A Linear-Time Algorithm for Computing the Optimal Bridge Connecting Two Disjoint Convex Polygons

Leonidas Palios
Department of Computer Science
University of Ioannina, Ioannina, Greece
palios@cs.uoi.gr

Abstract

Given two disjoint convex polygons P and Q, we are interested in computing the optimal bridge between P and Q, that is, the line segment connecting P and Q such that the maximum of the lengths of the paths from a point of P through the line segment to a point of Q is minimized. This problem has been considered by Cai, Xu, and Zhu, who described an $O(n^2 \log n)$-time algorithm for its solution, where n is the total number of vertices of the two given polygons. Recently, an $O(n)$-time algorithm has been proposed by Kim and Shin, who have also addressed the related problems of bridging a convex and a simple polygon, and two simple polygons.

In this paper, we describe an $O(n)$-time algorithm for the problem of optimally bridging two convex polygons. In contrast to Kim and Shin's algorithm, which relies on the total monotonicity of a table of distances, our algorithm is geometric in nature and relies on the notion of "far-wedges," which we introduce. Additionally, we present a simple linear-time algorithm for computing the intersection points of the boundary of a convex polygon with the farthest-neighbor Voronoi diagram of its vertices without computing the Voronoi diagram; we use this algorithm as a step in our bridging algorithm.

1. Introduction

Given two convex polygons P and Q that do not intersect, we are interested in finding the line segment connecting a point p of P and a point q of Q which minimizes the value of the expression

$$\max_{p' \in P}\{\, d(p', p) \,\} \; + \; d(p, q) \; + \; \max_{q' \in Q}\{\, d(q, q') \,\}, \tag{1}$$

where $d(a, b)$ denotes the Euclidean distance between points a and b. The line segment pq is the *optimal bridge* between P and Q, which guarantees that the maximum length of any path from a point of P through pq to a point of Q is minimized over all possible line segments connecting P and Q.

The problem has been considered by Cai, Xu, and Zhu [3]. They provide motivation for it and they describe an $O(n^2 \log n)$-time algorithm for computing the optimal bridge, where n is the total number of vertices of the two convex polygons P and Q. It is tempting to conjecture that the minimum distance line segment with an endpoint on either polygon is the optimal bridge connecting the two polygons. However, this is not necessarily the case, as indicated in Figure 1: the minimum distance line segment is ad, yet the optimal bridge is pq. Nevertheless, the minimum distance line segment provides a good approximate solution: if it is used to bridge P and Q, then the value of Expression (1) is at most twice its value for the optimal bridge [3]; this observation results in a simple linear-time 2-approximate solution, since the minimum distance line segment connecting two convex polygons can be computed in linear time (see [4] for a linear-time algorithm for computing the minimum distance line segment connecting two convex polyhedra, and [7] for a hierarchical representation of a convex polygon). Recently, Kim and Shin proposed an $O(n)$-time algorithm for this problem [5], which relies on the total monotonicity of a table of distances. Kim and Shin have also addressed the related problems of bridging a convex and a simple polygon, and two simple polygons.

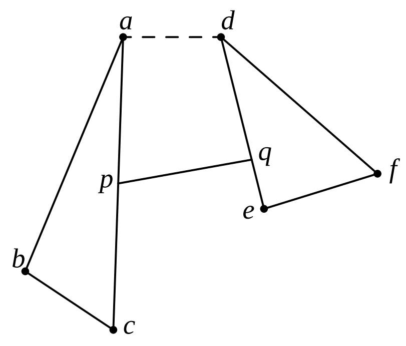


Figure 1

In this paper, we describe a simple linear-time algorithm for computing the optimal bridge between two given disjoint convex polygons P and Q. Unlike Kim and Shin's algorithm, our algorithm is geometric in nature and takes advantage of the properties of the *far-wedges* of a convex polygon, which we introduce. Then, the optimal bridge problem is reduced into a linear number of bridge computation subproblems each of which can be solved in constant time. Moreover, we describe a simple linear-time algorithm to compute the intersection points of the boundary of a convex polygon with the farthest-neighbor Voronoi diagram of its vertices without computing the Voronoi diagram; we use this algorithm as a step in our bridging algorithm.

The paper is structured as follows. In Section 2, we review the terminology that we will be using and present some properties that will be useful for our algorithm, while in Section 3, we describe and analyze the algorithm. Section 4 concludes the paper with a brief summary and extensions.

2. Theoretical Framework

A *polygon* P is the subset of the plane bounded by a number of closed polygonal lines that do not intersect themselves or each other. These polygonal lines form the *boundary* of P, usually denoted by ϑP, which separates the polygon's *interior* from its *complement*. A polygon is *convex* if it is a convex set, that is, for any two points a and b of the polygon the entire line segment ab belongs to the polygon.

Since the two polygons P and Q that we want to bridge are convex and disjoint, they can be separated by a line. Without loss of generality, we may assume that this line is vertical and that P lies on its left and Q on its right.

In order to efficiently compute the optimal bridge connecting P and Q, we need to investigate the possible locations of the endpoints of that bridge. In [3], it has been observed that the optimal bridge has the following property.

Observation 2.1. ([3]) *The optimal bridge connecting two disjoint convex polygons has one endpoint on the boundary of the first polygon, the other endpoint on the boundary of the second polygon, and the two endpoints "see" each other.*

Let us define the *in-hull chain* of the polygon P as the closure of the part of the boundary of P which lies in the interior of the convex hull of $P \cup Q$. If the points of tangency on P of the upper and lower common tangent of P and Q are u and v respectively, then the in-hull chain of P is the part of P's boundary clockwise from u to v (recall that P is to the left of the vertical line separating P and Q); this holds even if $u = v$. Similarly, we define the in-hull chain of Q. In light of these definitions, Observation 2.1 implies that the optimal bridge connects a point of the in-hull chain of P to a point of the in-hull chain of Q and it does not intersect the interior of either polygon. (Below, we show that the endpoints of the optimal bridge in fact belong to a subset of the in-hull chain of either polygon.)

If the endpoint of the optimal bridge on the boundary of the polygon P is p, then the quantity $\max_{p' \in P}\{\, d(p,p') \,\}$ is equal to the distance of p to its *farthest neighbor* in P. It is not difficult to show that the farthest neighbor of any point of a convex polygon is a vertex of that polygon. Finding the all-pairs farthest neighbors of a set of n points in the plane can be done in $O(n \log n)$ time by using the farthest-neighbor Voronoi diagram [9], and this is worst-case optimal. If, however, the points are vertices of a convex polygon and they are given in order along the polygon's boundary, then the all-pairs farthest neighbors of the set of points can be computed in linear time [2]. The algorithm is based on the total monotonicity of the table of pairwise distances of the points, which implies that:

Observation 2.2. *Let S be the sequence of the farthest neighbors of a convex polygon's vertices which we visit in counterclockwise (clockwise, respectively) order along the polygon's boundary. Then, the sequence that results from S after replacing each run of consecutive copies of the same vertex by a single representative is a subsequence of the polygon vertices traversed in counterclockwise (clockwise, respectively) order.*

For example, the farthest neighbors of the vertices 1, 2, …, 8 of the polygon shown in Figure 2 are 5, 6, 6, 1, 1, 1, 2, 5, respectively. If we replace runs of the same vertex by a single representative (including runs that wrap around the sequence), we get 5, 6, 1, 2, which is a subsequence of the polygon vertices traversed in counterclockwise order.

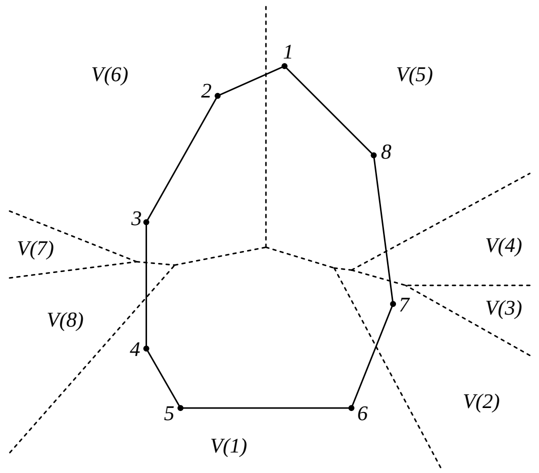


Figure 2

We say that a polygonal line $v_1 v_2 \dots v_k$ is *strictly monotone* with respect to a direction (or a line) ℓ if the intersection of $v_1 v_2 \dots v_k$ with any line perpendicular to ℓ is either the empty set or a single point. For example, the polygonal line which starts at vertex 6,

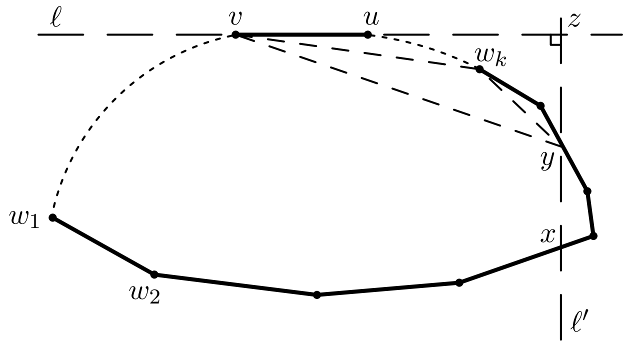


Figure 3

proceeds to vertices 7 and 8, and ends at vertex 1 of the polygon shown in Figure 2 is strictly monotone with respect to the vertical direction. Note that vertices 6 and 1 are the farthest neighbors of vertices 3 and 4 respectively, and that the edge connecting vertices 3 and 4 is vertical. In other words, the polygonal line counterclockwise from vertex 6 to vertex 1 is strictly monotone with respect to the line supporting the edge connecting vertices 3 and 4. This is just an example of a general fact, as shown in the following lemma.

Lemma 2.1. *Let uv be an edge of a convex polygon P, where u precedes v in a counterclockwise traversal of P's vertices along its boundary. Suppose further that the farthest neighbors of u and v in P differ. Then, the part of P's boundary counterclockwise from the farthest neighbor of u to the farthest neighbor of v forms a strictly monotone polygonal line with respect to the line supporting the edge uv.*

Proof: Let the polygonal line be $w_1 w_2 \ldots w_k$, where w_1 and w_k are the farthest neighbors of u and v in P respectively. Suppose for contradiction that the polygonal line is not strictly monotone with respect to the line ℓ supporting the edge uv. Then, there exists a line ℓ' perpendicular to ℓ which intersects $w_1 w_2 \ldots w_k$ in at least two points. Without loss of generality, we may assume that the situation is as shown in Figure 3; symmetric cases are treated in a similar fashion. Let x and y be the points of intersection of ℓ' with the polygonal line, where x is encountered before y in a counterclockwise traversal of P's boundary from w_1 to w_k (note that the intersection may be a line segment, in which case x and y are the endpoints of the line segment in the order mentioned above). Because P is convex and w_k belongs to the counterclockwise part of P's boundary from y to v, w_k lies on or above the line through v and y, on or below the line ℓ, and on or to the left of the line ℓ'; that is, w_k belongs to the triangle with vertices v, y and z, where z is the point of intersection of ℓ and ℓ' (in fact, it belongs to a smaller triangle, the triangle with vertices u, y and z). Then, $d(v, w_k) \leq d(v, y)$: the inequality clearly holds if w_k belongs to the line segment vy; if $w_k \notin vy$, the inequality follows from the fact that the angle $\widehat{v w_k y}$ is at least equal to $\widehat{vzy} = \pi/2$, and thus it is greater than the angle $\widehat{v y w_k}$. Moreover, $d(v, y) < d(v, x)$. The two inequalities yield $d(v, w_k) < d(v, x)$, in contradiction to the fact that w_k is the farthest neighbor of v in P. ∎

Of course, the optimal bridge may not necessarily be attached to a vertex of P or a vertex of Q; therefore, we need to determine the farthest neighbor of every boundary point of P and similarly for Q. We can do that by computing the partition of the boundary of each of these polygons into sets of points with the same farthest neighbor. Then, Observation 2.2 leads to the following corollary.

Corollary 2.1. *Suppose that the boundary of a convex polygon P has been partitioned into sets of points with the same farthest neighbor in P. Then, if we consider these sets in counterclockwise (clockwise, respectively) order along P's boundary, their farthest neighbors form a subsequence of P's vertices traversed in counterclockwise (clockwise, respectively) order.*

The statement of this corollary implies that these sets are connected. In fact, each such set is the intersection of the boundary of the polygon with a region of the farthest-neighbor Voronoi diagram of the polygon's vertices; see Figure 2, where the dotted line segments form the farthest-neighbor Voronoi diagram of the polygon's vertices.

The partition of the boundaries of P and Q into sets with the same farthest neighbor (in P and Q respectively) helps us simplify the computation of the optimal bridge connecting P and Q: we consider pairs consisting of a member from either partition; for such a pair, we compute the corresponding optimal bridge with endpoints on the boundary sets of the pair, which involves considering a small number of cases (see Section 3.2 below) and can be done in constant time. By considering all necessary pairs, we guarantee that the optimal bridge of P and Q is that among the computed bridges that minimizes the value of Expression (1).

In order to formalize our approach, we introduce the notion of the far-wedges of a convex polygon. We consider the partition of the boundary of a polygon P into sets with the same farthest neighbor in P; let the partition sets in counterclockwise order along the boundary of P (starting at an arbitrary set) be $s_1 ... s_2$, $s_2 ... s_3$, …, $s_k ... s_1$. Moreover, let the corresponding farthest neighbors be v_{j_1}, v_{j_2}, …, v_{j_k}; these are vertices of P and they form a subsequence of the sequence of P's vertices in counterclockwise order along P's boundary starting at v_{j_1}. Each boundary set $s_i ... s_{i+1}$ is associated with the vertex v_{j_i}; this pair defines two lines, one determined by the points v_{j_i} and s_i, the other determined by v_{j_i} and s_{i+1}. These two lines define four wedges, one of which contains in its closure the boundary set $s_i ... s_{i+1}$; this is the *far-wedge* with *apex* v_{j_i} and *boundary chain* $s_i ... s_{i+1}$. Figure 4 depicts the far-wedges of the convex polygon of Figure 2; note that there is no far-wedge associated with the vertices 3 and 7.

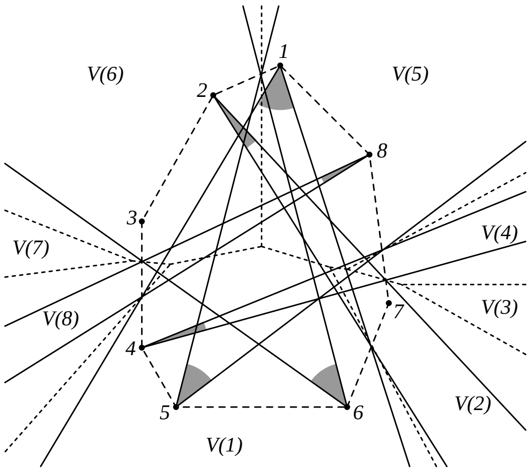


Figure 4

The far-wedges possess some interesting properties as described in the following lemmata.

Lemma 2.2. *The far-wedge of a vertex v of a convex polygon P lies in the wedge that is defined by the lines supporting v's incident edges and contains P.*

Proof: Easily seen to be true, since each of the two halflines bounding the far-wedge of v is defined by v and a point of the polygon P. ∎

Lemma 2.3. *No two far-wedges of a convex polygon P intersect in the complement of P.*

Proof: The proof is by contradiction. Let us consider two far-wedges, one with apex u and boundary chain $s...t$, and the other with apex v and boundary chain $x...y$, and let us suppose that these two wedges have a point r in common which lies outside P. Since a far-wedge is convex, the line segment ru lies in the far-wedge with apex u. Moreover, ru intersects the boundary chain $s...t$ at a point, say, p. Since p belongs to $s...t$, p's farthest neighbor among the points in P is u; therefore, $d(p,u) \geq d(p,v)$. Similarly, if q is the point of intersection of rv with the boundary chain $x...y$, q's farthest neighbor is v and $d(q,v) \geq d(q,u)$. If we sum these inequalities and add the term $d(r,p) + d(r,q)$ on both sides, we get

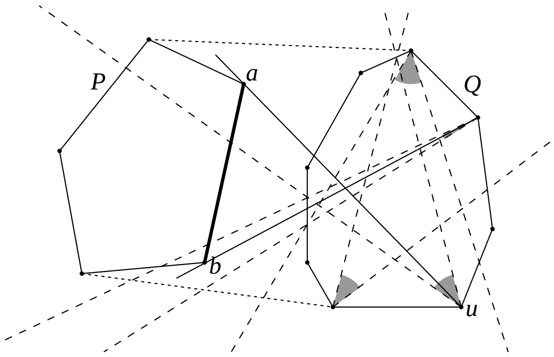
$$\begin{aligned} d(p,u) + d(q,v) + d(r,p) + d(r,q) &\geq d(p,v) + d(q,u) + d(r,p) + d(r,q) \\ \iff \qquad d(r,u) + d(r,v) &\geq d(r,q) + d(q,u) + d(r,p) + d(p,v), \end{aligned}$$

which, because of the triangle inequality, holds only if q belongs to the line segment ru and p belongs to the line segment rv. This implies that u, v, p, q, and r are all collinear. But then in light of Lemma 2.2, the intersection of the far-wedges of u and v and the line through u and v is a subset of the line segment uv; since r belongs to this intersection, r belongs to the line segment uv, which comes into contradiction with the fact that r lies outside P. ∎

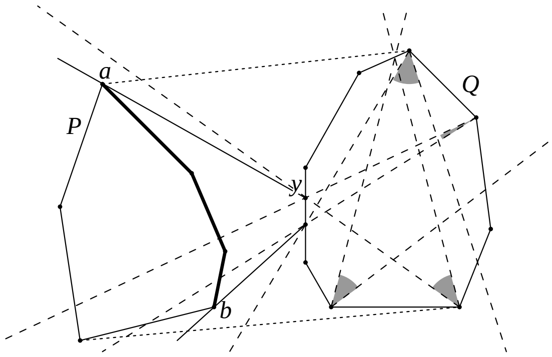
In fact, it can be shown that the intersection of the far-wedge of a vertex v of the convex polygon P with the complement of P lies entirely in the Voronoi region of v in the farthest-neighbor Voronoi diagram of the vertices of P; this directly implies Lemma 2.3. Moreover, it is worth noting that any two far-wedges whose boundary chains share an endpoint are separated in the complement of P by a wedge with apex the common endpoint (see Figure 4).

As mentioned above, our method for computing the optimal bridge between the convex polygons P and Q relies on the computation of the optimal bridges for all necessary pairs of a far-wedge of P and a far-wedge of Q. Since the endpoints of the optimal bridge "see" each other and thus belong to the in-hull chains of P and Q, we need only consider the far-wedges of the polygons P and Q whose boundary chains intersect the in-hull chain of P and of Q respectively; these are the *useful* far-wedges of P and Q respectively. (In Figures 5(a) and 5(b), only the useful far-wedges of the polygon Q are shown.) The useful far-wedges of P can be ordered in clockwise order of their boundary chains from the upper tangent point on P to the lower tangent point along P's boundary; note that the first and last useful far-wedges may need to be clipped about the upper and lower tangent points respectively (it is possible that the first and last entries may be contributed by the same far-wedge, in which case the clipping will produce two smaller wedges whose interiors are disjoint). In a similar fashion, we order the useful far-wedges of Q in counterclockwise order of their boundary chains from the upper tangent point on Q to the lower tangent point.

We noted earlier that the optimal bridge may be attached to any point of the in-hull chain of P. In fact, we show that it is attached to a point in a subset of this chain, which we will call the *reduced chain* of P. To define the reduced chain of P, we consider all the useful far-wedges of the polygon Q. The polygon P is intersected by some of them and by some of the wedges between consecutive far-wedges of Q, and is thus partitioned into slices; see Figure 5. (Note that P may entirely belong to a far-wedge or to a wedge between consecutive far-wedges, so that it may consist of a single slice.) If the topmost slice belongs to a far-wedge with apex u, then we draw the upper tangent halfline from u to P;



(a)



(b)

Figure 5

see Figure 5(a). If it lies between far-wedges, then it lies in a wedge with apex a point, say, y, on the boundary of Q; then, we draw the upper tangent halfline from y to P (see Figure 5(b)). Let a be the point at which the tangent touches P. We work in a similar fashion with the lowermost slice defining a lower tangent to P from a vertex or a point on the boundary of Q; let b be the point of tangency at P. Then, the *reduced chain* of P is the part of P's boundary clockwise from a to b (in Figures 5(a) and 5(b), the reduced chain of P is indicated by the thick polygonal line). Note that, since the tangents which define the endpoints of the reduced chain are drawn from points of Q, the reduced chain is a subset of the in-hull chain of P. The construction implies that:

Observation 2.3. *The tangent halflines defining the endpoints of the reduced chain of the convex polygon P do not intersect in the complement of Q any of the halflines bounding the far-wedges of Q.*

We show next that the optimal bridge is attached to a point of the reduced chain of P; thus, we will focus on the far-wedges of P whose boundary chains intersect the reduced chain of P. It is important to observe that since the polygons P and Q are separated by a vertical line, any line intersecting both P and Q is not vertical and therefore the above/below relation with respect to that line is well defined.

Lemma 2.4. *The optimal bridge connecting the convex polygons P and Q is attached to P at a point of P's reduced chain.*

Proof: Suppose, for contradiction, that the endpoint of the optimal bridge on the boundary of P does not belong to the reduced chain of P. Then, the endpoint belongs to the difference of the in-hull chain of P minus the reduced chain of P. In general, this difference consists of a subset of P's boundary counterclockwise from the top endpoint of the reduced chain and another subset of P's boundary clockwise from the bottom endpoint of the reduced chain (see Figure 6). Below, we will concentrate in the former case; the latter is handled similarly.

Let pq be the optimal bridge, whose endpoint p belongs to the part of P's in-hull chain which is counterclockwise from the top endpoint a of the reduced chain. We consider two cases depending on whether the top slice of P belongs to a far-wedge of Q or not. Suppose that the top slice belongs to a far-wedge U of Q, and let u be U's apex (Figure 6(a)). Then, since the line ua is an upper tangent from u to P, the point p lies on or below the line ua.

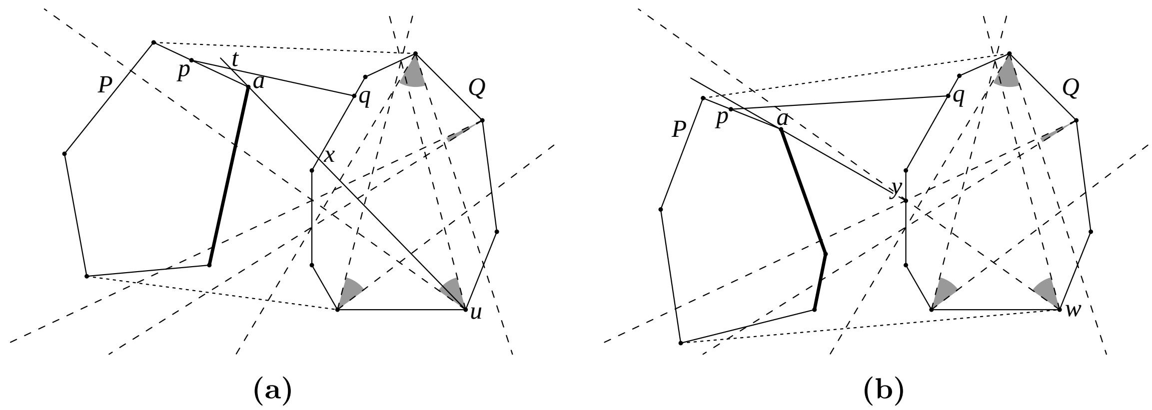


Figure 6

On the other hand, q lies above the line ua, since it "sees" the point p which is farther away from a in the counterclockwise direction along P's boundary. Thus, the line ua intersects the bridge pq; let t be the point of intersection. Additionally, if the point of intersection of the line ua with the boundary chain of U is x, then x belongs to the line segment ut; t cannot fall between u and x, because it would then belong to Q, in contradiction to the fact that the interior of the line segment pq does not intersect Q. Therefore, the point x belongs to the triangle with vertices u, p, and q, and does not coincide with q. This implies that $d(p,x) + d(x,u) < d(p,q) + d(q,u) \leq d(p,q) + d(q, FN(q))$, where $FN(q)$ denotes the farthest neighbor of q in Q. Since the farthest neighbor of x in Q is u, this inequality shows that the bridge px results in a smaller value of Expression (1), thus contradicting the fact that pq is the optimal bridge connecting P and Q. In the second case, the top slice belongs to a wedge between two consecutive far-wedges of Q (Figure 6(b)). Let y be the apex of this wedge (the common endpoint of the boundary chains of the two far-wedges), and let w be the apex of the far-wedge whose boundary chain is incident upon y in the clockwise direction along Q's boundary. Then, in a fashion similar to the previous case, we can show that y belongs to the triangle with vertices w, p, and q, and does not coincide with q. Consequently, $d(p,y)+d(y,w) < d(p,q)+d(q,w) \leq d(p,q)+d(q,FN(q))$, which contradicts the optimality of the bridge pq in light of the bridge py. ∎

An advantage of the reduced chain over the in-hull chain is the property given in the following lemma; we will exploit this property in our algorithm.

Lemma 2.5. *Any halfline bounding a far-wedge of the convex polygon Q intersects the reduced chain of the convex polygon P in at most one point.*

Proof: Suppose for contradiction that this is not true and that there is a halfline h bounding a far-wedge of Q that has two points of intersection with the reduced chain of P. Let a and b be the endpoints of the reduced chain. Then, since the region bounded by the reduced chain of P and the line segment ab is convex, the halfline h does not intersect the interior of the line segment ab. Let us consider now the quadrilateral bounded by the line segment ab, the upper and lower tangents defining the reduced chain, and the vertical line ℓ separating P and Q. Clearly, the reduced chain belongs to this quadrilateral. Moreover, since the halfline h has its apex to the right of ℓ, extends to the infinity to the left and intersects the reduced chain, it cuts through the quadrilateral. Since it does not intersect the interior of the line segment ab, it intersects the upper or the lower tangent or both. But this is impossible, because of Observation 2.3. ∎

Let F be a far-wedge of the polygon P, and let $\mathcal{S} = [W_1^Q, \ldots, W_k^Q]$ be the ordered set of the useful far-wedges of the polygon Q from the upper tangent point counterclockwise to the lower tangent point. We partition the set $\mathcal{S}$ into three subsets: $\mathcal{A} = [W_1^Q, \ldots, W_{i-1}^Q]$, $\mathcal{B} = [W_i^Q, \ldots, W_j^Q]$, and $\mathcal{C} = [W_{j+1}^Q, \ldots, W_k^Q]$, where $\mathcal{B}$ is the set of far-wedges of Q which intersect the interior of the boundary chain of F. Note that any (but not all) of the above three subsets may be empty. We show that a bridge with one endpoint on the boundary chain of F and the other on the boundary chain of any of the far-wedges in $\mathcal{A}$ other than W_{i-1}^Q cannot be optimal.

Lemma 2.6. *Let F be a far-wedge of the polygon P. Suppose that we partition the ordered set of the useful far-wedges of Q into the sets $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ with respect to F as indicated above. Then, a bridge with one endpoint on the boundary chain of F and the other on the boundary chain of any of the far-wedges in $\mathcal{A}$ other than W_{i-1}^Q cannot be optimal.*

Proof: Suppose for contradiction that the optimal bridge connects a point p of the boundary chain of F to a point q of the boundary chain of a far-wedge W in $\mathcal{A}$ other than W_{i-1}^Q. Let u be the apex and $s...t$ be the boundary chain (traversed in counterclockwise order along the boundary of Q) of W_{i-1}^Q. Since pq is the optimal bridge and the far-wedge W precedes W_{i-1}^Q in $\mathcal{A}$, we conclude that the point t lies below the line supporting pq. Moreover, the points p and q lie on opposite sides of the line through u and t. Therefore, t belongs to the interior of the triangle with vertices u, p, and q. This implies that

$$d(p,t) + d(t,u) < d(p,q) + d(q,u) \leq d(p,q) + d(q, FN(q)),$$

where $FN(q)$ denotes the farthest neighbor of q in Q. The above inequality, however, contradicts the optimality of the bridge pq in light of the bridge pt. ∎

A similar lemma holds for the far-wedges in $\mathcal{C}$: a bridge with one endpoint on the boundary chain of the far-wedge F and the other on the boundary chain of any of the far-wedges in $\mathcal{C}$ except for W_{j+1}^Q cannot be optimal. These lemmata imply the following corollary.

Corollary 2.2. *Let P and Q be two disjoint convex polygons and let pq be the optimal bridge connecting them, where $p \in P$ and $q \in Q$. Moreover, let F be a far-wedge of P whose boundary chain intersects the reduced chain of P, and let $\mathcal{A} = [W_1^Q, \ldots, W_{i-1}^Q]$, $\mathcal{B} = [W_i^Q, \ldots, W_j^Q]$, and $\mathcal{C} = [W_{j+1}^Q, \ldots, W_k^Q]$ be the partition of the set of useful far-wedges of Q with respect to F as described above. If the optimal bridge pq has its endpoint p on the boundary chain of F, then it has to be one of the (partial) optimal bridges connecting F to one of the far-wedges W_{i-1}^Q, W_i^Q, …, W_j^Q, W_{j+1}^Q.*

Lemma 2.3 and Corollary 2.2 help us prove that the number of pairs of a far-wedge of P and a far-wedge of Q that we need to consider in order to compute the optimal bridge is linear in the total number of vertices of the polygons P and Q.

3. The Algorithm

As mentioned earlier, we assume that the two given polygons P and Q are separated by a vertical line with P being on its left and Q on its right. The algorithm for computing the optimal bridge connecting P and Q is outlined in Algorithm 3.1. Note that Step 5 of the algorithm implements Corollary 2.2; while processing the far-wedge F of P, the pairs (F, W)

1. We compute the far-wedges of the two polygons P and Q.
2. We form a list L_P of the useful far-wedges of P ordered in clockwise order; similarly, we form a list L_Q of the useful far-wedges of Q ordered in counterclockwise order.
3. Starting at P's vertex with the largest x-coordinate and by moving up and down, we determine the reduced chain of P.
4. Let L'_P be the sublist of the far-wedges in L_P whose boundary chains intersect the reduced chain of P. Let L'_Q be the sublist of the far-wedges in L_Q which intersect the reduced chain of P; we clip the first and last element of L'_Q about the endpoints of the reduced chain of P and we augment L'_Q by including one far-wedge before and one after (if there exist).
5. $U \leftarrow$ first far-wedge in L'_Q;
   `for` each far-wedge F in L'_P in order
       `while` $U \neq$ one of the last 2 elements of L'_Q
           $T \leftarrow$ element following U in L'_Q;
           `if` F's boundary chain is below T
               $U \leftarrow T$;
           `else`
               we exit the `while` loop;
       $W \leftarrow U$;
       we compute optimal bridge of (F, W) and corresponding maximum path length ℓ;
       `if` $\ell <$ minimum value min_ℓ of maximum path length computed so far
           we update the overall optimal bridge and min_ℓ;
       `if` $W \neq$ last element of L'_Q
           $W \leftarrow$ element following W in L'_Q;
           `while` $W \neq$ last element of L'_Q `&&` F's boundary chain does not lie above W
               we compute opt. bridge of (F, W) and corresponding max path length ℓ;
               `if` $\ell <$ minimum value min_ℓ of maximum path length computed so far
                   we update the overall optimal bridge and min_ℓ;
               $W \leftarrow$ element following W in L'_Q;
           we compute optimal bridge of (F, W) and corresponding max path length ℓ;
           `if` $\ell <$ minimum value min_ℓ of maximum path length computed so far
               we update the overall optimal bridge and min_ℓ;

Algorithm 3.1. The algorithm for computing the optimal bridge connecting two disjoint convex polygons P and Q.

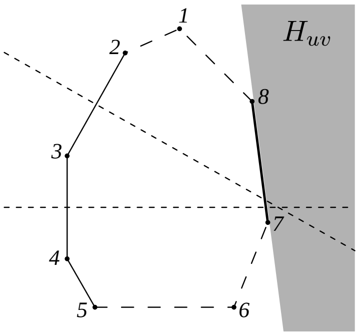
which are considered for optimal bridge computation are precisely the pairs (F, W^Q_{i-1}), (F, W^Q_i), …, (F, W^Q_{j+1}). It is also worth noting that the above/below tests with respect to far-wedges of Q involve elements of L'_Q except for the first and the last one; since both halflines bounding these far-wedges intersect the reduced chain of P (see Step 4), these tests are well defined.

Next, we describe in detail Step 1 of the algorithm and how one can compute the optimal bridge connecting a point of the boundary chain of a far-wedge of P to a point of the boundary chain of a far-wedge of Q.

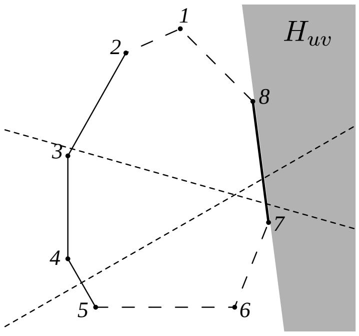
3.1. Computing the far-wedges of a convex polygon. All the far-wedges can be determined easily by intersecting the boundary of the given convex polygon with the farthest-neighbor Voronoi diagram of the vertices of the polygon; if the intersection of the Voronoi region of a vertex v with the boundary of the polygon is a polygonal line $s...t$, then there is a far-wedge with apex v and boundary chain $s...t$. Although the computation of the farthest Voronoi diagram for a general set of n points in the plane may necessitate $\Theta(n \log n)$ time in the worst case, the diagram for the vertices of a convex polygon can be computed in time linear in the size of the polygon [1]. This suggests a linear-time algorithm to compute the far-wedges, but it subsumes the computation of the Voronoi diagram as a preprocessing step. Below, we propose a simple alternative way to compute the far-wedges of a convex polygon in linear time without computing the farthest-neighbor Voronoi diagram of its vertices. We work as follows:

1. We compute the all-pairs farthest neighbors of all the vertices of the given convex polygon; this can be done in time linear in the size of the polygon by using the algorithm of Aggarwal et al [2].

2. For each edge uv of the polygon, we check the farthest neighbors of its endpoints u and v. If the vertices u and v have the same farthest neighbor, then the farthest neighbor of every point of the edge is the same as the (common) farthest neighbor of u and v. Otherwise, we refine the edge by partitioning it into subsegments, each having a different farthest neighbor. This is done as follows. If the farthest neighbors of u and v are adjacent vertices of the polygon, then the edge uv is partitioned into two subsegments about its point of intersection with the perpendicular bisector of the line segment connecting the farthest neighbors of u and v; clearly, this bisector intersects the edge uv, because u and u's farthest neighbor lie on opposite sides of the bisector and similarly for v and v's farthest neighbor. Suppose now that the farthest neighbors of u and v differ and they are not adjacent vertices of the polygon. We may assume without loss of generality that u is immediately before v in a counterclockwise traversal of the vertices along the boundary of the polygon. Note that Observation 2.2 implies that the farthest neighbors of the subsegments of the edge uv form a subsequence of the sequence of vertices from the farthest neighbor of u to the farthest neighbor of v traversed in counterclockwise order along the polygon's boundary.
   In order to compute the partition into subsegments and the associated farthest neighbors, we use a stack to store the current list of farthest neighbors. Initially, we push in the stack the farthest neighbor of u and the polygon vertex adjacent to it in counterclockwise order along the polygon boundary. Then, we process each of the remaining vertices up to the farthest neighbor of v in the following way. We consider the line segment defined by the current vertex and the vertex at the top of the stack and the line segment defined by the two vertices at the top two positions of the stack. If the perpendicular bisectors of these two segments intersect at a point belonging to H_{uv} (where H_{uv} is the closed halfplane defined by the line supporting the edge uv and not intersecting the interior of P), then the vertex at the top of the stack is popped, and the perpendicular bisector test is repeated for the line segment defined by the current vertex and the vertex which is now at the top of the stack and the line segment defined by the two vertices at the top two positions of the stack. The process is repeated until either the stack ends up containing only one element or the bisectors intersect at a



(a)



(b)

Figure 7

point in the complement of H_{uv}. Then, the current vertex is pushed in the stack; its processing is complete and we continue with the processing of the next vertex.
After all the vertices from the farthest neighbor of u counterclockwise to the farthest neighbor of v have been processed, the stack contains the farthest neighbors of the subsegments of the edge uv in order from u to v. The subsegments are determined by the points of intersection of the edge uv and the perpendicular bisectors of the line segments defined by pairs of vertices in consecutive positions of the stack.

3. The final partition of the boundary of the polygon is computed by merging adjacent edges and edge subsegments which are associated with the same farthest neighbor (the merging can be easily performed during the previous step by processing the edges in the order they appear along the boundary of the polygon). The far-wedges are determined by each element of the partition and the associated farthest neighbor.

To illustrate the algorithm let us consider the processing of the edge connecting vertices 7 and 8 of the polygon shown in Figure 2. The farthest neighbors of vertices 7 and 8 are the vertices 2 and 5 respectively. So, we start by pushing vertices 2 and 3 in the stack. Then, we consider the perpendicular bisectors of the line segments with endpoints the vertices 4 and 3, and 3 and 2 (Figure 7(a)); these bisectors intersect at a point of H_{uv}, and thus vertex 3 is popped from the stack. At this point, the processing of vertex 4 is complete, vertex 4 is pushed in the stack, and we continue with the processing of vertex 5. Now, we consider the perpendicular bisectors of the line segments with endpoints the vertices 5 and 4, and 4 and 2 (Figure 7(b)), which intersect at a point which does not belong to H_{uv}. The processing of vertex 5 is complete and vertex 5 is pushed in the stack. This completes the processing of the edge connecting vertices 7 and 8; the contents of the stack indicate that this edge contributes three subsegments with farthest neighbors the vertices 2, 4, and 5 (compare with Figure 2).

The correctness of this algorithm is established by means of the following two lemmata.

Lemma 3.1. *Let P be a convex polygon and let uv be one of its edges with u preceding v in a counterclockwise traversal of P's vertices along its boundary. Then, any vertex of P, which lies between the farthest neighbor of u and the farthest neighbor of v in a counterclockwise traversal of P's vertices and is not in the stack after the edge uv has been processed in Step 2 of the algorithm, is not the farthest neighbor of any point of the edge uv in P.*

Proof: Since each of the vertices, which lie between the farthest neighbor of u and the farthest neighbor of v in a counterclockwise traversal of P's vertices, is pushed in the stack when processed, a vertex that is not in the final stack must have been popped. Let w be such a vertex. Since w has been popped from the stack, then it must be the case that the perpendicular bisectors defined by two line segments connecting w to two other vertices of P intersect at a point in H_{uv}. The two bisectors define four wedges, one of which contains w; then, the wedge across from this wedge contains in its closure the farthest-neighbor Voronoi region of w. Since the bisectors intersect at a point in H_{uv}, the interior of the farthest-neighbor Voronoi region of w lies entirely outside P; therefore, the Voronoi region does not contribute any subsegments on the edge vu, and w has been correctly removed from the stack. ∎

Lemma 3.2. *Let P be a convex polygon and let uv be one of its edges. Then, for every vertex w which is in the stack after the edge uv has been processed in Step 2 of the algorithm, there exists a point p of the edge uv such that w is the farthest neighbor of p in P.*

Proof: Suppose without loss of generality, that u precedes v in a counterclockwise traversal of P's vertices along its boundary. Let the contents of the stack be $w_1 = FN(u)$, w_2, $\ldots$, $w_k = FN(v)$ from bottom to top, where $FN(u)$ and $FN(v)$ are the farthest neighbors of u and v in P respectively; then, the vertices w_1, w_2, $\ldots$, w_k form a subsequence of the sequence of P's vertices in a counterclockwise traversal of P's boundary from w_1 to w_k. Because of the convexity of P and of Lemma 2.1, this implies that the polygonal line $w_1 w_2 \ldots w_k$ is convex and strictly monotone with respect to the line supporting the edge uv.

Let h_u be the halfline that is produced if we extend the line segment uv to infinity past its endpoint v. Then, the perpendicular bisector of the line segment $w_1 w_2$ intersects h_u; otherwise, u and $FN(u) = w_1$ would be on the same side with respect to the bisector, and thus $d(u, FN(u)) < d(u, w_2)$, in contradiction to the fact that $FN(u)$ is the farthest neighbor of u. Since the perpendicular bisectors of the line segments $w_1 w_2$ and $w_2 w_3$ intersect in the complement of H_{uv}, the perpendicular bisector of $w_2 w_3$ intersects h_u and the point of intersection is farther from u than the point of intersection of the bisector of $w_1 w_2$ with h_u. By applying this argument over and over for the pairs of line segments defined by triples of consecutive elements of the stack, we conclude that the perpendicular bisectors of all the line segments defined by pairs of consecutive elements of the stack intersect h_u. In a similar fashion, one can show that the perpendicular bisectors of all the line segments defined by pairs of consecutive elements of the stack intersect the halfline h_v which is produced if we extend the line segment uv past u. In other words, all these bisectors intersect the edge uv. Moreover, the points of intersection are in the same order from u to v as the pairs of consecutive elements of the stack (from bottom to top) that define them. So, if the perpendicular bisector of the line segment $w_i w_{i+1}$ intersects the edge uv at the point p_i $(1 \leq i < k)$, and if $p_0 = u$ and $p_k = v$, then the line segments $p_i p_{i+1}$ $(0 \leq i < k)$ define a partition of the edge uv. We will show that every point in the open line segment $p_j p_{j+1}$ has w_{j+1} as its farthest neighbor in the polygon. Let x be a point in the open line segment $p_j p_{j+1}$. Then, the perpendicular bisectors of all the line segments defined by pairs of consecutive elements of $[w_1, w_2, \ldots, w_j]$ intersect the line segment ux. This implies that

$$d(x, w_j) \; > \; d(x, w_{j-1}) \; > \; \ldots \; > \; d(x, w_1).$$

Moreover, the perpendicular bisectors of all the line segments defined by pairs of consecutive elements of $[w_j, w_{j+1}, \ldots, w_k]$ intersect the line segment xv, which implies that

$$d(x, w_j) \;>\; d(x, w_{j+1}) \;>\; \ldots \;>\; d(x, w_k).$$

Observation 2.2 implies that the farthest neighbor of x has to be between the farthest neighbor of u and the farthest neighbor of v in a counterclockwise traversal of P's vertices; hence, in light of Lemma 3.1, it has to be one of the vertices in the stack. The above inequalities imply that this vertex is precisely w_j. Therefore, for every vertex in the stack, there exists a point of the edge uv with that vertex as its farthest neighbor among the points of the convex polygon P. ∎

3.2. Computing the optimal bridge between two far-wedges. Suppose that we have a far-wedge F (with apex u and boundary chain $s...t$) of the polygon P and a far-wedge F' (with apex v and boundary chain $x...y$) of the polygon Q; the boundary chain $s...t$ is assumed to be traversed clockwise along the boundary of P, whereas the chain $x...y$ counterclockwise along the boundary of Q. One needs to distinguish the following four main cases; any other case is top-to-bottom or left-to-right symmetric to one of these cases.

1. *the points u and v belong to both far-wedges* (Figure 8(a)): Then, so does the line segment connecting u and v. The optimal bridge connecting F and F' is the closure of the intersection of the line segment uv with the complement of $P \cup Q$. In fact, this is the optimal bridge of P and Q, and we can thus stop the processing and report it.

2. *the point u belongs to the far-wedge F' and the far-wedge F lies above the line through u and v:* Then, if t belongs to F', the optimal bridge connecting F and F' is the closure of the intersection of the line segment tv with the complement of $P \cup Q$, and the value of Expression (1) is the sum of the lengths of the line segments ut and tv (Figure 8(b)). If t does not belong to F', then it lies above F'; then, the optimal bridge connecting F and F' is the line segment tx and the value of Expression (1) is the sum of the lengths of the line segments ut, tx, and xv (Figure 8(c)).

3. *both far-wedges F and F' lie above the line through u and v*: This case is similar to the previous one. If t belongs to F', the optimal bridge connecting F and F' is the closure of the intersection of the line segment tv with the complement of $P \cup Q$, and the value of Expression (1) is the sum of the lengths of the line segments ut and tv

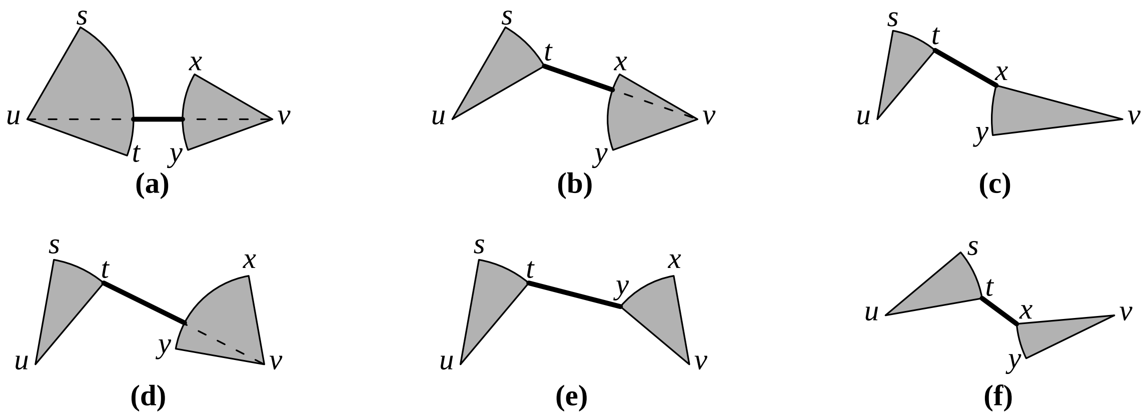


Figure 8

(Figure 8(d)). If t does not belong to F', then it lies below F'; then, the optimal bridge connecting F and F' is the line segment ty and the value of Expression (1) is the sum of the lengths of the line segments ut, ty, and yv (Figure 8(e)).

4. *the far-wedge F lies above the line through u and v whereas the far-wedge F' lies below it* (Figure 8(f)): Then, the optimal bridge connecting F and F' is the line segment tx and the value of Expression (1) is the sum of the lengths of the line segments ut, tx, and xv.

3.3. Time complexity of the algorithm. As shown in Section 3.1, Step 1 of the Algorithm 3.1 takes time linear in the total size of the polygons P and Q. Step 2 also takes linear time: note that computing the common tangents of two disjoint convex polygons can be done in time linear in their combined size [8]. It is not difficult to see that Steps 3 and 4 are completed in linear time as well. Finally, so is Step 5. To see this, we first observe that U does not move backwards in L'_Q; thus, the total number of times the condition of the first `while` loop in Step 5 does not exceed $|L'_P| + |L'_Q|$, which is linear in the combined size of P and Q. Moreover, the total number of (partial) optimal bridge computations for pairs of a far-wedge of P and a far-wedge of Q is also linear in the combined size of P and Q: if the partition of the set of useful far-wedges of Q with respect to a far-wedge F of P is $\mathcal{A} = [W_1^Q, \ldots, W_{i-1}^Q]$, $\mathcal{B} = [W_i^Q, \ldots, W_j^Q]$, and $\mathcal{C} = [W_{j+1}^Q, \ldots, W_k^Q]$, then $|\mathcal{B}| \le k_F + 2$, where k_F denotes the number of far-wedges of Q whose intersection with the reduced chain of P is a subset of the boundary chain of F. Then, since our algorithm computes (partial) optimal bridges for the pairs (F, W_{i-1}^Q), (F, W_i^Q), ..., (F, W_{j+1}^Q) during the processing of F, it considers at most $k_F + 4$ such pairs. Because of Lemmata 2.3 and 2.5, we have that $\sum_F k_F \le |Q|$. Therefore, $\sum_F (k_F + 4) = O(|P| + |Q|)$, that is, the total number of (partial) optimal bridge computations is linear in the total size of P and Q.

4. Concluding Remarks – Extensions

We presented an algorithm for computing the optimal bridge between two disjoint convex polygons, which takes time linear in the total size of the given polygons, and is therefore optimal. The algorithm is fairly simple and can be easily coded for use in practical applications. We also presented a simple linear-time algorithm for computing the intersection points of the boundary of a convex polygon with the farthest-neighbor Voronoi diagram of its vertices without computing the Voronoi diagram; we use this algorithm as a step in our bridging algorithm.

The problem can be generalized in several ways. First, the convexity restriction on the one or both polygons can be relaxed: compute the optimal bridge between a convex and a non-convex polygon, or between two disjoint non-convex polygons. Kim and Shin provided an $O((n+m)\log(n+m))$-time algorithm for the computation of the optimal bridge between a non-convex simple polygon of n vertices and a convex polygon of m vertices which are disjoint, and an $O(n\,m + m\log m)$-time algorithm to bridge two disjoint non-convex polygons of n and m vertices where $m \ge n$. It is interesting to investigate whether faster algorithms for these problems are possible, and if yes, to come up with such algorithms. Additionally, the problem can be extended into three-dimensional space, where one wishes to compute the optimal bridge connecting two disjoint convex polyhedra. Currently, no algorithms are known for this problem, nor for the problems arising when the convexity restriction on one or both polyhedra is relaxed.

5. References

1. A. Aggarwal, L.J. Guibas, J. Saxe, and P.W. Shor, "A linear-time algorithm for computing the Voronoi diagram of a convex polygon," *Discrete and Computational Geometry* 4(6) (1989), 591–604.

2. A. Aggarwal, M.M. Klawe, S. Moran, P. Shor, and R. Wilber, "Geometric Applications of a Matrix-Searching Algorithm," *Algorithmica* 2(2) (1987), 195–208.

3. L. Cai, Y. Xu, and B. Zhu, "Computing the optimal bridge between two convex polygons," *Information Processing Letters* 69 (1999), 127–130.

4. D.P. Dobkin and D.G. Kirkpatrick, "A linear algorithm for determining the separation of convex polyhedra," *Journal of Algorithms* 6 (1985), 381–392.

5. S.K. Kim and C.S. Shin, "Computing the Optimal Bridge between two Polygons," *Research Report HKUST-TCSC-1999-14*, Hong-Kong University, 1999.

6. D.T. Lee and F.P. Preparata, "The all-nearest neighbor problem for convex polygons," *Information Processing Letters* 7(4) (1978), 189-192.

7. K. Mehlhorn, *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry,* Springer-Verlag, 1984.

8. F.P. Preparata and S.J. Hong, "Convex Hulls in Two and Three Dimensions," *Communications of the ACM* 20 (1977), 87–93.

9. F.P. Preparata and M.I. Shamos, *Computational Geometry, An Introduction,* Springer-Verlag, 1985.