December 1992

UILU-ENG-92-2248
ACT 125

Applied Computation Theory

AN OPTIMAL ALGORITHM FOR DETERMINING THE SEPARATION OF TWO NONINTERSECTING SIMPLE POLYGONS

Nancy Amato

Coordinated Science Laboratory
College of Engineering
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-92-2248 (ACT#125)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) 1308 W. Main St. Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) Arlington, VA 22217
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Joint Services Electronics Program 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-90-J-1270
8c. ADDRESS (City, State, and ZIP Code) Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
An Optimal Algorithm for Determining the Separation of Two Nonintersecting Simple Polygons

12. PERSONAL AUTHOR(S)
Amato, Nancy

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	December 1992	16

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	separation, simple polygons, sequential, parallel

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Given nonintersecting simple polygons P and Q, their *separation*, denoted by $\sigma(P, Q)$, is defined to be the minimum distance between their boundaries; a pair of points $p \in P$ and $q \in Q$ realize $\sigma(P, Q)$, if $d(p, q) = \sigma(P, Q)$. We present an optimal $\Theta(N)$ time algorithm for determining the separation of two disjoint simple polygons P and Q, and finding a pair of points (p, q), $p \in P$ and $q \in Q$, realizing $\sigma(P, Q)$, where $N = |P| + |Q|$. The best previous algorithm for this problem is due to Kirkpatrick and requires time $O(N \log N)$. In addition, a parallel version of this algorithm can be implemented in $O(\log N)$ time using $O(N)$ processors on a CREW PRAM.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

An Optimal Algorithm for Determining the Separation of Two Nonintersecting Simple Polygons*

Nancy M. Amato
Coordinated Science Laboratory and Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
amato@cs.uiuc.edu

Abstract

Given nonintersecting simple polygons P and Q, their *separation*, denoted by $\sigma(P,Q)$, is defined to be the minimum distance between their boundaries; a pair of points $p \in P$ and $q \in Q$ realize $\sigma(P,Q)$, if $d(p,q) = \sigma(P,Q)$. We present an optimal $\Theta(N)$ time algorithm for determining the separation of two disjoint simple polygons P and Q, and finding a pair of points (p,q), $p \in P$ and $q \in Q$, realizing $\sigma(P,Q)$, where $N = |P|+|Q|$. The best previous algorithm for this problem is due to Kirkpatrick and requires time $O(N \log N)$. In addition, a parallel version of this algorithm can be implemented in $O(\log N)$ time using $O(N)$ processors on a CREW PRAM.

1 Introduction

The problem of computing the minimum distance between two polygons P and Q has received much attention in the literature. The two most common variants of this problem are (i) finding the minimum distance between the boundaries of the two polygons, and (ii) finding the minimum distance between visible vertices on the boundaries of the two polygons, where two vertices $p \in P$ and $q \in Q$ are said to be visible if $\overline{pq}$ does not properly intersect P or Q. Note that in the former variant the minimum distance may be realized by two vertices or by a vertex and an edge, whereas in the latter variant, the minimum distance is realized by vertices only.

Computing the minimum distance between the boundaries of two nonintersecting polygons P and Q has been studied extensively; this problem is also known as finding the *separation* of the two polygons [DK90], denoted by $\sigma(P,Q)$. The separation problem has been addressed sequentially in various cases: when both P and Q are convex their separation can be determined in $\Theta(\log N)$ time [CW83, E85, CD87, DK90], when only one polygon is convex their separation can be determined in $\Theta(N)$ time [CWS85], and when neither polygon is convex their separation can be found in $O(N \log N)$ time by traversing the contour (external skeleton) between the two polygons in the generalized voronoi diagram [K79], where $N = |P| + |Q|$.

*This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy, U.S. Air Force) under contract N00014-90-J-1270.

In this paper we study the problem, both sequentially and in parallel, of determining $\sigma(P,Q)$ in the most general case, i.e., when P and Q are nonintersecting nonconvex simple polygons. We present a new sequential algorithm that computes $\sigma(P,Q)$ optimally in $\Theta(N)$ time; this improves on the complexity of the technique proposed in [K79] by a factor of $O(\log N)$. Although the separation and closest visible vertex problems seem to be closely related, it appears no benefit for either problem has been gained by this similarity in the past; in fact, the algorithms proposed for these two problems have differed greatly from one another. However, in this paper we show that the general strategy used in [A92] to optimally solve the closest visible vertex problem can indeed be used to optimally solve the separation problem. In particular, although the geometric basis and the actual implementation of each step of our algorithm differs significantly from that of the closest visible vertex algorithm of [A92], the overall structure of the two algorithms is the same and we show that many of the ideas first appearing in algorithms for the closest visible vertex problem [CW83, AMSS89, A92] prove valuable for the separation problem as well. Our algorithm can also be implemented in parallel providing, to our knowledge, the first parallel algorithm proposed for this problem; the complexity of our parallel version of the algorithm is $O(\log N)$ time using $O(N)$ processors on a CREW PRAM.

2 Nonintersecting Simple Polygons

As seems natural, the problem of determining the separation of two nonintersecting polygons P and Q, $\sigma(P,Q)$, is closely related to the problem of finding the closest visible vertex distance between P and Q, denoted by $CVV(P,Q)$. In fact, as we will see below, many of the techniques used to find $CVV(P,Q)$ can be adapted to our present problem of determining $\sigma(P,Q)$. Recall that $\sigma(P,Q)$ is either realized by a vertex pair or a vertex and a edge, and note that in the former case $\sigma(P,Q) = CVV(P,Q)$ and a pair realizing $CVV(P,Q)$ also realizes $\sigma(P,Q)$. Thus, in order to extend the techniques for computing $CVV(P,Q)$ to the more general problem of finding $\sigma(P,Q)$, we must augment them to consider vertices and edges of the two polygons, rather than only considering vertices. Our approach is loosely modeled after the method of computing $CVV(P,Q)$ described in [A92], which uses some ideas first appearing in the CVV algorithms of Wang and Chan [WC86] and Aggarwal *et al.* [AMSS89].

As in the CVV algorithms of [A92], [AMSS89], and [WC86], we reduce the problem of computing $\sigma(P,Q)$ to solving a number of restricted versions of the problem. Specifically, the primitive operation is: compute $\sigma(P',Q')$ where P' and Q' are *linearly separable* subchains of P and Q, respectively. In this section we describe how the decomposition is accomplished, and in the next section we show how each subproblem can be solved in time linear in its size. Although the actual decomposition of the original problem into subproblems is necessarily different than that used in the CVV algorithm of [A92], in both cases the decomposition is based on a sequence of line segments separating P and Q.

There are two general situations that we distinguish: the *containing* case, when $CH(P) \subset CH(Q)$ or $CH(Q) \subset CH(P)$, and the *non containing* case (see Fig. 1). Note that in the non containing case we can reduce, in $O(N)$ time, the problem of computing $\sigma(P,Q)$ to that of computing $\sigma(P',Q')$, where P' and Q' are subchains of P and Q, respectively. Let $CH(P)$ and $CH(Q)$ denote the convex hulls of P and Q, respectively, and recall that the convex hull of a simple polygon with m vertices can be found in time $O(m)$ (see, e.g., [PS85]). Even though $CH(P)$ and $CH(Q)$ may

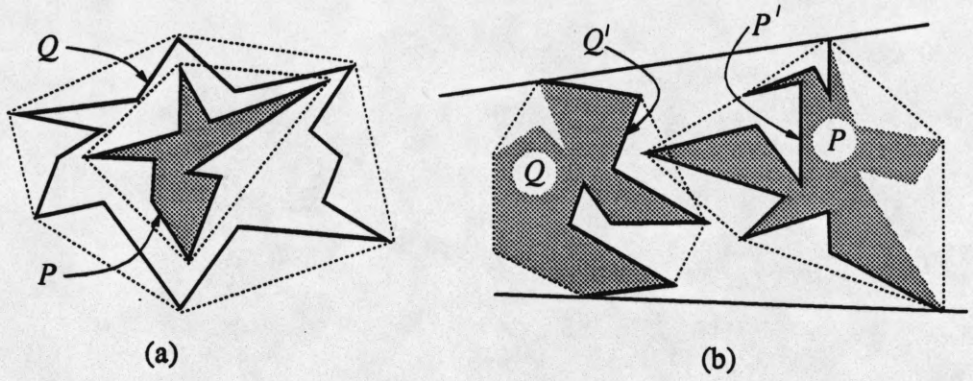

Figure 1: (a) The containing case. (b) The non containing case in which $\sigma(P,Q) = \sigma(P',Q')$.

intersect, it is a simple matter to verify that the convex polygon separation technique of Dobkin and Kirkpatrick [DK90] can be used to find the two lines tangent to $CH(P)$ and $CH(Q)$ in time $O(\log N)$ (see Fig. 1(b)). It is easy to verify that the facing portions of P and Q between these two tangents (P' and Q', respectively) contain all points of P and Q that could potentially realize $\sigma(P,Q)$, i.e., $\sigma(P,Q) = \sigma(P',Q')$ since any pair realizing $\sigma(P,Q)$ must be visible. In the non containing case, we assume that the subchains P' and Q' are indexed bottom-to-top, and that $|P'| = n_p$ and $|Q'| = n_q$; in the containing case, the vertices are also indexed consecutively, but there is no stipulation as to where the indexing begins.

In the following, the terms "highest" and "lowest" refer to positions of points (which follow the ordering of the indices of the vertices) on the polygons, and P_{p_i,p_j} denotes the subchain $(p_i, p_{i+1}, \ldots, p_j)$; Q_{q_i,q_j} is defined analogously. Following [A92], a sequence of (intersecting) line segments $S(P,Q) = (l_0, l_1, \ldots, l_{m-1})$, where $l_i = (l_i^-, l_i^+)$, is called a *separator* of P and Q if, for $0 \leq i < m$, (i) $l_i \cap l_{i+1} \neq \emptyset$, (ii) $l_i \cap l_j = \emptyset$ for $j \notin \{i-1, i, i+1\}$, (iii) l_i does not intersect the interior of P or Q, (iv) l_i^- and l_i^+ lie on the boundaries of P and/or Q, and (v) each l_i is maximal in the following sense: if $l_i^+ \in P$ and $l_{i+1}^+ \in Q$, then no point of Q above l_{i+1}^+ is visible from l_i, and if $l_i^+ \in P$ and $l_{i+1}^+ \in P$, then l_{i+1} intersects the highest point (a vertex) of Q that is visible from l_i; analogous statements hold if $l_i^+ \in Q$ (see Fig. 2). (In the containing case, all arithmetic is modulo m and we require $l_{m-1} \cap l_0 \neq \emptyset$.)

Note that property (v) ensures that the interior of each l_i, $0 \leq i < m$, intersects the boundary of P and/or Q; more specifically, this property guarantees that each l_i intersects the boundaries of both P and Q, either at an endpoint or an interior point. Let $p^+(i)$ and $q^+(i)$ denote the highest points of intersection of l_i with P and Q, respectively; for convenience, in the non containing case let $p^+(j) = p_1$ and $q^+(j) = q_1$ for $j < 0$, and let $p^+(k) = p_{n_p}$ and $q^+(k) = q_{n_q}$ for $k \geq m-1$. We assign a subchain P_i of P to segment l_i, $0 \leq i < m$, as follows; a subchain Q_i of Q is assigned to l_i analogously.

$$P_i = \begin{cases} P_{p^+(i-1),p^+(i)} & \text{if } l_i^+ \in P \text{ or } i = m-1 \\ P_{p^+(i-1),p^+(i+1)} & \text{otherwise} \end{cases}$$

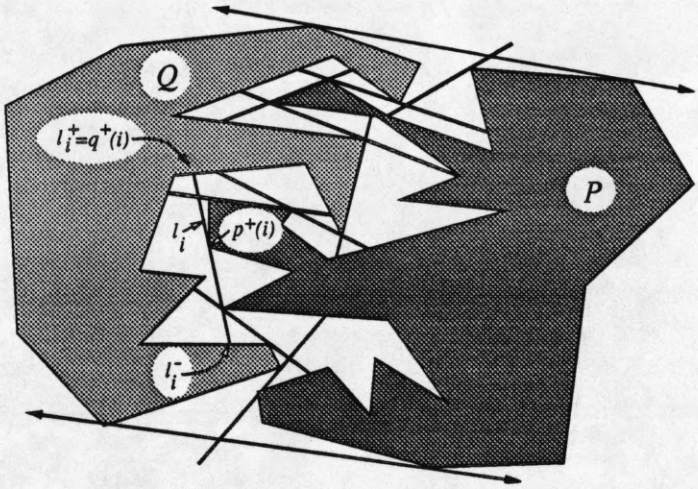

Figure 2: A set of separating line segments $S(P,Q)$ satisfying properties (i-v).

It is immediate to verify that no point of P or Q appears in more than three such subchains, and that each pair (P_i, Q_i), $0 \leq i \leq m$, is separated by l_i. The following lemma shows that $\sigma(P,Q)$ can be determined by computing $\sigma(P_i, Q_i)$, for all $0 \leq i \leq m$, i.e., if we independently examine pairs of subchains P_i and Q_i, $0 \leq i \leq m$, then we will not neglect any pair of points that could realize $\sigma(P,Q)$.

Lemma 1: Let P and Q be two nonintersecting simple polygons with $S(P,Q)$ as defined above. If $p \in P$ and $q \in Q$ are visible, $\overline{pq} \cap l_i \neq \emptyset$, and $\overline{pq} \cap l_{i-1} = \emptyset$, then $p \in P_i$ and $q \in Q_i$.

Proof: Let $p \in P$ and $q \in Q$ be visible points such that $\overline{pq} \cap l_i \neq \emptyset$, and $\overline{pq} \cap l_{i-1} = \emptyset$. Since $\overline{pq}$ does not intersect l_{i-1}, it must be that $p \geq p^+(i-1)$ and $q \geq q^+(i-1)$. Without loss of generality assume $l_i^+ \in Q$ (see Fig. 2), so that $q \leq q^+(i) = l_i^+$, i.e., $q \in Q_i$. Since, by property (v), no point $p' \geq p^+(i+1)$, $p' \in P$, is visible from l_i, and by hypothesis p and q are visible, we have $p \leq p^+(i+1)$, i.e., $p \in P_i$. □

Since it is obvious that $\sigma(P_i, Q_i) \geq \sigma(P,Q)$, for all $0 \leq i \leq m$, the above lemma establishes that $\sigma(P,Q) = \min\{\sigma(P_i, Q_i) | 0 \leq i \leq m\}$. We note here that a more complex decomposition technique is required for the closest visible vertex problem than the above specified method since it is not necessarily true that $CVV(P_i, Q_i) \geq CVV(P,Q)$, whereas it is trivially true that $\sigma(P_i, Q_i) \geq \sigma(P,Q)$; in fact, it is easy to construct examples in which $CVV(P_i, Q_i) < CVV(P,Q)$, so that the current decomposition strategy would incorrectly determine $CVV(P,Q)$. Assuming that $\sigma(P_i, Q_i)$ can be determined in $O(|P_i| + |Q_i|)$ time, as will be established in the next section, we have the following theorem.

Theorem 1: If P and Q are two non intersecting simple polygons, where $N = |P| + |Q|$, then $\sigma(P,Q)$, and a pair of points realizing it, can be computed optimally in $\Theta(N)$ time.

Proof: First, the technique of [A92] is used to compute a separator $S(P,Q)$ satisfying properties (i-v) in $O(N)$ time; this technique constructs $S(P,Q)$, in both the containing and non containing

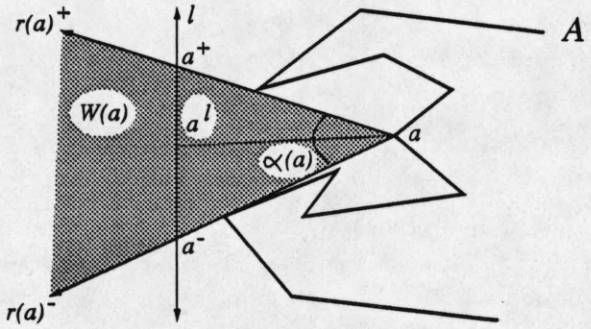

Figure 3: The wedge $W(a)$ for vertex $a \in A$.

cases, by modifying a shortest path between two distinguished vertices in a simple polygon that lies between P and Q. Then, since each point of P and Q appears in at most three subchains, and $\sigma(P_i, Q_i)$ can be determined in $O(|P_i| + |Q_i|)$ time, for all $0 \leq i \leq m$, we see that $\sigma(P, Q) = \min\{\sigma(P_i, Q_i) | 0 \leq i \leq m\}$ can be found in $O(\sum_{i=0}^{m} |P_i| + |Q_i|) = O(N)$ time. This is clearly optimal since it is shown in [CWS85] that $\Omega(N)$ time is required even if one of P or Q is convex. □

3 Linearly Separable Polygonal Chains

We now consider two disjoint polygonal chains P and Q that are separated by a line l; we allow vertices and edges of P and/or Q to lie on the separating line, but no edge may cross the line. The solution presented to the problem in this section is loosely patterned after the algorithm of [A92] for determining the closest visible vertices between two linearly separable polygonal chains. In particular, the algorithm consists of the same general steps as those of the algorithm in [A92], but the geometric foundation and actual implementation of each step differs in the algorithm presented here. For this reason, we briefly sketch the algorithm of [A92] for computing $CVV(P, Q)$ before turning to the problem of determining $\sigma(P, Q)$, where, in both cases, P and Q are linearly separable polygonal chains. Without loss of generality we assume that l is vertical, Q lies to the left of l, $|P| = n_p$ and $|Q| = n_q$. Let $(p_1, p_2, \ldots, p_{n_p})$ and $(q_1, q_2, \ldots, q_{n_q})$ denote the sequences of vertices of P and Q, respectively, indexed bottom-to-top. We begin with some useful definitions.

Consider polygonal chain A and line l, where no edge of A crosses l. A point v is *A-visible* from a point w if the segment (v, w) does not properly intersect A. Similarly, a point v is *A-visible* from a line l if some point of l is A-visible from v. For each $a \in A$, $W(a)$ denotes the interior of the maximal wedge with apex a whose interior contains no vertex of A and all points of l A-visible from a. The upper and lower rays defining $W(a)$ are denoted by $r(a)^+$ and $r(a)^-$, respectively, and a^+ and a^- denote $r(a)^+ \cap l$ and $r(a)^- \cap l$, respectively. (If $r(a)^+ \cap l = \emptyset$ then $a^+ = +\infty$, and if $r(a)^- \cap l = \emptyset$ then $a^- = -\infty$.) The angle, in $W(a)$, between the rays $r(a)^-$ and $r(a)^+$ is denoted by $\alpha(a)$. Finally, let l_a be the line perpendicular to l that passes through a, and denote $l \cap l_a$ by a^l (see Fig. 3). Throughout this paper, $d(u, v)$ denotes the Euclidean distance between points u and v, $x(v)$ the x-coordinate of point v, and $y(v)$ the y-coordinate of point v.

We first sketch the algorithm of [A92] for computing $CVV(P,Q)$, when P and Q are linearly separable polygonal chains, and then discuss how this technique can be adapted so that it will determine the separation of P and Q. The following lemmas, which where established in [A92] and [WC86], respectively, are used to form subchains P' and Q' of P and Q, respectively, so that $CVV(P',Q') = CVV(P,Q)$, and P' and Q' have a restricted form that can be exploited to calculate $CVV(P,Q)$ more easily.

Lemma 2 [A92]: Let P and Q be two polygonal chains that are separated by a line l, and let p be a vertex of P. If $p^l \notin W(p)$, then $CVV(p,Q) > CVV(P,Q)$, i.e., if p is not perpendicularly visible from l, then p cannot be a nearest vertex of P to Q. (Removal of these vertices creates a new chain monotone with respect to l.)

Lemma 3 [WC86]: Let P and Q be two polygonal chains that are separated by a line l such that P and Q are monotone with respect to l. If $\alpha(p) < 90°$, then $CVV(p,Q) > CVV(P,Q)$, i.e., p cannot be a nearest vertex of P to Q.

After pruning P and Q according to Lemmas 2 and 3, a new region $R() \subset W()$ is constructed for each remaining vertex of P and Q. (The precise structure for the $R()$ regions will be described in the separation algorithm.) The following lemma establishes that if (p,q) realizes $CVV(P,Q)$, then $p \in R(q)$ and $q \in R(p)$.

Lemma 4 [A92]: Let P and Q be two polygonal chains that are separated by a line l and monotone with respect to it, $\alpha(p) \geq 90°$, and $\alpha(q) \geq 90°$, for all vertices $p \in P$ and $q \in Q$. If $p \notin R(q)$ or $q \notin R(p)$ then (p,q) cannot realize $CVV(P,Q)$, $p \in P$ and $q \in Q$.

We next consider the $n_p \times n_q$ matrix M whose entries are defined as follows; if $n_p > n_q$ we instead consider the $n_q \times n_p$ matrix that is analogously defined. Let B be some constant that is greater than the maximal distance between P and Q. For notational convenience, we use $q < R(p)$ or $q > R(p)$ to indicate that q lies *below* $R(p)$ or *above* $R(p)$, respectively.

$$M[i,j] = \begin{cases} B + n_q - j & \text{if } q_j < R(p_i) \text{ or } p_i > R(q_j) \\ d(p_i, q_j) & \text{if } p_i \in R(q_j) \text{ and } q_j \in R(p_i) \\ \infty & \text{otherwise} \end{cases}$$

By Lemma 4, the above matrix M contains the distances between all pairs of vertices that are candidates for closest visible vertices. Note that if $q_j \notin R(p_i)$ or $p_i \notin R(q_j)$, then $M[i,j] \geq B + n_q - j \geq B$, and recall that B is greater than the maximal distance between P and Q, i.e., if (p_i, q_j) is not a candidate pair and (p_k, q_l) is a candidate pair, then $M[i,j] > M[k,l]$. Further note that, since $R(p) \subset W(p)$ and $R(q) \subset W(q)$, all candidate pairs (p,q) are visible. Thus, a minimum entry in M will give a closest visible pair for P and Q. We now give the following definitions. Following [AKMSW87], an $m \times n$ matrix consisting of real entries is called *monotone* if the minimum entry in its ith row lies below or to the right of the minimum entry in the $(i-1)$st row, for all $1 < i \leq m$. (If a row has several minima then we take the leftmost one.) Furthermore, a matrix is called *totally monotone* if each of its 2×2 submatrices (minors) is monotone. In [AKMSW87], it is shown that if an $m \times n$, $m \leq n$, matrix is totally monotone and every entry of the matrix can be computed in constant time, then a minimum entry in every row can be found in $O(m)$ time. The following lemma establishes that the algorithm of [AKMSW87] can be used to

find a minimum entry in each row of M in $O(N)$ time. ([AMSS89] was the first to use the totally monotone matrix techniques of [AKMSW87] for the closest visible vertex problem.)

Lemma 5 [A92]: Let P and Q be two polygonal chains that are separated by a line l and monotone with respect to it, $\alpha(p) \geq 90°$ and $\alpha(q) \geq 90°$, for all vertices $p \in P$ and $q \in Q$. If, for every $p \in P$ and $q \in Q$, the regions $R(p)$ and $R(q)$ are available, then the matrix M described above is totally monotone and each entry of M can be computed in constant time.

The above discussion gives us the following algorithm for computing $CVV(P,Q)$.

Algorithm: $CVV(P,Q)$

1. Eliminate those vertices of P and Q that are not perpendicularly visible from l (Lemma 2).
2. Form $W(p)$ and $W(q)$ for all remaining vertices $p \in P$ and $q \in Q$. Eliminate those vertices $p \in P$ and $q \in Q$ that have $\alpha(p) < 90°$ or $\alpha(q) < 90°$ (Lemma 3).
3. Determine the $R()$ regions for all vertices remaining under consideration.
4. Find the smallest entry in the matrix M.

The complexity of the above algorithm is easily seen to be $O(N)$, where $N = |P| + |Q|$. It is easy to verify that Steps 1-3 can be accomplished by linear scans of the relevant subchains. In particular, Step 1 can be implemented by a single bottom-to-top scan of P or Q. Step 2 requires a bottom-to-top (to identify the lower rays of $W()$) and a top-to-bottom (to identify the upper rays of $W()$) scan of P or Q; in fact, this process is essentially the same as a standard linear time algorithm for determining the convex hull of a simple polygon (see, e.g., [BE84]). Next, using the $W()$ regions, Step 3 is accomplished by a top-to-bottom and a bottom-to-top linear scan of both P and Q. (The precise structure of the $R()$ regions is given below.) Finally, Step 4 is accomplished by using the $O(N)$ time algorithm of [AKMSW87] to find a minimum in each row, and then determining the minimum of the row minima. Thus, each step is accomplished in $O(N)$ time, yielding the stated complexity of the algorithm.

We now return to the problem of determining the separation of P and Q, $\sigma(P,Q)$. Recall that $\sigma(P,Q)$ is either realized by a vertex-vertex pair or a vertex-edge pair; i.e., $\sigma(P,Q) = \min\{\sigma(P_v,Q_v), \sigma(P_e,Q_v), \sigma(P_v,Q_e)\}$, where P_v (Q_v) and P_e (Q_e) represent the vertices and (open) edges, respectively, of P (Q). Clearly $\sigma(P_v,Q_v) = CVV(P,Q)$, and can be found by the CVV algorithm described above. Thus, we now concentrate on the problem of determining $\sigma(P_e,Q_v)$; $\sigma(P_v,Q_e)$ can be found analogously. Our goal is to adapt the CVV algorithm of [A92] sketched above to the present scenario; the modified version of the CVV algorithm that computes $\sigma(P_e,Q_v)$ will be referred to as the CVE (closest vertex edge) algorithm. We first note that Lemmas 2 and 3 continue to hold if, rather than only vertices of P, we instead consider any point on the boundary of P, e.g., an interior point of an edge. This fact is readily verified by realizing that there is nothing special about any interior point of an edge that distinguishes it from a vertex, indeed, such an "interior point" could in fact be a vertex, and the lemmas would continue to hold. Thus, since $CVV(P,Q) \geq \sigma(P,Q)$, Lemmas 2 and 3, respectively, establish that we can eliminate all boundary points of P and vertices of Q that (i) are not perpendicularly visible from l (Lemma 2), and (ii) that have $\alpha() < 90°$ (Lemma 3).

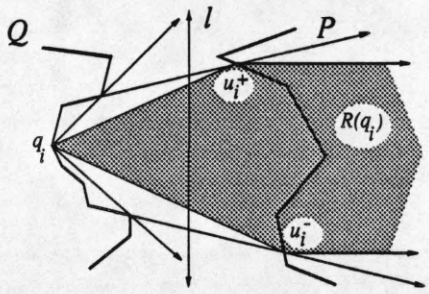

Figure 4: The region $R(q_i)$ for vertex $q_i \in Q_v$.

The CVE algorithm has a pattern analogous to that of the CVV algorithm. Specifically, we form $W()$ and then $R()$ regions for each *vertex* of Q and each *edge* of P, and then we define a totally monotone matrix whose entires contain the distances between all candidates for closest vertex edge pair and use the algorithm of [AKMSW87] to find a minimum entry in this matrix. Assuming that P and Q have been pruned according to Lemmas 2 and 3, we construct a new region $R(q) \subset W(q)$, for each remaining vertex $q \in Q$. (The $W()$ regions for each vertex of Q are defined as in the CVV algorithm.) Specifically, we associate with each vertex $q_i \in Q$, two (boundary) points of P, u_i^+ and u_i^-, as follows. The point u_i^+ is the highest point of P that satisfies (i) $y(q_i) \leq y(u_i^+) \leq y(u_{i+1}^+)$, and (ii) $u_i^+ \in W(q_i)$ and $q_i \in W(u_i^+)$ (i.e., q_i and u_i^+ are visible); if no such point exists then $u_i^+ = q_i^l$. Similarly, the point u_i^- is the lowest point of P that satisfies (i) $y(q_i) \geq y(u_i^-) \geq y(u_{i-1}^-)$, and (ii) $u_i^- \in W(q_i)$ and $q_i \in W(u_i^-)$; if no such point exists then $u_i^- = q_i^l$. (The y-coordinate of $u_{n_q+1}^+$ is assumed to be $+\infty$ and the y-coordinate of u_0^- is assumed to be $-\infty$.) The region $R(q_i)$ is bounded above by the segment (q_i, u_i^+) and the horizontal ray originating at u_i^+, and is bounded below by the segment (q_i, u_i^-) and the horizontal ray originating at u_i^- (see Fig. 4). (The $R()$ regions used in the CVV algorithm are very similar to the $R()$ regions described above for $q_i \in Q$, the only difference being that the points u_i^+ and u_i^- of P are restricted to *vertices* of P, rather than allowing them to be any boundary point of P.)

Our task now is to define, and construct, appropriate $W()$ and $R()$ regions for the *edges* of P. Since Lemma 2 applies to all points of P, the obvious analog of CVV, Step 1, is to remove the portions of P (including, perhaps, portions of edges) that are not perpendicularly visible from l; clearly this can be accomplished with a linear scan of P. In the CVV algorithm, the $W()$ regions are used for two distinct purposes: (i) to eliminate those vertices v with $\alpha(v) < 90°$ and (ii) to help construct the $R()$ regions efficiently. (The fact that all vertices remaining under consideration have $\alpha() \geq 90°$ is used in the proof of Lemma 4, which is crucial in establishing the correctness of the algorithm.) With these considerations in mind, we want to define and construct our $W()$ regions for edges so that it is easy to eliminate all points p on the boundary of P that have $\alpha(p) < 90°$.

Consider an edge $e = (p_i, p_{i+1}) \in P$, and the wedges $W(p_i)$ and $W(p_{i+1})$ as defined in the CVV algorithm. Recall that $r(p)^+$ ($r(p)^-$) passes by the highest (lowest) vertex of P that is visible from p; denote these vertices by $h(p)$ and $l(p)$, respectively. In the special (and unusual) case in which $h(p_i) = h(p_{i+1})$ and $l(p_i) = l(p_{i+1})$, we can easily determine which points of e have $\alpha() \geq 90°$ as follows. Let C_e denote the circle with diameter $d(l(p_i), h(p_i))$ that is centered at the midpoint of

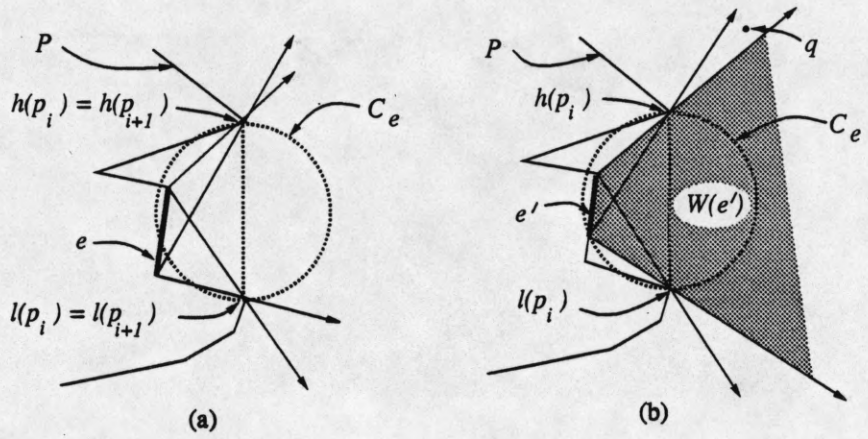

Figure 5: (a) All points on the portion of e that is external to C_e have $\alpha() < 90°$. (b) The region $W(e')$ for that portion e' of e that has $\alpha() \geq 90°$.

$(l(p_i), h(p_i))$ (see Fig. 5(a)). It is easy to verify that all points of e external to C_e have $\alpha() < 90°$, and all points of e on the boundary or internal to C_e have $\alpha() \geq 90°$. After eliminating those points of e with $\alpha() < 90°$, leaving at most one segment $e' = (p'_i, p'_{i+1})$, $W(e')$ is defined as follows: $W(e')$ is bounded above by the ray originating at p'_{i+1} and passing by $h(p_i)$, and is bounded below by the ray originating at p'_i and passing by $l(p_i)$ (see Fig. 5(b)). Clearly, $W(e')$ contains all points of l that are visible from *every* point of e'. Let Q' denote the portion of Q that is external to $W(e')$. Although there may be points of Q' that are visible to some point of e', it is a simple matter to verify that $\sigma(e', Q') > \sigma(P, Q')$, i.e., any point of Q external to $W(e')$ that is visible to some point of e' is closer to some other point of P than it is to e'. Without loss of generality, consider a point $q \in Q'$ that lies above $W(e')$ but is visible from some point of e' (see Fig. 5(b)); in this case it is easy to see that $\sigma(P, q) \leq d(h(p_i), q) < d(e', q)$. This property of $W(e')$ will be referred to as W.1.

In general, however, for each $e = (p_i, p_{i+1}) \in P$, we do not have $h(p_i) = h(p_{i+1})$ and $l(p_i) = l(p_{i+1})$. Our solution to this problem is simply to partition each edge into a number of segments so that within each segment this special property holds. In fact, as we will see later, this partitioning is rather simple and can be accomplished by the same scans that were used in the CVV algorithm to determine the $W()$ regions. In the following we denote the partitioned set of edges of P by $E' = \{e_i | 1 \leq i \leq n_e\}$, where $e_i = (v_i^-, v_i^+)$ and $y(v_i^-) < y(v_i^+)$; later we will show that $n_e = O(n_p)$.

Given the $W()$ regions for the partitioned set of edges, their $R()$ regions are defined similarly to those of the vertices $q \in Q_v$. We associate with each edge $e_i = (v_i^-, v_i^+) \in E'$ two *vertices*, u_i^- and u_i^+, of Q. The vertex u_i^+ is the highest vertex of Q that satisfies (i) $y(v_i^+) \leq y(u_i^+) \leq y(u_{i+1}^+)$, and (ii) $u_i^+ \in W(e_i)$ and $v_i^+ \in W(u_i^+)$. Similarly, the vertex u_i^- is the lowest vertex of Q that satisfies (i) $y(v_i^-) \geq y(u_i^-) \geq y(u_{i-1}^-)$, and (ii) $u_i^- \in W(e_i)$ and $v_i^- \in W(u_i^-)$. The region $R(e_i)$ is bounded above by (v_i^+, u_i^+) and the horizontal ray originating at u_i^+ and is bounded below by (v_i^-, u_i^-) and the horizontal ray originating at u_i^- (see Fig. 6); if u_i^+ (u_i^-) does not exist, then $R(e_i)$ is bounded above (below) by the horizontal ray originating at v_i^+ (v_i^-), i.e., $u_i^+ = v_i^+$ $(u_i^- = v_i^-)$. The sufficiency of the $R()$ regions is established by the following lemma.

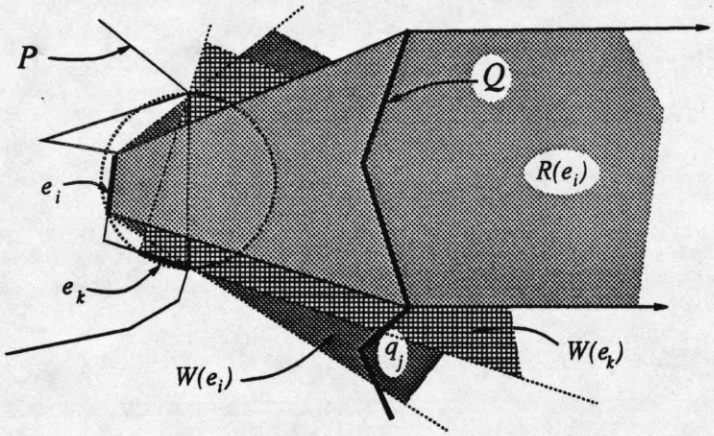

Figure 6: The region $R(e_i)$ for edge segment $e_i \in E'$.

Lemma 6: Let P and Q be two polygonal chains (or portions thereof) that are separated by a line l and monotone with respect to it, where $E' = \{e_i | 1 \leq i \leq n_e\}$ is the set of partitioned edges of P described above, and $\alpha(p) \geq 90°$ and $\alpha(q) \geq 90°$, for each point $p \in E'$ and each vertex $q \in Q$. If $p \notin R(q)$ or $q \notin R(e)$, for some $p \in e$, then (e, q) cannot realize $\sigma(P, Q)$, where $e \in E'$ and q is a vertex of Q.

Proof: Assume $q_j \notin R(e_i)$ or $e_i \cap R(q_j) = \emptyset$. Clearly if some $p \in e_i = (v_i^-, v_i^+)$ and q_j are not visible, then (e_i, q_j) cannot realize $\sigma(P, Q)$; therefore we assume some $p \in e_i$ and q_j are visible. Without loss of generality, assume q_j lies below $R(e_i)$, the other cases are similar. If $q_j \notin W(e_i)$, then, by W.1, $d(e_i, q_j) > d(l(v_i^-), q_j) \geq \sigma(P, Q)$, where $l(v_i^-)$ is the lowest vertex of P visible from v_i^-. Thus, we now assume $q_j \in W(e_i)$ and $q_j \notin R(e_i)$ (see Fig. 6); therefore, by definition of $R(e_i)$, there must be some edge $e_k \in E'$, $k < i$, such that q_j lies below $W(e_k)$. Again, by W.1, we have $d(e_k, q_j) > d(l(v_k^-), q_j) \geq \sigma(P, Q)$. Moreover, since $\alpha(x) \geq 90°$, for all $x \in e_k$, and both e_i and q_j lie outside $W(e_k)$, we have $d(e_i, q_j) > d(e_k, q_j) > \sigma(P, Q)$. □

The previous lemma establishes that we can use the $R()$ regions in the CVE algorithm in the same manner that they were used in the CVV algorithm. Specifically, we consider the $n_e \times n_q$ matrix M, defined as follows, where B is, once again, a constant greater than the maximal distance between P and Q; if $n_e > n_q$ we instead consider the $n_q \times n_e$ matrix that is analogously defined.

$$M[i,j] = \begin{cases} B + n_q - j & \text{if } q_j < R(e_i) \text{ or } e_i > R(q_j) \\ d(e_i, q_j) & \text{if } p \in R(q_j) \text{ and } q_j \in R(e_i), \text{ for some } p \in e_i \\ \infty & \text{otherwise} \end{cases}$$

It is clear, by Lemma 6, that M contains the distances between all edge-vertex pairs that are candidates for $\sigma(P, Q)$, and thus a minimum entry in M will yield $\sigma(P_e, Q_v)$. The following lemma establishes that the algorithm of [AKMSW87] can be used to find the minimum of each row in M.

Lemma 7: Let P and Q be two polygonal chains (or portions thereof) that are separated by a line l and monotone with respect to it, where $E' = \{e_i | 1 \leq i \leq n_e\}$ is the set of partitioned edges of P,

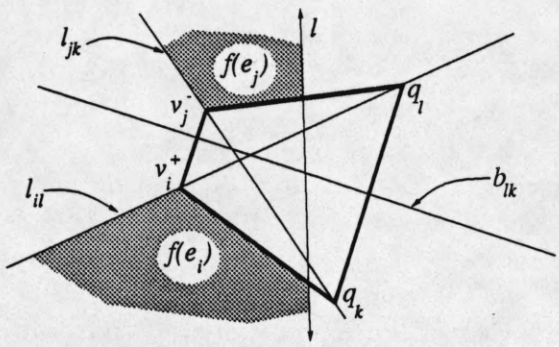

Figure 7: The edges e_i and e_j must lie in $f(e_i)$ and $f(e_j)$, respectively. In this case $d(e_i, q_k) < d(e_i, q_l)$ and $d(e_j, q_l) < d(e_j, q_k)$.

and $\alpha(p) \geq 90°$ and $\alpha(q) \geq 90°$ for each point $p \in E'$ and each vertex $q \in Q$. If, for every $e \in E'$ and $q \in Q_v$, the $R()$ regions are available, then the matrix M described above is totally monotone and each entry can be computed in constant time.

Proof: It is clear that each $M[i,j]$ can be computed in constant time since we are given $R(e_i)$ and $R(q_j)$, $1 \leq i \leq n_e$ and $1 \leq j \leq n_q$.

We first establish (i) if $M[i,j] = B+n_q-j$, then $M[i',j] = B+n_q-j$, and (ii) if $M[i,j] = d(e_i, q_j)$, then $M[i',j] \neq \infty$, for all $1 \leq i < i' \leq n_e$. To establish claim (i) we argue as follows. Since $M[i,j] = B + n_q - j$, either e_i lies above $R(q_j)$ or q_j lies below $R(e_i)$. If e_i lies above $R(q_j)$, then $e_{i'}$ must also lie above $R(q_k)$, and $M[i',j] = B + n_q - j$. If q_j lies below $R(e_i)$, then q_j must also lie below $R(e_{i'})$, and $M[i',j] = B + n_q - j$. The following argument establishes claim (ii). Since $M[i,j] = d(e_i, q_j)$, it must be that $x \in R(q_j)$, for some $x \in e_i$, and $q_j \in R(e_i)$, and if $M[i',j] = \infty$, then it must be that $e_{i'}$ lies below $R(q_j)$ or q_j lies above $R(e_{i'})$. However, $e_{i'}$ cannot lie below $R(q_j)$ since $x \in R(q_j)$, for some $x \in e_i$, and $e_{i'}$ is above e_i, and q_j cannot lie above $R(e_{i'})$ since $q_j \in R(e_i)$ and $R(e_i)$ is below $R(e_{i'})$.

To see that M is totally monotone, consider the 2×2 submatrix $\widehat{M}$ of M formed by the intersection of rows i and j, $i < j$, and columns k and l, $k < l$; note that $\widehat{M}$ is *not* monotone if and only if (a) $M[i,l] < M[i,k]$ and (b) $M[j,k] \leq M[j,l]$. It is easy to verify that $(M[i,k], M[i,l]) \in \{(B+n_q-k, B+n_q-l), (B+n_q-k, d(e_i,q_l)), (d(e_i,q_k), d(e_i,q_l)), (B+n_q-k, \infty), (d(e_i,q_k), \infty), (\infty, \infty)\}$ (an analogous set can be constructed for the possible values of $(M[j,k], M[j,l])$, by replacing i with j). Since condition (a) cannot be satisfied if $M[i,l] = \infty$, the first three values in the above set are the only possibilities in which $\widehat{M}$ might not be monotone.

If $M[i,k] = B+n_q-k$ and $M[i,l] = B+n_q-l$, then, as was established above, $M[j,k] = B+n_q-k$ and $M[j,l] = B + n_q - l$, and $\widehat{M}$ is monotone. If $M[i,k] = B + n_q - k$ and $M[i,l] = d(e_i, q_l)$, then, as established above, $M[j,k] = B + n_q - k$ and $M[j,l] \in \{B + n_q - l, d(e_j, q_l)\}$; since, $M[j,k] = B + n_q - k > B + n_q - l > d(e_j, q_l)$, $\widehat{M}$ is monotone.

The final possibility to consider is when $M[i,k] = d(e_i, q_k)$ and $M[i,l] = d(e_i, q_l)$. In this case, as established above, $M[j,k] \neq \infty$ and $M[j,l] \neq \infty$, i.e., $(M[j,k], M[j,l]) \in \{(B+n_q-k, B+n_q-l), (B + n_q - k, d(e_j, q_l)), (d(e_j, q_k), d(e_j, q_l))\}$. If $M[j,k] = B + n_q - k$, then $\widehat{M}$ is monotone, i.e.,

$M[j,k] > M[j,l]$ since $B + n_p - k > B + n_q - l > d(e_j, q_l)$. The remaining case is when $M[j,k] = d(e_j, q_k)$ and $M[j,l] = d(e_j, q_l)$. This means that $v_j^- \in R(q_k)$, $v_j^- \in R(q_l)$, $v_i^+ \in R(q_k)$, and $v_i^+ \in R(q_l)$, since e_i lies below e_j, q_k lies below q_l, and E' is monotone with respect to l. Thus, v_i^+ and v_j^- are both visible from q_k and q_l, and v_i^+, v_j^-, q_k, and q_l form a convex quadrilateral. It is easy to verify that e_i must lie in the region $f(e_i)$ bounded by l, the segment (v_i^+, q_k), and the line, l_{il}, through the segment (v_i^+, q_l) (see Fig. 7); by definition e_i lies to the left of l, e_i must lie below (v_i^+, q_k) since v_i^+ and q_k are visible, and e_i must lie below l_{il} since $q_l \in R(e_i) \subset W(e_i)$. Similarly, it can be shown that $e_j \in f(e_j)$. Let b_{kl} denote the perpendicular bisector of (q_k, q_l); clearly b_{kl} cannot intersect *both* $f(e_i)$ and $f(e_j)$. Since every point below (above) b_{kl} is closer to e_i (e_j) than it is to q_l (q_k), it is clear that $d(e_i, q_k) < d(e_i, q_l)$ and/or $d(e_j, q_l) < d(e_j, q_k)$. Thus, it is not possible that $d(e_i, q_l) < d(e_i, q_k)$ and $d(e_j, q_k) \leq d(e_j, q_l)$, which is the only case in which $\widehat{M}$ is *not* monotone. □

Thus, the CVE algorithm can be summarized as follows.

Algorithm: $CVE(P_e, Q_v)$

1. Eliminate the portions of P and vertices of Q that are not perpendicularly visible form l (Lemma 2).

2. Partition the edges of P into a new set set of edges E' so that $h(x) = h(y)$ and $l(x) = l(y)$ for all $x, y \in e$, for each edge $e \in E'$. Eliminate those edges, or portions thereof, that do not have $\alpha() \geq 90°$, and form the $W()$ regions for all remaining edges (Lemma 3). Form the $W()$ regions for each vertex of Q and eliminate those vertices with $\alpha() < 90°$ (Lemma 3).

3. Using the $W()$ regions for edges of P and vertices of Q, determine the $R()$ regions for each remaining edge of P, and vertex of Q.

4. Find the smallest entry in the totally monotone matrix M defined above.

It is clear that all steps, with the exception of Step 2, can be accomplished with the same techniques and within the same time bounds as in the CVV algorithm, i.e., they are all accomplished in $O(N)$ time, where $N = |P| + |Q|$. We now explain how an edge $e = (p_i, p_{i+1}) \in P$ can be partitioned into segments s_j, $1 \leq j \leq k$, where $s_j = (u_j, u_{j+1})$, $u_1 = p_i$ and $u_{k+1} = p_{i+1}$, so that $h(x) = h(y)$ and $l(x) = l(y)$ for all $x, y \in s_j$, $1 \leq j \leq k$. (Recall that $h(x)$ ($l(y)$) denote the highest (lowest) vertices of P visible from $x \in P$.) Without loss of generality we show how to partition e into segments s_j, $1 \leq j \leq k'$, $k' \leq k$, so that $h(x) = h(y)$ for all $x, y \in s_j$; a similar process is used to ensure that $l(x) = l(y)$ for all $x, y \in s_j$. Clearly, if $h(p_i) = h(p_{i+1})$ no subdivision is necessary. Next note that $h(p_i) \geq p_{i+1}$, and if $h(p_i) = p_{i+1}$ then $h(x) = p_{i+1}$ for all $x \in e$ and no subdivision is required (recall that we are dealing with open line segments since the vertices were tested separately). So we assume $h(p_i) > p_{i+1}$ and $h(p_i) \neq h(p_{i+1})$; it is easy to verify that $h(p_i) > h(p_{i+1})$, i.e., it is not possible that $p_{i+1} < h(p_i) < h(p_{i+1})$. Recall that the rays $r(p)^+$ (and thus the vertices $h(p)$) were determined by a top-to-bottom scan of P, and moreover, that this scanning process actually computed the successive convex hulls $C_i = CH(p_i \cup p_{i+1} \cup ... \cup p_{n_p})$, $1 \leq i \leq n_p$, i decreasing. Thus, after determining $h(p_{i+1})$ we have C_{i+1}, and note that $h(p_i) \in C_{i+1}$, and in particular $r(p_i)^+$ is the ray originating at p_i and tangent to C_{i+1}. Now it is clear that the scanning (up) of C_{i+1} to form C_i will actually determine the points on e which have different $h()$ values (see Fig. 8). Thus, once again, the same scanning process used in the CVV algorithm to

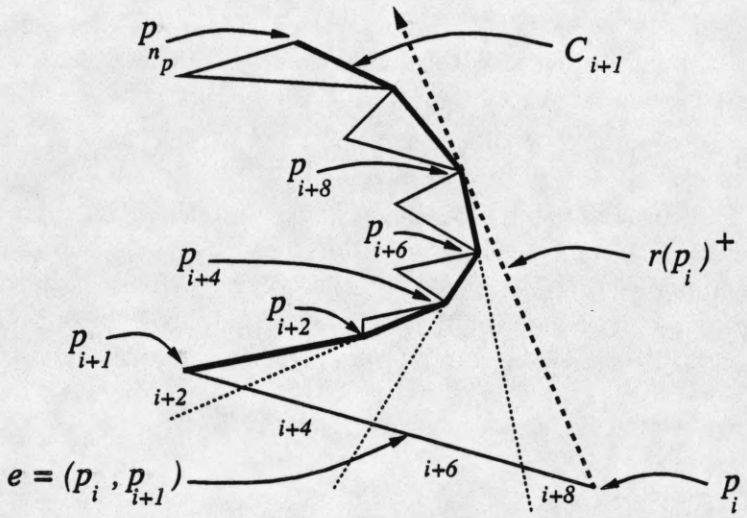

Figure 8: The segments of $e = (p_i, p_{i+1})$ are labeled with the index of the highest vertex of P that is visible from them.

form the $W()$ regions can be used in the CVE algorithm. We can also see that the number of new endpoints ("vertices") introduced in this manner is at most n_p, since a particular edge of C_i can only introduce one point as it disappears, and if it doesn't disappear it cannot introduce any new points. Thus, the total number of endpoints (and thus edges) we end up with after partitioning all edges is at most $3n_p$, i.e., the original n_p endpoints combined with the additional n_p from the top-to-bottom and the additional n_p from the bottom-to-top scans of the partitioning process.

The above discussion establishes the following theorem.

Theorem 2: If P and Q are two linearly separable polygonal chains, where $N = |P| + |Q|$, then $\sigma(P, Q)$, and a pair of points realizing it, can be computed optimally in $\Theta(N)$ time.

Proof: Since $\sigma(P_v, Q_v)$ can be computed by the CVV algorithm in $O(N)$ time, and the above discussion establishes that $\sigma(P_e, Q_v)$ and $\sigma(P_v, Q_e)$ can be determined in $O(N)$ time by the CVE algorithm, it is clear that $\sigma(P, Q) = \min\{\sigma(P_v, Q_v), \sigma(P_e, Q_v), \sigma(P_v, Q_e)\}$, can be found in $O(N)$ time as well. This is clearly optimal since it is shown in [CWS85] that $\Omega(N)$ time is required even if one of P or Q is convex. □

4 A Parallel Implementation

The parallel complexity of the separation problem has only been addressed in the special case in which both polygons are convex: [AG88] gives an algorithm for a CREW PRAM having $N^{1/k}$ processors that requires $O(k^{1+\epsilon})$ time, and in [DaK89] it is shown that time $O(\log N/(1 + \log p))$ is sufficient on a CREW PRAM with p processors. Note that both of the above algorithms essentially achieve constant time if a linear number of processors is available. Since voronoi diagrams (and thus

also generalized voronoi diagrams) can be constructed in $O(\log^2 N)$ time using $O(N)$ processors on a CREW PRAM [ACG87, ACGOY88], the separation of two nonintersecting simple polygons, in the most general case, can be found within these same time and processor bounds by a naive parallelization of Kirkpatrick's sequential approach [K79].

As we have seen, the separation problem can be solved by the same primitive operations and techniques that were used to solve the CVV problem in [A92]. Since [A92] also gives a parallel implementation of the CVV algorithm requiring $O(\log N)$ time and using $O(N)$ processors on a CREW PRAM, it is clear that $\sigma(P, Q)$ can be computed within these same time and processor bounds, even when P and Q are nonintersecting, nonconvex simple polygons. Thus, a parallel implementation of the separation algorithm given here improves the time complexity by a factor of $O(\log N)$ over the naive adaptation of the parallel voronoi diagram algorithms of [ACG87, ACGOY88].

References

[ACGOY88] A. Aggarwal, B. Chazelle, L. Guibas, C. Ó'Dúnlaing, C. Yap, Parallel Computational Geometry, *Algorithmica* **3** (1988), pp. 293-327.

[AKMSW87] A. Aggarwal, M. Klawe, S. Moran, P. W. Shor, R. Wilber, Geometric Applications of a Matrix Searching Algorithm, *Algorithmica* **2**(2) (1986), pp. 195-208.

[AMSS89] A. Aggarwal, S. Moran, P. Shor, S. Suri, Computing the Minimum Visible Vertex Distance Between Two Polygons, *Proc. of WADS 1989*, and *Lecture Notes in Computer Science* **382**, Eds. F. Dehne, J. R. Sack, N. Santoro, Springer Verlag, Berlin, (1989), pp. 115-134.

[A92] N. Amato, Computing the Minimum Visible Vertex Distance Between Two Nonintersecting Simple Polygons, *Proceedings of the 1992 Conference on Information Sciences and Systems* Vol. II, Princeton, NJ, (1992), pp. 800-805. (Also, Coordinated Science Laboratory Tech. Report, No. UILU-ENG-92-2206 (ACT 120), University of Illinois at Urbana-Champaign.)

[ACG87] M. Atallah, R. Cole, and M. Goodrich, Cascading Divide-and-Conquer: a Technique for Designing Parallel Algorithms, *Proc. of 28th Annual Symposium on Foundations of Computer Science* (1987), pp. 151-160.

[AG88] M. Atallah and M. Goodrich, Parallel Algorithms for Some Functions of Two Convex Polygons, *Algorithmica* **3** (1988), pp. 535-548.

[BE84] B. Bhattacharya and H. El Gindy, A New Linear Convex Hull Algorithm for Simple Polygon, *IEEE Inform. Theory* **c29** (1984), pp. 571-573.

[CD87] B. Chazelle and D. Dobkin, Intersection of Convex Objects in Two and Three Dimensions, *Journal of the ACM* **34** (1987), pp. 1-27.

[CW83] F. Chin and C. Wang, Optimal Algorithms for the Intersection and the Minimum Distance Problems between Planar Polygons, *IEEE Trans. on Computers* **c32** (1983), pp. 1205-1207.

[CWS85] F. Chin, C. Wang, J. Sampson, An Unifying Approach for a Class of Computational Geometry Problem, *The Visual Computer - Internat. Journal of Computer Graphics* **1**(2) (1985), pp. 124-133.

[DaK89] N. Dadoun and D. Kirkpatrick, Optimal Parallel Algorithms for Convex Polygon Separation, Technical Report #89-21, Department of Computer Science, University of British Columbia, Vancouver BC, Canada (1989).

[DK90] D. Dobkin and D. Kirkpatrick, Determining the Separation of Preprocessed Polyhedra - A Unified Approach, *ICALP* (1990), pp. 400-413.

[E85] H. Edelsbrunner, On Computing the Extreme Distance Between Two Convex Polygons, *Journal of Algorithms* **6** (1985), pp. 213-224.

[K79] D. Kirkpatrick, Efficient Computation of Continuous Skeletons, *Proc. of the 20th Annual Symposium on the Foundations of Computer Science* (1979), pp. 18-27.

[PS85] F. Preparata and M. Shamos, *Computational Geometry*, Springer-Verlag, New York (1985).

[WC86] C. A. Wang and E. P. F. Chan, Finding the Minimum Visible Vertex Distance Between Two Nonintersecting Simple Polygons, *Proc. of the Second ACM Annual Symposium on Computational Geometry* (1986), pp. 34-42.