Widest-Corridor Problems

Ravi Janardan* Franco P. Preparata†

June 2, 1993

1 Introduction

Let S be a set of n points in the Euclidean plane. A *corridor* through S is the open region of the plane that is bounded by two parallel straight lines that intersect the convex hull, CH(S), of S. The *width* of a corridor is the distance between the bounding lines. A corridor is called *k-dense* if it contains k points of S, where $0 \leq k \leq n-2$. (As we will see later, each of the bounding lines must contain at least one point of S, and hence $k \leq n-2$.) A *widest k-dense corridor* through S is a k-dense corridor of maximum width.

The problem of constructing a widest k-dense corridor arises in robot motion-planning. In [HM88], Houle and Maciel gave an $O(n^2)$-time and $O(n)$-space algorithm. Subsequently, Chattopadhyay and Das [CD90] showed how to compute a widest k-dense corridor, $0 < k \leq n-2$, in $O(n^2 \log n)$ time and $O(n^2)$ space. In this paper, we present efficient algorithms for several versions of the widest k-dense corridor problem. Our results include:

1. A simple sweepline algorithm to compute a widest k-dense corridor through S in $O(n^2 \log n)$ time and $O(n)$ space, where $0 \leq k \leq n-2$. This improves upon the space bound in [CD90]. Moreover, using this approach we can (i) compute a widest k-dense corridor for each k, where $k = 0, 1, \ldots, n-2$, in $O(n^3)$ time and $O(n)$ space and (ii) compute in $O(n^2 \log n)$ time and $O(n)$ space a minimum-density corridor of width at least w for any given real number w (provided such a corridor exists for the given w).

2. An algorithm to dynamically maintain a widest empty corridor in $O(n \log n)$ time and $O(n^2)$ space as points are inserted and deleted online in S. This is considerably more efficient than recomputing a widest empty corridor from scratch after each update.

3. An algorithm to compute a widest empty corridor through a set of polygonal obstacles in the plane in $O(n^2)$ time and $O(n)$ space, where n now is the total number of edges of the polygons.

4. An $O(kn^2)$-time and $O(kn^2)$-space algorithm for computing a widest k-dense corridor through S, $0 < k \leq n-2$. This algorithm is faster than our algorithm in 1 above for small k, i.e., when $k = o(\log n)$; however, it uses more space.

5. A *k-dense closed corridor* through S is the *closed* region of the plane that is bounded by two parallel lines that intersect CH(S) and which contains k points of S, where $2 \leq k \leq n$. For $2 \leq k \leq n-2$, our results (and those of [CD90]) carry over to k-dense closed corridors as well. As observed in [CD90], a widest n-dense closed corridor is determined by a diametral pair of S and hence can be computed in $O(n \log n)$ time and $O(n)$ space [PS88]. Here we show how to also compute a widest $(n-1)$-dense closed corridor in $O(n \log n)$ time and $O(n)$ space.

Our approach in Results 1–4 are based on geometric duality and on the topological-sweep paradigm [EG89]. Result 5 is based on an efficient technique for searching in the convex layers of a planar point-set. Due to space constraints, we discuss here only Results 2–5 and omit all proofs and many details. We refer the reader to [JP93] for the full paper.

2 Preliminaries

The following theorem states a key property of widest k-dense corridors.

*Dept. of Computer Science, University of Minnesota, Minneapolis, MN 55455. Email: janardan@cs.umn.edu. Research supported in part by NSF grant CCR–92–00270.

†Dept. of Computer Science, Brown University, Providence, RI 02912. Email: franco@cs.brown.edu. Research supported in part by NSF grant CCR–91–96176.

Theorem 2.1 ([HM88, CD90]) *Let C^* be a widest k-dense corridor through S, with bounding lines ℓ' and ℓ''. Then one of the following conditions must hold:*

(a) *One of the lines, say ℓ', passes through two points p_i and p_j of S and ℓ'' passes through a point p_k of S,* or

(b) *there are points p_i and p_j of S such that ℓ' passes through p_i, ℓ'' passes through p_j, and ℓ' and ℓ'' are perpendicular to the line passing through p_i and p_j.* □

Let us reinterpret conditions (a) and (b) in the dual plane, under the standard duality transform $\mathcal{F}$, which maps a point $p = (a, b)$ to the line $\mathcal{F}(p) : y = ax - b$ and the non-vertical line $\ell : y = mx - c$ to the point $\mathcal{F}(\ell) = (m, c)$. Let H be the set of lines $\ell_i = \mathcal{F}(p_i)$, where $p_i \in S$. Let $\mathcal{A}$ be the arrangement of H. Let $v_{ij} = \ell_i \cap \ell_j$ be any vertex of $\mathcal{A}$ and let $x(v_{ij})$ denote its abscissa. Let C be any k-dense corridor through S. It is easy to show that:

If C is a type (a) corridor, then $\mathcal{F}(\ell') = v_{ij}$ and $\mathcal{F}(\ell'')$ is the point where the vertical line through v_{ij} intersects ℓ_k. If C is a type (b) corridor, then $\mathcal{F}(\ell')$ and $\mathcal{F}(\ell'')$ are points on ℓ_i and ℓ_j, respectively, where both points have abscissa $-1/x(v_{ij})$. Moreover, if C is a type (a) (resp. type (b)) corridor and ℓ is any line that is parallel to ℓ' and ℓ'' and contained in C, then $x(\mathcal{F}(\ell)) = x(v_{ij})$ (resp. $x(\mathcal{F}(\ell)) = -1/x(v_{ij})$) and $y(\mathcal{F}(\ell')) < y(\mathcal{F}(\ell)) < y(\mathcal{F}(\ell''))$.

Thus, in either case, the dual of C, which we denote by $\mathcal{F}(C)$, is the open vertical line segment bounded by $\mathcal{F}(\ell')$ and $\mathcal{F}(\ell'')$. Moreover, C is a k-dense corridor through S if and only if $\mathcal{F}(C)$ intersects k lines of H. Also, the width of C is $|y(\mathcal{F}(\ell')) - y(\mathcal{F}(\ell''))|/\sqrt{1 + x(\mathcal{F}(C))^2}$.

We can now compute C^* as follows: We consider each vertex v_{ij} of $\mathcal{A}$ in turn, find the $(k+1)$st line vertically above it (if it exists), and compute the corresponding type (a) k-dense corridor. Similarly for the $(k+1)$st line vertically below v_{ij}. We also check whether at abscissa $-1/x(v_{ij})$ there are exactly k lines between ℓ_i and ℓ_j (excluding ℓ_i and ℓ_j themselves) and, if so, then we compute the corresponding type (b) k-dense corridor. Throughout the search, we keep track of the widest corridor found so far and update this information whenever a wider corridor is found.

3 Widest empty corridors

3.1 Widest empty corridor through a planar point-set

We give an $O(n^2)$-time and $O(n)$-space algorithm to compute a widest empty corridor C^*. Except for minor differences, the algorithm is analogous to the one given by Houle and Maciel [HM88]. We present it here for completeness since our results in Sections 3.2, 3.3, and 4 use some of the ideas.

Let C be an empty corridor. Since $k = 0$, both endpoints of $\mathcal{F}(C)$ lie on the boundary of the same face, f, of $\mathcal{A}$. Thus the search for C^* can be restricted to the faces of $\mathcal{A}$.

We construct the arrangement $\mathcal{A}$ in $O(n^2)$ time and space using the algorithm of [CGL85]. To compute the empty type (a) corridors, C, corresponding to f, we decompose f's boundary into a lower chain, L, and an upper chain, U, whose endpoints are the leftmost and rightmost vertices of f. For each interior vertex on U, we find the edge of L vertically below it in $O(|f|)$ total time, as follows: For the leftmost interior vertex of U, we scan L from left to right and stop as soon as we find an edge vertically below the vertex. By convexity, for each subsequent interior vertex on U, the scan of L can be resumed from its previous position on L. Similarly, we can compute for each interior vertex of L the first edge vertically above it. The total time is $O(|L| + |U|) = O(|f|)$.

We determine the empty type (b) corridors corresponding to f as follows: Consider the sorted list of abscissas of the vertices of f and let I be some open interval defined by consecutive abscissas in the list. With I we can associate two lines, ℓ_i and ℓ_j, where wlog ℓ_i supports an edge of U and ℓ_j supports an edge of L, such that for any abscissa in I, ℓ_i is the first line vertically above ℓ_j and ℓ_j is the first line vertically below ℓ_i. Thus we only need to check whether $-1/x(v_{ij}) \in I$, which can be done in constant time given ℓ_i and ℓ_j.

To determine the lines associated with each interval, we merge L's and U's vertices (which, by convexity, are in sorted x-order) into a single sorted list, $v_1, v_2, \ldots, v_{|f|}$, where v_1 is the leftmost vertex of f. Assume that v_1 is the intersection of lines ℓ_p and ℓ_q, where ℓ_p is above ℓ_q to the immediate right of v_1. Then the lines associated with the interval $(x(v_1), x(v_2))$ are ℓ_p and ℓ_q. Now v_2 must be the intersection of one of the lines ℓ_p and ℓ_q, say ℓ_p, with a new line ℓ_r. Thus ℓ_q and ℓ_r are associated with $(x(v_2), x(v_3))$. And so on. The total time is clearly

$O(|f|)$.

Thus, the total time is $O(\sum_{f \in \mathcal{A}} |f|) = O(n^2)$. The space is $O(n^2)$ as all of $\mathcal{A}$ is stored.

To reduce the space to $O(n)$, we compute only only the portions of $\mathcal{A}$ that are relevant at any given time. For this, we use the topological sweep paradigm of Edelsbrunner and Guibas [EG89], which allows us to visit the vertices, edges, and faces of an arrangement in a systematic way in $O(n^2)$ time and $O(n)$ space. Moreover, as soon as a face f is reached in the sweep, its vertices and edges can be extracted in $O(|f|)$ time from the supporting data structures. Thus, we can process f in $O(|f|) = O(n)$ time and space and then reuse the space for the next face.

Theorem 3.1 ([HM88]) *A widest empty corridor through a set of n points in the plane can be computed in $O(n^2)$ time and $O(n)$ space.* □

3.2 Dynamic maintenance of a widest empty corridor

We now describe how the widest empty corridor through S can be updated online as points are inserted into or deleted from S. Our approach is based on the following properties:

1. When a point p is inserted into S, an empty corridor C is destroyed only if the endpoints of $\mathcal{F}(C)$ lie on the boundary of a face of $\mathcal{A}$ that is intersected by $\mathcal{F}(p)$. Moreover, if $\mathcal{A}'$ is the arrangement after the insertion of $\mathcal{F}(p)$, then the search for new empty corridors can be restricted to those faces of $\mathcal{A}'$ to which $\mathcal{F}(p)$ contributes an edge. Symmetrically for deletions.

2. In an arrangement of n lines, the sum of the sizes of the faces intersected by any line is $O(n)$ [CGL85]. This implies that only $O(n)$ empty corridors are affected by an update.

In addition to $\mathcal{A}$, we also maintain a heap, H, which stores all the empty corridors of S. For each empty corridor of S, with bounding lines ℓ' and ℓ'' and width w, there is an entry in H consisting of ℓ', ℓ'', and w. H is organized as a max-heap on the w's; thus the widest empty corridor can be retrieved in $O(1)$ time following an update.

With each vertex v_{ij} of $\mathcal{A}$ can be associated up to three empty corridors, namely: (i) one for which v_{ij} is the lower endpoint of the dual vertical line segment, (ii) one for which v_{ij} is the upper endpoint, and (iii) one for which the endpoints of the dual line segment at abscissa $-1/x(v_{ij})$ lie on the lines ℓ_i and ℓ_j. We store with v_{ij} a pointer to each of the three associated corridors in H. With this setup, if one of the corridors associated with v_{ij} is to be deleted, its position in H can be determined in $O(1)$ time. Since, H has size $O(n^2)$, it supports insertions and arbitrary deletions (given the position of the entry to be deleted) in $O(\log n)$ time.

Consider the insertion of a point p. Starting with, say, the leftmost intersection of $\mathcal{F}(p)$ with $\mathcal{A}$, we successively traverse the boundaries of the faces intersected by $\mathcal{F}(p)$ and update the representation for $\mathcal{A}$ by adding the edges and vertices induced by the intersection of $\mathcal{F}(p)$ with $\mathcal{A}$. This takes $O(n)$ time because of property 2 above. During the traversal, we also do the following: Suppose that $\mathcal{F}(p)$ splits a face f into faces f_1 and f_2. We walk around the boundary of f, find (as in Section 3.1) the corridors whose duals have both their endpoints on the boundary of f, and delete from H those corridors for which one endpoint of the dual is on the boundary of f_1 and the other endpoint is on the boundary of f_2. We then find the corridors whose duals have both their endpoints on the boundary of f_1 (and similarly for f_2) and insert these into H. This takes $O(|f| \log n)$ time, which when summed up over all faces f intersected by $\mathcal{F}(p)$ yields a time bound of $O(n \log n)$, by property 2 above. Deletion of a point p is essentially the reverse.

Theorem 3.2 *A widest empty corridor through a planar point-set S can be maintained dynamically, under online insertions and deletions in S, in $O(n \log n)$ time and $O(n^2)$ space, where n is the current size of S.* □

3.3 Widest empty corridor through polygonal obstacles

We show how to compute a widest empty corridor through a set, P, of (not necessarily simple) polygonal obstacles in the plane in $O(n^2)$ time and $O(n)$ space, where n now is the total number of edges in the obstacles. Formally, a corridor, C, through P is the open region of the plane that is enclosed by the two parallel straight lines that intersect the convex hull of the vertices of the polygons in P and such that the region does not intersect any polygon in P. (Note that for a given P it is possible that no empty corridor exists.) It is easy to prove that if C^* is a widest empty corridor through P, with bounding lines ℓ and ℓ', then one of the conditions (a) or (b) of Theorem 2.1 must

hold, where p_i, p_j, and p_k are now vertices of polygons in P.

Let T be the set of edges of the polygons. Under $\mathcal{F}$, a segment $t \in T$, with endpoints p and p', is mapped to the doublewedge, $W(t)$, formed by $\mathcal{F}(p)$ and $\mathcal{F}(p')$ and not containing the vertical line through the point $\mathcal{F}(p) \cap \mathcal{F}(p')$. Furthermore, a line ℓ intersects t if and only if $\mathcal{F}(\ell)$ lies in $W(t)$.

Let $\mathcal{A}$ be the arrangement of the lines bounding the doublewedges $W(t)$ for $t \in T$. It is well-known that lines ℓ_1 and ℓ_2 intersect the same subset of segments of T (and hence the same number of segments of T) if and only if $\mathcal{F}(\ell_1)$ and $\mathcal{F}(\ell_2)$ lie in the same face, f, of $\mathcal{A}$. Let $count(f)$ denote the number of line segments of T intersected by any line whose dual point falls in f. Of particular interest to us are the *count-zero* faces of $\mathcal{A}$, i.e., faces f such that $count(f) = 0$, since an open vertical line segment whose endpoints lie on the boundary of a count-zero face is the dual of a corridor that intersects no line segment in T (and hence no polygon in P).

Thus, if C is an empty corridor through P, then both endpoints of $\mathcal{F}(C)$ lie on the boundary of the same count-zero face of $\mathcal{A}$. To find C^*, we use the topological-sweep-based algorithm described in Section 3.1, but perform corridor computations only for the count-zero faces of $\mathcal{A}$. In order to identify the count-zero faces of $\mathcal{A}$ during the sweep, we use a technique of Edelsbrunner and Guibas [EG89, page 182], who show how to compute $count(f)$ in $O(1)$ time for each face f of $\mathcal{A}$ when it is first encountered in the sweep.

Theorem 3.3 *A widest empty corridor through a set of (not necessarily simple) polygonal obstacles in the plane can be computed in $O(n^2)$ time and $O(n)$ space, where n is the total number of edges of the polygons.* □

4 Computing a widest k-dense corridor for $k > 0$

We show how to compute a widest k-dense corridor, in $O(kn^2)$ time and $O(kn^2)$ space, where $k > 0$. Thus, when $k = o(\log n)$, this algorithm is faster than the one mentioned as Result 1 in the Introduction (and omitted here due to page limitations); however, it uses more space.

As in Section 3.1, we process $\mathcal{A}$ face-by-face, determining for each vertex on the upper chain of a face the $(k+1)$st line, if any, that is vertically below it. Similarly, in a second face-by-face pass, we determine the $(k+1)$st line vertically above each vertex. In a third pass, we determine type (b) corridors.

However, unlike Section 3.1, the $(k+1)$st lines below and above a vertex do not lie on faces containing the vertex but are instead several faces away, and so the processing involves cutting across face boundaries. To do this efficiently and systematically, we process faces in a certain order. Towards this end, following [EGS86], we define a total order on the faces, as follows:

For distinct faces f and g of $\mathcal{A}$, say that $f \gg g$ if there is a vertical line that intersects both f and g, f above g. It is well-known [EGS86] that the relation $\gg$ is acyclic. Consider the subset $\succ$ of $\gg$ consisting of those pairs (f, g) that share an edge. We can compute $\succ$ in $O(n^2)$ time by traversing the boundary of each face f and determining for each edge e of f the other face in $\mathcal{A}$ that shares e with f. The relation $\succ$ has cardinality $O(n^2)$ and, moreover, its transitive closure coincides with that of $\gg$. The desired total order is obtained by doing a topological sort on $\succ$, which takes $O(n^2)$ time.

Let $f_1, f_2, \ldots, f_s$ be the faces of $\mathcal{A}$ under this total order, where f_1 (resp. f_s) is the topmost (resp. bottommost) face of $\mathcal{A}$. With the exception of these two faces, each f_i can be decomposed into a lower chain L_i and an upper chain U_i; faces f_1 and f_s consist of only a lower chain L_1 and an upper chain U_s, respectively.

The first two passes

We begin with face f_2 and, as in Section 3.1, proceed to map each arrangement vertex on U_2 to a point vertically below it on L_2. In general, let f_i be the face to be processed next. The set, P_i, of points on U_i that must be resolved (i.e., mapped to points vertically below on L_i) consists of (i) arrangement vertices lying on U_i and (ii) downward projections of arrangement vertices lying on upper chains U_j, where $j < i$. With each such point $p \in P_i$, we associate a pointer, $vert(p)$, to the arrangement vertex that projects to p and a level number, $level(p)$, where $level(p) = h$ implies that $vert(p)$ has been projected vertically downwards h times, where $0 \leq h \leq k+1$. In particular, if p is an arrangement vertex lying on U_i, then $vert(p) = p$ and $level(p) = 0$.

To resolve P_i, we traverse the boundary of U_i and the boundary of L_i simultaneously and for each point $p \in P_i$ we do the following: If $level(p) = k+1$ then the open vertical line segment with endpoints p and

vert(p) intersects k lines and thus is the dual of a type (a) corridor. We compute this corridor and then discard p. If, however, *level*$(p) < k+1$, then we reset p to the point vertically below it on L_i and increment *level*(p).

The processing of faces in the first pass terminates once f_{s-1} has been processed. To calculate the $(k+1)$st line vertically above each arrangement vertex, we perform a symmetric second pass, processing faces in the reverse order. In addition, we determine for each vertex of the arrangement the kth line vertically above it. The reason for this will become clear in the sequel.

The third pass

We perform a face-by-face pass, from f_1 to f_s, which simulates the first pass but, in addition, also does the following: Let f_i be the current face and let $Q_i = p_1, p_2, \ldots, p_q$ be the left–to–right sequence of points consisting of the arrangement vertices on L_i and of the points that have been projected down from U_i, where p_1 is the leftmost arrangement vertex on L_i. Let p_1 be the intersection of lines ℓ_x and ℓ_y, where ℓ_x supports an edge of U_i and ℓ_y supports an edge of L_i. Let ℓ_z be the kth line above p_1. Now, for any abscissa that is infinitesimally to the right of p_1, there are k lines between ℓ_y and ℓ_z. This is because at $x(p_1)$, there are $k-1$ lines between p_1 and ℓ_z and just to the right of p_1, ℓ_x contributes one to this count.

Let $j > 1$ be the smallest index such that *vert*(p_j) is the intersection of one of ℓ_y and ℓ_z, say ℓ_y, with a new line ℓ_w. Then, for any abscissa in the interval $(x(p_1), x(p_j))$, there are k lines between ℓ_y and ℓ_z. Let v_{yz} be the arrangement vertex defined by the intersection of ℓ_y and ℓ_z. If $-1/x(v_{yz}) \in (x(p_1), x(p_j))$, then an empty type (b) corridor has been found. Again, for any abscissa that is infinitesimally to the right of p_j, there are k lines between ℓ_z and ℓ_w and so we proceed to determine, as before, the interval $(x(p_j), x(p_{j'}))$ for which this property holds. The processing of f_i in the third pass terminates once p_q is reached.

In each pass, the time to scan a face f_i is $O(|f_i|)$, which sums up to $O(n^2)$ for all faces. The time to resolve a vertex is $O(k)$, which is $O(kn^2)$ for all vertices. The space is clearly $O(kn^2)$.

Theorem 4.1 *A widest k-dense corridor through a set of n points in the plane can be computed in $O(kn^2)$ time and $O(kn^2)$ space, where $0 < k \leq n-2$.* □

5 Computing a widest $(n-1)$-dense closed corridor efficiently

Since any $(n-1)$-dense closed corridor through S excludes exactly one point of S, the excluded point must be a vertex of the convex hull, CH(S), of S. Let $p_0, p_1, \ldots, p_{h-1}$ be the vertices of CH(S). One strategy is to delete each hull vertex p_i in turn and compute a widest $(n-1)$-dense closed corridor through $S_i = S - \{p_i\}$, i.e., a widest closed corridor that includes all the points of S_i. As observed in [CD90], such a corridor is determined by a diametral pair of CH(S_i), which can be computed in time linear in the size of CH(S_i) [PS88] once CH(S_i) is available.

Computing CH(S_i) from scratch for each hull vertex p_i deleted will lead to an overall time of $O(n^2 \log n)$ in the worst case. Fortunately, there is enough coherence in the successively considered collections of points that a substantially more efficient algorithm can be developed.

Our algorithm is based on the notion of convex layers [PS88]. For our purposes, the first two layers, $L_0 = \text{CH}(S)$ and $L_1 = \text{CH}(S - L_0)$, suffice. When p_i is deleted, CH(S_i) can be obtained by simply attaching between p_{i-1} and p_{i+1} a suitable subchain, $H(p_i)$, of L_1. (Throughout, indices are taken modulo h.) $H(p_i)$ is defined as follows: If the line segment $\overline{p_{i-1}p_{i+1}}$ does not intersect L_1, then $H(p_i)$ is simply this line segment. Otherwise, the first (resp. last) edge of $H(p_i)$ is the line segment $\overline{p_{i-1}p'}$ (resp. $\overline{p_{i+1}p''}$) that is tangential to L_1 at the vertex p' (resp. p'') of L_1. Of the two possible such tangential edges from p_{i-1} (resp. p_{i+1}) to L_1, we take the one that makes the smaller acute angle with the edge $\overline{p_{i-1}p_i}$ (resp. $\overline{p_{i+1}p_i}$). See FIGURE 4 for an example.

For future use, we now establish that, for $i \neq j$, $H(p_i)$ and $H(p_j)$ are edge-disjoint. Clearly, it suffices to show that $H(p_i)$ and $H(p_{i+1})$ have this property. Referring to FIGURE 4, note that p_i must lie in the region r facing bay b for otherwise L_0 intersects L_1. Thus the tangent from p_i to L_1 (the one which contains the first clockwise edge of $H(p_{i+1})$) cannot meet L_1 at any interior vertex of $H(p_i) \cap L_1$. This establishes the claim.

For any vertex $p_i \in \text{CH}(S)$, a diametral pair of CH(S_i) is one of the *antipodal pairs* of vertices of CH(S_i), i.e., a pair of vertices through which parallel lines supporting CH(S_i) can be drawn [PS88]. Clearly, the $(n-1)$-dense closed corridors through S that exclude p_i correspond to those antipodal pairs

of $\mathrm{CH}(S_i)$ that are not also antipodal pairs of $\mathrm{CH}(S)$. Thus, each $(n-1)$-dense closed corridor through S can be associated with an antipodal pair of some $\mathrm{CH}(S_i)$, $0 \leq i \leq h-1$.

Given any planar point-set S', for each edge $e \in \mathrm{CH}(S')$ there is a vertex $\alpha(e)$ of $\mathrm{CH}(S')$, such that the line through $\alpha(e)$ and parallel to e does not intersect the interior of $\mathrm{CH}(S')$. Thus, if e' and e'' are clockwise consecutive edges on $\mathrm{CH}(S')$, then the vertices clockwise from $\alpha(e')$ to $\alpha(e'')$ are exactly those forming antipodal pairs with the vertex shared by e' and e''.

Thus each $(n-1)$-dense closed corridor through S can be determined by generating the edge-vertex pairs $(e, \alpha(e))$ of all $\mathrm{CH}(S_i)$, $0 \leq i \leq h-1$. Moreover, for each $\mathrm{CH}(S_i)$, we need only consider edges e that belong to $H(p_i)$. For each such e, $\alpha(e)$ is a vertex of $\mathrm{CH}(S)$.

The preceding discussion suggests a *rotating caliper* algorithm [Tou83, PS88]. Let e^* denote the edge for which $\alpha(e^*)$ is currently being sought. Initializing e^* as the first clockwise edge of $H(p_0)$, we determine $\alpha(e^*)$. Next we let e^* march clockwise along $H(p_0)$ while $\alpha(e^*)$ marches clockwise along L_0. In the process, we find all antipodal pairs of $\mathrm{CH}(S_0)$ and thus determine the widest $(n-1)$-dense closed corridors through S that exclude p_0. This phase terminates when we reach p_1. We then start with the first clockwise edge of $H(p_1)$ and repeat the process for $H(p_1)$; and so on until we complete the processing of $H(p_{h-1})$.

Consider the running time. L_0 and L_1 can be computed in $O(n \log n)$ time. Also, $H(p_i)$ can be found in $O(\log n)$ time, $0 \leq i \leq h-1$. The caliper always moves monotonically clockwise since, as shown earlier, $H(p_i)$ and $H(p_j)$ have no common edges. For each edge e, the cost of determining $\alpha(e)$ can be charged as $O(1)$ per vertex of $\mathrm{CH}(S)$ scanned. Moreover, each such vertex is charged only $O(1)$ times, since, as can be seen from FIGURE 4, for any i the tangent from p_i to L_1 (the one which contains the first clockwise edge of $H(p_{i+1})$) turns clockwise with respect to the tangent t_{i+1} (which contains the last clockwise edge of $H(p_i)$). Thus the rotating caliper stage runs in $O(n)$ time. The space used is clearly $O(n)$.

Theorem 5.1 *A widest $(n-1)$-dense closed corridor through a set S of n points in the plane can be computed in $O(n \log n)$ time and $O(n)$ space.* □

References

[CD90] S. Chattopadhyay and P. Das. The k-dense corridor problems. *Pattern Recognition Letters*, 11:463–469, 1990.

[CGL85] B.M. Chazelle, L.J. Guibas, and D.T. Lee. The power of geometric duality. *BIT*, 25:76–90, 1985.

[EG89] H. Edelsbrunner and L. Guibas. Topologically sweeping an arrangement. *Journal of Computer and System Sciences*, 38:165–194, 1989.

[EGS86] H. Edelsbrunner, L.J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15:317–340, 1986.

[HM88] M. Houle and A. Maciel. Finding the widest empty corridor through a set of points. In G.T. Toussaint, editor, *Snapshots of computational and discrete geometry*, pages 201–213. TR SOCS–88.11, Dept. of Computer Science, McGill University, Montreal, Canada, 1988.

[JP93] R. Janardan and F.P. Preparata. Widest-corridor problems. Technical Report TR–93–17, Dept. of Computer Science, University of Minnesota, 1993. Submitted.

[PS88] F.P. Preparata and M.I. Shamos. *Computational Geometry - An Introduction.* Springer–Verlag, 1988.

[Tou83] G.T. Toussaint. Solving geometric problems with the 'rotating calipers'. In *Proceedings of IEEE MELECON '83*, 1983.

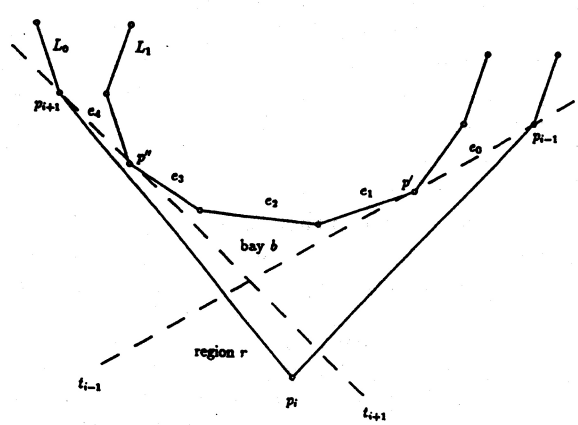

Figure 1: *Chain $H(p_i)$ consists of the edges e_0 through e_4. Here t_{i-1} and t_{i+1} are, respectively, the tangents from p_{i-1} and p_{i+1} to L_1. Bay b is the region bounded by the edges e_1, e_2, e_3 and the upward-facing wedge formed by t_{i-1} and t_{i+1}. Region r is the region bounded by the downward-facing wedge formed by t_{i-1} and t_{i+1}.*