane and no subset of $H(K(a, b))$ is a stabbing half plane so the desired result is achieved. A symmetric argument can be used to reverse the implication. □

This lemma implies that $KCH(S)$ is well defined and is equal to $CH(\tau(S))$. Therefore, to compute $KCH(S)$ we obtain $\tau(S)$ and then algorithm Hull of Section 2 to compute $CH(\tau(S))$.

We proceed by showing how $KCH(S)$ can be used to determine the stabbing regions of S. Let K_a denote a partial support line tangent to a, and let K_b denote an anti-parallel partial support line tangent to b. Let $\mathrm{Cor}(K_a, K_b)$ denote the *corridor* of a and b, defined as the closed complement of $H(K_a) \cup H(K_b)$.

Lemma 3.2. *Every line contained in* $\mathrm{Cor}(K_a, K_b)$ *is a stabbing line of* S.

Proof. Let L be a line contained in $\mathrm{Cor}(K_a, K_b)$ and let L^+ and L^- denote L directed in opposite directions. $H(K_a)$ and $H(K_b)$ are each contained in either $H(L^+)$ or $H(L^-)$, since L is not contained in either $H(K_a)$ or $H(K_b)$. Thus both $H(L^+)$ and $H(L^-)$ are partial supporting half planes, so L is a stabbing line of S. □

Lemma 3.3. $\mathrm{Cor}(K_a, K_b)$ *is non-empty if and only if there exists an undirected line parallel to* K_a *and* K_b *that stabs* a *and* b.

Proof. If $\mathrm{Cor}(K_a, K_b)$ is non-empty, then every line contained in $\mathrm{Cor}(K_a, K_b)$ is a stabbing line. Suppose a line L parallel to K_a and K_b stabs both a and b. Let L^+ and L^- represent L oriented in opposite directions. $H(K_a)$ and $H(K_b)$ are each contained in either $H(L^+)$ or $H(L^-)$, thus implying that L stabs S. □

Consider $a \in S$, and let $PS(a)$ denote a continuous interval of $[0, 2\pi)$ such that for every value θ in $PS(a)$ there exists a line L_a with angle θ such that L_a is a partial support line of S. For $a, b \in S$ let $\mathrm{Ant}(a, b)$ denote a continuous interval of $[0, 2\pi]$, such that for every value θ in $\mathrm{Ant}(a, b)$, there exist anti-parallel partial support lines L_a and L_b with angles θ and $\theta + \pi$ respectively. We say that a and b are an *antipodal pair*. For convenience assume that $PS(s_0) = [0, \alpha_1]$, $PS(s_1) = [\alpha_1, \alpha_2], \ldots, PS(s_{k-1}) = [\alpha_{k-1}, \alpha_k]$. We can find an initial antipodal pair $a = s_0$, $b = s_j$, by traversing $KCH(S)$. Subsequent pairs can be found by using two pointers in $KCH(S)$ at a and b. At each iteration we advance one of the pointers so that a and b remain an antipodal pair. We terminate when $a = s_j$ and $b = s_{k-1}$. Since $|KCH(S)|$ is $O(n)$ we conclude that all antipodal pairs are found in $O(n)$ time. This technique is identical to the method proposed by Shamos [13] and subsequently described by Toussaint [17] as the *rotating callipers* algorithm. For each antipodal pair a, b, let $\mathrm{Stab}(a, b)$ denote a continuous interval of $[0, 2\pi]$ so that for every value θ in $\mathrm{Stab}(a, b)$ there exists at least one line with angle θ that

simultaneously stabs both a and b. For every value $\eta \in (\text{Stab}(a, b) \cap \text{Ant}(a, b))$ there exists a non-empty corridor $\text{Cor}(K_a, K_b)$ parallel to a line with angle η. Thus, the stabbing region of S can be described in terms of antipodal pairs and a range of angles for which there exist non-empty corridors. Therefore, by using this method, an application of the convex hull algorithm of Section 2, the stabbing regions of a set of discs can be computed in $O(n \log n)$ time.

A lower bound of $\Omega(n \log n)$ has been shown for stabbing a set of n discs of equal radii [4]. Thus we conclude with the following theorem.

Theorem. *Given a set of discs S the stabbing region of S can be found in $\Theta(n \log n)$ time.*

We should point out that the algorithm that has been presented is very similar to an algorithm that was originally proposed by Atallah and Bajaj [2]. In [2] the stabbing region is defined under a transformed domain. A line l_i is represented by a pair (ρ_i, θ_i), meaning that l_i passes through the point with polar coordinates (ρ_i, θ_i) and has polar angle $\theta_i + \pi/2$. A disc s_i is given by its centre (ρ_i, θ_i) and radius r_i. Thus a line l_i stabs a disc s_i iff:

$$\rho_i \cos(\theta - \theta_i) - r_i \leq \rho \leq \rho_i \cos(\theta - \theta_i) + r_i.$$

Let $f_i(\theta) = \rho_i \cos(\theta - \theta_i) - r_i$ and let $g_i(\theta) = \rho_i \cos(\theta - \theta_i) + r_i$. Then (ρ, θ) is a stabbing line iff:

$$\operatorname*{Max}_{1 \leq i < n} f_i(\theta) \leq \rho \leq \operatorname*{Min}_{1 \leq i < n} g_i(\theta).$$

Let f^* represent the pointwise max of $f_i(\theta)$ and let g^* represent the pointwise max of $g_i(\theta)$. Thus all points ρ below f^* and above g^* represent a stabbing line. Attalah and Bajaj use a divide and conquer approach to compute f^* and g^*. Although not explicitly described, their algorithm would be similar to algorithm Hull of Section 2. One of the advantages of our algorithm is that analytic functions do not have to be evaluated. The correspondence between the stabbing region produced in [2] and that given here can be seen by noticing that each connected component of the intersection of a horizontal line and the region bounded above by f^* and below by g^* corresponds to our definition of a non-empty corridor.

4. Diameter of a set of discs

The *diameter* of a set of points is defined as the greatest inter-point distance. The diameter of a convex set is the greatest distance between parallel lines of support [18]. A well-known method [13] coined the 'rotating callipers' method [17] for determining the diameter of a set of points begins with computing the

convex hull of the points. A pair of parallel lines of support are then rotated about the hull to obtain all antipodal pairs. The diameter is realized by an antipodal pair with maximum distance.

To determine the diameter of a set of discs S, we use $CH(S)$ and angular intervals $\mathrm{Ant}(a, b)$, for antipodal pairs. The diameter is found by using an easy variant of the algorithm of [13].

We can also use $KCH(S)$ to obtain a different maximal measure of the set S. Rather than finding the greatest distance between parallel lines of support, we can use the greatest distance between parallel partial support lines. Again, the 'rotating callipers' method [13] can be used.

5. The intersection of a set of axis parallel cones

Consider a three dimensional Cartesian coordinate system with x, y and z axes. We say that a plane is horizontal if it is parallel to the x, y plane. Let Σ denote a set of three dimensional right circular cones such that the intersection of any horizontal plane and the cones is a set of discs. Each cone can be described by three coefficients a, b, c and the inequality $(x-a)^2+(y-b)^2 \leqslant (z-c)^2$, and either $z-c \geqslant 0$, in which case we are considering upper half-cones, or $z-c \leqslant 0$, and we are considering lower half-cones. Note, we will use cone to denote either a lower or upper half cone. Observe that every set of discs can be represented as a horizontal cross section of such a set of cones.

We show that a description of the intersection of the cones, $I(\Sigma)$, can be obtained in $O(n \log n)$ time as an application of our convex hull algorithm for a set of discs. In [3], a similar problem is addressed where the vertices of the cones are constrained to lie on one plane. This restriction does not apply for this algorithm.

We assume that the cones are in general position. Thus, the intersection of the cones can be characterized by a collection of vertices and edges bounding a volume, where each *vertex* is formed by the intersection of three cones, and each *edge* is formed by the intersection of two cones. We describe the intersection of a set of cones as a list of vertices on the boundary of the volume of intersection. The volume produced by the intersection of cones is convex and unbounded. We proceed with an algorithm to produce the description of a set of upper half-cones. The algorithm to compute the intersection of lower half-cones is symmetric.

Consider a horizontal plane ω such that no vertex of $I(\Sigma)$ appears above ω. Let S_ω denote the set of discs on the cross section of Σ on the plane ω. Let $\mathrm{INT}(S_\omega) = (s_0, s_1, \ldots, s_h)$ denote the arcs as traversed in clockwise order on the boundary of the intersection of S_ω. We will use $h(s)$ to denote the host cone for the disc s. We assume that $s_0 = s_h$. We will use $i(s, t)$ to denote the intersection point of adjacent arcs in $\mathrm{INT}(S_\omega)$. Observe that $\mathrm{INT}(S_\omega)$ corresponds to the cross-section of $I(\Sigma)$ on the plane ω.

Lemma 5.1. *Two arcs s and t are adjacent in* INT(S_ω) *if and only if the arcs are also adjacent in* $KCH(S_\omega)$.

Proof. Assume that s and t are adjacent in INT(S_ω). We show that if there exists an s and t that are adjacent in INT(S_ω) but are not adjacent in $KCH(S_\omega)$, then there exists a vertex of INT(Σ) above the plane $z = \omega$. As illustrated in Fig. 2, $i(s, t)$ is a vertex of INT(S_ω) but $L(s, t)$ is not a partial support line. This implies the existence of a disc, call it u, containing $i(s, t)$ in its interior and lying entirely in $H(L(s, t))$. Therefore, there exists a disc, call it σ, simultaneously tangent and not interior to s, t and u. If the radius of σ is denoted by $r(\sigma)$ then the cones $h(s)$, $h(t)$ and $h(u)$ intersect in their boundaries at a point v_n at the horizontal plane $z = \omega + r(\sigma)$. So v_n is a vertex of $I(\Sigma)$ above the plane $z = \omega$.

We now show that if s and t are adjacent in $KCH(S_\omega)$, but not adjacent in INT(S_ω) then there exists a vertex of $I(\Sigma)$ that lies above the horizontal plane $z = \omega$. Again we consider a third disc u with the property that $i(s, t)$ is not interior to u and u is not entirely contained in $H(L(s, t))$. Consider the disc σ simultaneously tangent and not interior to s, t and u, a vertex of $I(\Sigma)$ is found at the plane $z = \omega + r(\sigma)$ as was shown above.

We have implied by contradiction that if no vertex of $I(\Sigma)$ lies above the horizontal plane $z = \omega$, then the occurrence of $i(s, t)$ in $I(S_\omega)$ and $L(s, t)$ in $KCH(S_\omega)$ coincide. □

Let v denote a horizontal plane that passes below the vertex of every cone in Σ, and let S_v denote the cross section of Σ. Using the results of Section 3, we can obtain $KCH(S_\omega)$ by computing $CH(S_v)$. Once INT(S_ω) is obtained, we can get all of the vertices of $I(\Sigma)$ by using the following algorithm.

Algorithm Sweep
Input: INT(S_ω).
Output: The vertices in INT(Σ).
Denote pred(a) and succ(a) as the predecessor and successor of a in INT(S_ω)

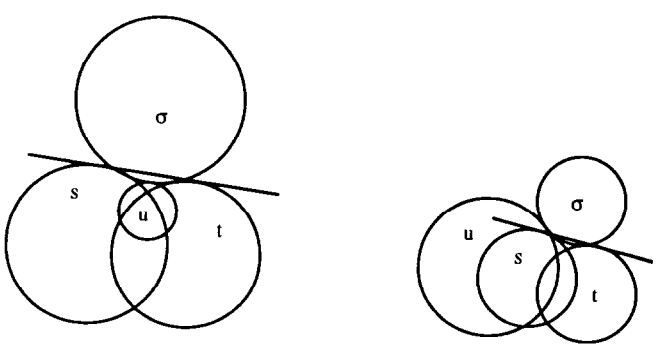

Fig. 2. An illustration showing the relation between INT(S_ω) and $KCH(S_\omega)$.

respectively. Let q(a) denote the intersection point of the cones h(pred(a)), $h(a)$ and h(succ(a)). Let $z(q(a))$ denote the z-coordinate of $q(a)$.

$\Psi \leftarrow \text{INT}(S_\omega)$.

While there are at least 3 arcs in Ψ **do**

Find an arc, a, so that $z(q(a))$ is maximized.

Insert $q(a)$ into INT(Σ).

Remove a from Ψ so that $\Psi \leftarrow \Psi - a$.

We prove that the algorithm Sweep is correct by analyzing the contour of $I(\Sigma)$ above and below one of its vertices.

Lemma 5.2. *Let q be a vertex of $I(\Sigma)$ and let ε be a positive real value such that no other vertex of $I(\Sigma)$ has z-coordinate in the range bounded by planes $z = z(q) + \varepsilon$ and $z = z(q) - \varepsilon$. Consider the planes $\beta = z(q) - \varepsilon$, and let $\alpha = z(q) + \varepsilon$. There exists an arc a in* INT(S_α) *such that q is on the surface of $h(a)$ and* INT(S_β) = INT(S_α) $- a$.

Proof. We assume the cones $h(a_0)$, $h(a_1)$, $h(a_2)$ intersect at the vertex q. Let b_i denote the disc supporting the arc a_i, and let c_i denote the centre of b_i. Let q_α represent the orthogonal projection of q onto the plane $z = \alpha$, that is, $q_\alpha = (x(q), y(q), z(q) + \varepsilon)$. Extend lines through c_i, and q_α to the boundary of b_i for $i = 0, 1, 2$, to obtain points p_i as shown in Fig. 3. Observe that $d(q_\alpha, p_2) = d(q_\alpha, p_1) = d(q_\alpha, p_0) = \varepsilon$. Therefore, the angles $\theta_1 = \theta_2$ and $\theta_3 = \theta_4$. Considering the triangle $\Delta c_2 p_2 p_1$, the angle at p_1 is larger than the angle at p_2, which implies that $d(c_2, p_1) < d(c_2, p_2)$. Similarly, we can show that $d(c_0, p_1) < d(c_0, p_0)$. Therefore, the point p_1 is interior to the intersection of b_0, b_1, b_2. This implies that a_1 is an arc of INT(S_α). A similar construction can be used to show that a_1 is

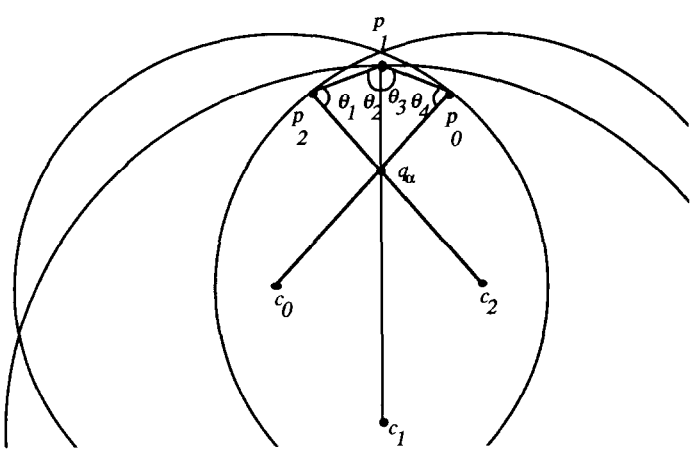

Fig. 3. Illustrating the proof of Lemma 5.2.

not an arc of $\mathrm{INT}(S_\beta)$. Since we have assumed there are no vertices of $I(\Sigma)$ between the planes $z = \alpha$ and $z = \beta$ we have $\mathrm{INT}(S_\beta) = \mathrm{INT}(S_\alpha) - a$. □

At every iteration of algorithm Sweep the list Ψ represents the boundary of the contour of $\mathrm{INT}(S)$ at a horizontal cross section. An inductive argument based on the previous lemma leads to the main result of this section.

Theorem. *Given* S_ω *algorithm Sweep obtains* $\mathrm{INT}(\Sigma)$ *in* $O(n \log n)$ *time, where* $n = |S_\omega|$.

As was shown at the beginning of the section we can obtain S_ω in $O(n \log n)$ time by using the convex hull for discs algorithm. Thus the intersection of a set of axis parallel cones can be computed in $O(n \log n)$ time.

6. The intersection of a set of discs

We consider the problem of computing the intersection of a set of discs. The intersection of n planar discs can be found in $O(n \log n)$ time using a technique described by Brown [5]. Brown's method requires transforming a two dimensional problem into a three dimensional problem of intersecting half-planes, and then transforming back to obtain the intersection of discs.

We demonstrate a method that is more straightforward to obtain the same results. Our method is based on the previous discussion. We show that the intersection of a set of discs can be obtained by using the convex hull algorithm for discs.

Let $I(S)$ denote the intersection region of a set of discs S. We can describe this region by a list of arcs of discs $\mathrm{INT}(S) = (s_0, s_1, \ldots, s_h)$, and as before to avoid having to use a circular list we set $s_0 = s_h$.

Every collection of discs represents a cross section of cones as shown in the previous section. As an artifact of the method presented in Section 5 the intersection regions of all cross sections of a base set of cones is obtained. Our method should now be clear. We obtain a cross section above all vertices of the base set of cones, and in particular above the cross section of our input set of cones. We then apply algorithm Sweep until we arrive at our desired cross section, and the sequence Ψ corresponds to $\mathrm{INT}(S)$.

7. Minimum spanning circle, and the furthest site Voronoi diagram

The notion of the classic Voronoi diagram can be generalized to the so called *furthest site Voronoi diagram.* Whereas the classic Voronoi diagram describes the locus of points closest to a site, the furthest site diagram finds the locus of points

furthest from a site. Typically, algorithms that use the divide and conquer approach can be easily modified to yield algorithms for computing the furthest site Voronoi diagram. Fortune's algorithm, computing the intersections of cones, does not have this property. The minimum spanning circle of a set is a smallest circle that intersects the set. Shamos [12] showed that by first computing the furthest site Voronoi diagram for a set of points, the minimum spanning circle of a set of points could be found in $O(n \log n)$ time. Megiddo [10] subsequently showed that the minimum spanning sphere of a set of balls in $\mathbb{R}^d$ can be found in linear time. We show that the vertices of the furthest site Voronoi diagram can also be represented as a set of vertices on the intersection of a set of axis parallel cones. We will use this fact to develop an $O(n \log n)$ algorithm to compute the furthest site Voronoi diagram for a set of discs. These results have previously appeared in [11]. The method used is an application of a convex hull algorithm for a set of discs and will be described.

In order to discuss the properties of a Voronoi diagram, we must define how the distance from a point to a disc is measured. Thus, given point p and a disc s with centre $c(s)$ and radius $r(s)$ we define $\Delta(p, s)$ as $d(p, c(s)) + r(s)$. Alternatively, we can define the distance from a point p to a disc s as $\Phi(p, s) = d(p, c(s)) - r(s)$. We can compute the Voronoi diagram for a set of discs by using either of the distance functions Δ or Φ. Note, by using Φ we may obtain a negative distance, so strictly speaking the definition is somewhat unsatisfactory with regards to distance functions. However, for our purposes, this does not present any problems. We will use the Δ distance function, with all results carrying over directly to the Φ distance function. Let FVOR(S) denote the furthest site Voronoi diagram of a set of discs S using the Δ distance function. Define $\text{freg}(s) = \{p \in \mathbb{R}^2 / \Delta(p, s) \geq \Delta(p, s') \text{ for all } s' \text{ in } S\}$. That is freg($s$) denotes the region of FVOR(S) associated with the site $s \in S$. FVOR(S) is the uniton of freg(s) for all $s \in S$. Note that freg(s) may be empty for some sites s and freg(s) need not be connected for others. See Fig. 4. Each Voronoi region is bounded by the arc of an hyperbola. We will call the arcs bounding Voronoi regions *Voronoi edges*. We will call the points found at the intersection of two or more Voronoi edges, *Voronoi vertices*. We list some properties of FVOR(S).

Lemma 7.1. *A Voronoi vertex v is adjacent to exactly three Voronoi regions, and is the centre of a spanning circle of S that is internally tangent to the discs in S corresponding to those regions.*

Proof. Follows from the definition and that no four discs are tangent to the same circle. □

Let $\Sigma = \{h_1, h_2, \ldots, h_n\}$ denote a set of axis parallel cones that represent the set of discs S. That is, for each disc $s_i \in S$ with centre (a_i, b_i) and radius r_i there is a cone h_i defined by the inequality $((x - a_i)^2 + (y - b_i)^2)^{1/2} \leq (z - r_i)$. Observe

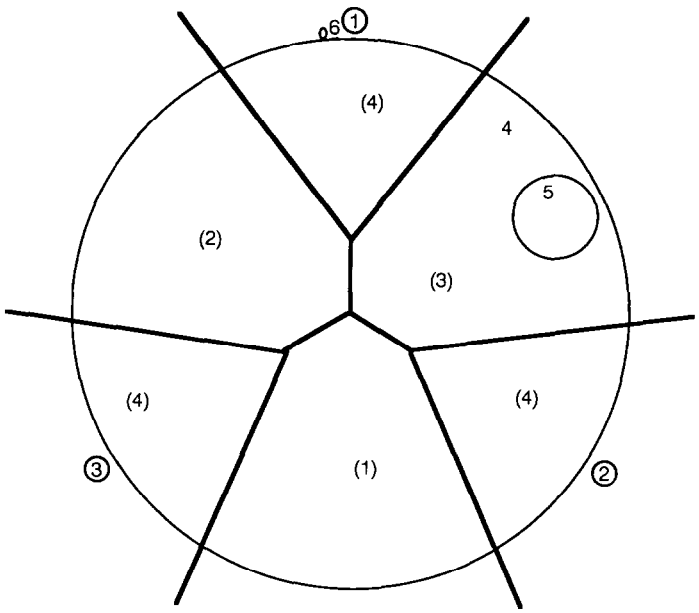

Fig. 4. There are no regions freg(5), and freg(6). The region freg(4) consists of three disconnected components.

that the intersection of the lower half-cones of Σ and the plane $z = 0$ is the set of discs S. Let $I(\Sigma)$ represent the intersection of the cones Σ above the plane $z = 0$, that is, the intersection of the upper half-cones. As defined in Section 5 the intersection of two cone surfaces will be called an *edge*, and the intersection at three cone surfaces will be called a *vertex*.

Lemma 7.2. *The parallel projection of a vertex v on the boundary of $I(\Sigma)$ onto the plane $z = 0$ is a Voronoi vertex of* FVOR(S). *The parallel projection of an edge on the boundary of $I(\Sigma)$ onto the plane $z = 0$ is a Voronoi edge of* FVOR(S).

Proof. The intersection point, $v = (x, y, z)$, of three cones $h_1, h_2, h_3 \in \Sigma$, is the solution to the system of equations:

$$((x - a_i)^2 + (y - b_i)^2)^{1/2} = (z - r_i), \quad \text{for } i = 1, 2, 3.$$

The centre of a circle of radius r spanning the three discs $s_1, s_2, s_3 \in S$ is the solution to the equations:

$$((x - a_i)^2 + (y + b_i)^2)^{1/2} = (r - r_i), \quad \text{for } i = 1, 2, 3.$$

Clearly the equations are the same, and the z-coordinate of a vertex v corresponds to r, the radius of the spanning circle.

Similarly, we can show that edges of $I(\Sigma)$ project to Voronoi edges. □

As shown in Section 5 we can compute INT(Σ) in $O(n \log n)$ time as an application of the convex hull algorithm.

The algorithm we have presented can be used to compute the smallest radius spanning circle of a set of discs [11]. Megiddo [10] has shown that the minimum radius spanning circle of a set of discs can be found in $O(n)$ time. However, this method requires solving a linear programming problem in three dimensions. The algorithm, although it is asymptotically linear, is quite complex and has a fairly large constant of computation. The constant of computation is asymptotically $O(2^{2^d})$, where d is the dimension of the linear program. On the other hand, the methods described in this paper are conceptually much easier. Thus practical implementations of our method should be much easier to obtain.

8. Discussion

The disc is an important basic geometric object. The centrally symmetric structure of the disc was exploited to contribute to the collection of diverse results we have obtained. In every case we use the convex hull of the discs as a method to impose some structure on the input. This structure can then be used to efficiently search for other properties, depending on the applications.

Recently, Affentranger and Dwyer [1] have studied the size of the expected number of circles on the convex hull of a set of circles. They consider circles chosen randomly from the uniform distribution from a unit disc under three different models. In the first model, a random circle is defined by three randomly chosen points in the unit disc. In the second model, three lines intersecting the unit disc are picked randomly. The incircle of the triangle obtained by the intersection points of the lines is the random circle. In models one and two Affentranger and Dwyer show that the expected number of circles on the convex hull of the circles is $O(1)$. In the third model, two points are chosen randomly from a unit disc, and these points serve as the diameter of the random circle. The expected number of circles on the convex hull of the circles under the third model is $O(n^{1/6})$. The divide and conquer approach we use works on subsets of the input which are all from the same probability distribution. Thus the merge time per recursive step is $O(n^{1/6})$ in expectation for all three models described above. This leads to an $O(n)$ expected time complexity to compute the convex hull of discs. For every application of the convex hull algorithm, we use an additional step that is $O(k \log k)$ where k denotes the number of arcs on the convex hull. Thus the expected time complexity for the algorithms that have been presented is $O(n)$. As a result of this analysis, coupled with the inherent simplicity of the algorithms, implementations should perform favourably in practice, and they should be quite easy to program.

Acknowledgements

I would like to thank Chee Yap for carefully reading the manuscript, and for his helpful commments. I would also like to acknowledge two anonymous referees for their valuable suggestions have improved the presentation.

References

[1] F. Affentranger and R. Dwyer, The convex hull of random circles, Technical Report TR-91-01, Dept. of Computer Science, North Carolina State University, January 1991.
[2] M. Atallah and C. Bajaj, Efficient algorithms for common transversals, Inform. Process. Lett. 25 (1987) 87–91.
[3] F. Aurenhammer, Power diagrams: Properties, algorithms and applications, SIAM J. Comput. 16 (1) (1987) 78–96.
[4] D. Avis, J.-M. Robert and R. Wenger, Lower bounds for line stabbing, Inform. Process. Lett. 33 (1989) 59–62.
[5] K.Q. Brown. Geometric transforms for fast geometric algorithms, Ph.D. Dissertation Carnegie-Mellon University, 1979.
[6] H. Davenport and A. Schinzel, A combinatorial problem connected with differential equations, Amer. J. Math. LXXXVII (3) (1965) 684–694.
[7] H. Edelsbrunner, H.A. Maurer, F.P. Preparata, A.L. Rosenberg, E. Welzl and D. Wood, Stabbing line segments, BIT 22 (1982) 274–281.
[8] S. Fortune, A sweepline algorithm for Voronoi diagrams, Algorithmica 2 (2) (1987) 153–174.
[9] D. Kirkpatrick, Efficient computation of continuous skeletons, Proc. 20th IEEE Symp. on Foundations of Computer Science (1979) 18–27.
[10] N. Megiddo, On the ball spanned by balls, Discrete Comput. Geom. (6) (1989).
[11] D. Rappaport, Computing the furthest site Voronoi diagram for a set of discs, Tech. Report no. 89-250, Dept. of Computing and Information Science, Queen's University, 1989.
[12] M. Shamos, Computational Geometry, Ph.D. Dissertation, Yale University, 1975.
[13] M. Shamos and D. Hoey, Closest point problems, Proc. 16th IEEE Symp. on Foundations of Computer Science (1975) 151–162.
[14] M. Sharir, Intersection and closest-pair problems for a set of planar discs, SIAM J. Comput. 14 (2) (1985) 448–468.
[15] M. Sharir, R. Cole, K. Kedem, D. Leven, R. Pollack and S. Sifrony, Geometric applications of Davenport–Schinzel sequences, Proc. 26th Ann. Symp. on Foundations of Computer Science (1986) 77–86.
[16] E. Szemeredi, On a problem of Davenport and Schinzel, Acta Arithmetica XXV (2) (1974) 213–224.
[17] G.T. Toussaint, Solving geometric problems with the *rotating callipers,* Proc. of IEEE MELECON '83, Athens, Greece (1983)
[18] I.M. Yaglom and V.G. Boltyanskii, Convex Figures, (Holt, Rinehart and Winston, New York, 1961).
[19] C.K. Yap, An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments, Discrete Comput. Geom. 2 (1987) 365–393.