Optimal Parallel Algorithms For Convex Polygon Separation[1]

N. Dadoun, D.G. Kirkpatrick

Technical Report 89-21
September, 1989

Abstract

Cooperative parallel algorithms are presented for determining convex polygon separation and constructing convex polygon mutual tangents. Given two n-vertex convex polygons, using k CREW processors ($1 \leq k \leq n$), each of these algorithms has an $\Theta(\log n/(1 + \log k))$ time bound. This provides algorithms for these problems which run in $O(\log n)$ time sequentially or in constant time using a quasi-linear (n^{α} for some $\alpha > 0$) number of processors.

These algorithms make use of hierarchical data structures to solve their respective problems. The polygonal hierarchy used by our algorithms is available implicitly (with no additional preprocessing) within standard representations of polygons.

---

[1] This research was supported in part by the Natural Sciences and Engineering Research Council of Canada. A preliminary version of these results were reported in [DaK4].

Optimal Parallel Algorithms For Convex Polygon Separation[1]

N. Dadoun,
D.G. Kirkpatrick

Department of Computer Science,
University of British Columbia,
Vancouver, BC, CANADA V6T 1W5

Abstract

Cooperative parallel algorithms are presented for determining convex polygon separation and constructing convex polygon mutual tangents. Given two n-vertex convex polygons, using k CREW processors ($1 \leq k \leq n$), each of these algorithms has an $\Theta(\log n/(1 + \log k))$ time bound. This provides algorithms for these problems which run in $O(\log n)$ time sequentially or in constant time using a quasi-linear (n^{α} for some $\alpha > 0$) number of processors.

These algorithms make use of hierarchical data structures to solve their respective problems. The polygonal hierarchy used by our algorithms is available implicitly (with no additional preprocessing) within standard representations of polygons.

1.0 Introduction

We present parallel algorithms for a number of convex polygon problems which make use of a hierarchical representation. We concentrate on cooperative algorithms, algorithms for which multiple processors are employed to answer a single query. These algorithms are competitive with previously known algorithms which address the same problems and provide constant time solutions

---

[1] A preliminary version of these results were reported in [DaK4].

when a quasi-linear (n^{α} for some $\alpha > 0$) number of processors is available. They have the interesting property that as the number of processors available is reduced to one, the time bounds increase smoothly to provide sequential algorithms which match the best sequential times known for the same problems.

We assume a synchronous parallel model of computation in which all processors have access to a global shared memory which is used for all communication between processors. Our algorithms use the concurrent-read exclusive-write (CREW) rule for accessing the common memory. The implications of exclusive-read with respect to our algorithms are addressed in Section 5.

In the design of parallel algorithms, a major goal is the minimization of the product T(n)*P(n) where T(n) is the time bound of the algorithm and P(n) is the number of processors required by the algorithm. In general, an algorithm exhibits *optimal speedup* if its time-processor product matches that of the best known sequential algorithm [KR]. Usually, a parallel algorithm provides a corresponding sequential algorithm with an O(T(n)*P(n)) time bound by employing a single processor to simulate P(n) parallel processors.

When the sequential algorithm for a given problem is already sublinear, as is the case for the problems addressed here, an optimal speedup is clearly not possible for arbitrarily large numbers of processors. Hence we use the term *optimal* in a more classical sense to refer to algorithms whose upper bounds match their respective problem's lower bounds. The algorithms presented here run in constant time with a quasi-linear number of processors and as the number of processors available decreases, the time bound increases smoothly to match the best known sequential time bound with a single processor.

In a sense, the algorithms described here have a 'decreasing return' in their application of parallelism; the speedup may only be polylogarithmic in the number of processors. One might ask why issues of massive parallelism should be explored for problems which already admit a sublinear

sequential algorithm. In addition to their purely theoretical interest, such problems may appear as recurring subproblems in a setting for which there are a varying number of processors available.

We denote by n the number of vertices of a typical input polygon. The number of processors available, k for $1 \leq k \leq n$, is an input parameter of our algorithms. Although our algorithms are applicable to the entire range of k, occasionally minor modifications are necessary to ensure correct behaviour for small values of k. When these modifications are straightforward, we omit their details in the interest of an uncluttered presentation.

Throughout the paper, all polygons considered are convex. We assume a very natural presentation of the input; the vertices of a convex polygon are provided in an array in clockwise order of their appearance on the boundary. As described in the next section, this representation contains enough structure to provide the hierarchy of polygons used in our algorithms without preprocessing. This is in contrast to the corresponding polyhedral representation in which each of the polyhedron hierarchies must be constructed explicitly [DaK3].

The problems addressed are separation, and separating/common tangents. Determining the *separation* of two convex polygons is a generalization of detecting their intersection; it is the identification of a witness to the minimum distance separating them. If the polygons do intersect the separation is zero and the witness is a point in their intersection.

A *separating* (respectively, *common*) *tangent* of two convex polygons P and Q is a line tangent to both polygons such that P and Q lie in different half-planes (respectively, the same half-plane). The separating tangents delimit the set of possible separating lines for P and Q. The common tangent in which both polygons lie in the half-plane below (respectively, above) the tangent is known as the upper (respectively, lower) common tangent. Determining the common tangents of two convex polygons is a step in forming the convex hull of their union.

Previous work has provided logarithmic time sequential algorithms for common tangents [OvL] and convex polygon separation [E]. Viewing the boundary of each polygon as a sequence of

values, both of these algorithms involve a 'binary search' approach; within each iteration, the midpoints of the remaining sequence of vertices of each polygon are compared. Then, a case analysis is applied to limit the remaining search to one half of the sequence associated with one of the polygons. The iterations continue until one of the two sequences is reduced to a constant size. Therefore, there are a logarithmic number of constant time iterations. Although these algorithms do not seem to admit to straightforward parallel implementations in our model, their binary search approach forms the basis for the k-way search algorithms developed here.

Chazelle and Dobkin [CD], Dobkin and Kirkpatrick [DK1][DK2][DK3][DK4] and Avis, El Gindy and Seidel [AES] have presented sequential algorithms for a number of polygon and polyhedron separation problems. Some of the results presented here can be viewed as non-trivial parallel generalizations of their work.

The parallel solutions presented by Atallah and Goodrich [AG1] for the common tangents and separation problems are described for a quasi-linear number of processors. However, the natural extensions of their algorithms for n-vertex convex polygons run in $O(\log^2 n/(1 + \log k)^2)$ parallel time using k CREW processors. Therefore, although they provide constant time algorithms (with a sublinear number of processors) for the problems which they address, they do not achieve the theoretical maximum speed-up of log k (cf. Section 5). In particular, since they embed a point-polygon test within each iteration of their polygon-polygon algorithms, as k the number of processors available decreases and approaches one, the natural extensions of their algorithms run in $O(\log^2 n)$ sequential time. We extend the basic design of their algorithms using convexity and properties of the inner polygonal hierarchy of Dobkin and Kirkpatrick [DK1].

The results reported here unify and extend the work cited above. In Section 2, we present details of the hierarchical representations which are used in the design of our algorithms. In Sections 3 & 4, we describe algorithms which solve separation and separating tangent queries for n-vertex convex polygons in $O(\log n/(1 + \log k))$ time using k CREW processors ($1 \leq k \leq n$). In

Section 5, we present a reduction which provides the matching lower bounds for the problems addressed here.

2.0 Polygon Hierarchies

All polygons considered are convex and are assumed to be presented in an array listing the vertices in clockwise order. For simplicity, we assume that no three vertices of a given polygon are collinear. We denote the set of vertices of polygon P by V(P). *Segment* ab is the line segment whose endpoints are vertices a and b. *Line* ab is the line which contains segment ab. *Ray* ab is the half-line directed from a to b which contains segment ab.

A convex polygon P can be defined as the intersection of a set of bounding half-planes. Hence, we define the edges as having a direction (clockwise) around the polygon such that each edge ab (defined on distinct vertices a and b) defines two closed bounding half-planes H_{ab}^{+} (on the right side of the edge from a to b) and H_{ab}^{-} (on the left side of the edge from a to b). Note that within this definition, a and b need not be consecutive vertices of P. This notation is generalized to define corresponding closed half-planes H_r^{+} and H_r^{-} with respect to any ray r.

Our algorithms make use of a hierarchical representation of convex polytopes [DK1] [DK2] [DK3] [DK4]. In two dimensions, this hierarchical representation can be motivated by a simple observation: Given a convex polygon P, any subsequence of P's vertices defines another convex polygon completely contained within P. For a convex polygon P, a (nested) sequence of convex polygons $P^0, P^1, \ldots, P^N$ is said to be an (*inner*) *polygon hierarchy* of P if

i) $P^0 = P$;

ii) $|P^N| = 3$;

iii) $V(P^{i+1}) \subset V(P^i)$ and $|P^{i+1}| < |P^i| / \alpha$ for some constant $\alpha > 1$ and

iv) there is at most some constant number β of vertices of P^i between any two consecutive vertices of P^{i+1}.

Conditions (ii) and (iii) ensure that $N = O(\log n)$ and condition (iv) ensures that the vertices of P^{i+1} are evenly selected from the vertices of P^i. A sequence of polygons $H(P) = P^0, P^1, \ldots, P^N$ that forms a polygon hierarchy is described simply: if the vertices of P are $p_0, p_1, \ldots p_{n-1}$, then the vertices of P^t are taken to be those vertices of P whose index is congruent to 0 MOD 2^t. This hierarchical representation has been used to solve a number of number of geometric separation problems in a sequential model of computation [DK4]. The algorithms which use this representation generally have the same basic design: Having answered the separation query for hierarchy element P^i, step to element P^{i-1} and use a constant number of tests to answer the query for P^{i-1}.

One of our goals is to demonstrate that this hierarchical representation is adapted easily for use in cooperative search queries. This is accomplished by relating the values α and β in conditions (iii) and (iv) of the polygon hierarchy definition to k, the number of processors we have available. A sequence of polygons $H_k(P) = P_k^0, P_k^1, \ldots, P_k^M$ that forms an *accelerated polygon hierarchy* attuned to the value k ($2 \leq k \leq n$) is described accordingly: if the vertices of P are $p_0, p_1, \ldots p_{n-1}$, then the vertices of P_k^t are taken to be those vertices of P whose index is congruent to 0 MOD k^t. Therefore, this accelerated hierarchy has $O(\log n/(1 + \log k))$ elements.

Viewed explicitly in this way, this hierarchical representation can be used by k processors to solve a point-polygon separation query. Each iteration of the algorithm deals with an element of the polygon hierarchy which is a "current approximation" of the initial polygon. When a point which realizes the separation property of interest is localized, the algorithm steps to the preceding element of the hierarchy which is a refinement of the current polygon approximation. Solving a polygon-polygon separation query is analogous, the algorithm deals with a current approximation of each polygon; an iteration of the algorithm steps to the preceding element of (at least) one of the polygon hierarchies[1].

---

[1] It is worth noting that Atallah and Goodrich's [AG1] approach is an informal variant of this paradigm. Their algorithms take a step by refining the region of interest dynamically using a k-way subdivision.

We maintain the invariant that when examining an element of the polygon hierarchy, only a portion of size O(k) need be considered. The algorithm initializes this invariant by examining the last element of the sequence; this element forms the coarsest approximation of P and is of size O(k). Given P_k^{i+1}, from the portion of size O(k) under consideration, the algorithm identifies a constant size portion which is guaranteed to contain a point which realizes the separation property of interest. This expands into a portion of size O(k) in P_k^i, and maintains the invariant.

When a polygon P is presented in an array, all elements of the hierarchy (for all values of k) are available implicitly by indexing.[1] To avoid excess notation, we denote the current hierarchy element P_k^t by $\tilde{P}^t$ (or, more commonly, $\tilde{P}$ when the context is clear or unimportant). At any given point within an algorithm, a vertex may have two different labels, its index in the original polygon P and its index in the current hierarchy element $\tilde{P}$. To avoid possible ambiguities, we adopt some notational conventions: The vertex p_t is the t^{th} vertex of polygon P, and the vertex $\tilde{p}_r$ is the r^{th} vertex of the current hierarchy element $\tilde{P}$. The sequence of vertices of polygon P between p_s and p_t inclusive is denoted by $P[p_s, p_t]$, consecutive subscripts refer to consecutive vertices on P. The sequence of m vertices of interest within hierarchy element $\tilde{P}$ between p_s (= $\tilde{p}_1$) and p_t (= $\tilde{p}_m$) inclusive is denoted by $\tilde{P}[\tilde{p}_1, \tilde{p}_m]$, consecutive subscripts refer to consecutive vertices on $\tilde{P}$. For example, $P[\tilde{p}_i, \tilde{p}_{i+1}]$ refers to the complete sequence of vertices on polygon P between two vertices $\tilde{p}_i$ and $\tilde{p}_{i+1}$ inclusive which are consecutive on the current hierarchy element $\tilde{P}$. In the discussion of polygon separation where the separation may be realized by a point which is not necessarily a vertex, we occasionally abuse this notation by using $P[\tilde{p}_i, \tilde{p}_{i+1}]$ to refer to the complete sequence of points on the boundary of P between two vertices $\tilde{p}_i$ and $\tilde{p}_{i+1}$ inclusive.

Given hierarchy element $\tilde{P}$, convexity imposes implicit restrictions on how far P can extend beyond the edges of $\tilde{P}$: Let a, b and c be vertices of convex polygon P appearing in that order and

---

[1] In three dimensions, the corresponding polyhedral hierarchy must be explicitly constructed in a preprocessing phase [DaK3].

consider the two chords ab and bc of P. Consider the two half-planes H^+_{ab} and H^-_{ab} (respectively, H^+_{bc} and H^-_{bc}) defined by ray ab (respectively, ray bc).

Observation 1: $P[a,b] \subset H^-_{ab}$ and $P[b,a] \subset H^+_{ab}$.

Observation 2: $P[c,a] \subset H^+_{ab} \cap H^+_{bc}$ and $H^-_{ab} \cap H^-_{bc} \cap P = \emptyset$.

Applying these observations about chords of P to the edges of an element of the polygon hierarchy $\tilde{P}$ implicitly defines $\hat{P}$: the star-shaped polygon (possibly unbounded) which is the union of all possible convex polygons which could have $\tilde{P}$ as a hierarchy element. The boundary of the polygon P must lie within the envelope of points between any hierarchy element $\tilde{P}$ and its corresponding $\hat{P}$ (see Figure 2.1).

The key to the efficient parallel use of the hierarchy element approximations is the judicious assignment of processors. Intuitively, this can be thought of as a direct analogue of parallel search in a sorted table [S]; each iteration of the k-way search corresponds to a step through the implicit polygon hierarchy. This correspondence, formalized by the reduction in Section 5, allows the design of k-processor convex polygon algorithms which are natural extensions of k-way search. Within each iteration of the natural k-way search algorithm for the sorted table lookup problem, the search key's containment in a particular subsequence can be determined by a single processor in constant time. This process is slightly more involved for the polygon tangents/separation problems but can be done efficiently by employing tests which exploit convexity and properties of a polygon hierarchy.

Within each iteration of the algorithms for polygon tangents and separation, a solution is found for the current approximations (hierarchy elements) of the polygons and then the appropriate case analysis is applied to reduce the region of interest. This can be thought of as 'electing' an active processor (the one that witnesses the hierarchy element/subsequence solution) which then applies the case reduction. The active processor then writes the reduced ranges into a predetermined location and the remaining processors use the concurrent read facility to update their information. Since only

one processor is assigned to the pair realizing the hierarchy element solution, this election process avoids write contention.

As an example of our general approach, consider the following k-processor algorithm for finding p^d, the extremum of an n-vertex convex polygon P in the direction of ray d: Initialize $\tilde{P}$ to be the last element of P's polygon hierarchy and repeat the following until $\tilde{P}$ is reduced to a single point. Assign a processor to each vertex of the current hierarchy element $\tilde{P}^t$. Vertex $\tilde{p}_i$ realizes the extremum of $\tilde{P}^t$ in the direction of ray d iff there exists a line L perpendicular to d such that $\tilde{p}_i \in L \cap \tilde{P}^t$, $\tilde{P}^t \subset H_L^+$ and d extends to infinity in H_L^-. This can be determined in constant time by comparing the slopes of segments $\tilde{p}_{i-1}\tilde{p}_i$ and $\tilde{p}_i\tilde{p}_{i+1}$ with ray d. If so, then by Observation 2, all points of P clockwise between $\tilde{p}_{i+1}$ and $\tilde{p}_{i-1}$ lie in the wedge formed by rays $\tilde{p}_i\tilde{p}_{i+1}$ and $\tilde{p}_i\tilde{p}_{i-1}$. Therefore, p^d must lie in $P[\tilde{p}_{i-1}, \tilde{p}_{i+1}]$. Replace $\tilde{P}$ with the portion of hierarchy element $\tilde{P}^{t-1}$ between vertices $\tilde{p}_{i-1}$ and $\tilde{p}_{i+1}$.

Each iteration of this procedure performs a constant number of tests for each element of the hierarchy. Essentially, this algorithm identifies the vertex of P which realizes the separation between P and a line L perpendicular to direction d placed at infinity in direction d. Therefore, we have immediately:

Lemma 2.1: An n-vertex convex polygon's extremum in a given direction d can be identified in $O(\log n/(1 + \log k))$ time using k CREW processors.

For the problems we consider, we present a uniform treatment and simplify aspects of our algorithms by adding an initial preprocessing step to reduce a convex polygon under consideration to a constant number of polygonal chains each of which delimits a convex region of the plane. From [DK1], we recall the definition of a Monotone Polygonal Sector (see Figure 2.2): The boundary of a convex polygon is decomposed into two monotone polygonal chains of edges by cutting at its highest and lowest y coordinates. This yields two sequences of vertices and edges in order of increasing y-coordinate. By convexity, any such chain will be either left- or right-oriented. Semi-

infinite rays are attached to the beginning and end of each chain and the interior is included to form two (convex) *monotone polygonal sectors* (MPS). These rays run parallel to the x-axis towards $+\infty$ if right-oriented or $-\infty$ if left-oriented and define the area contained by the MPS. By Lemma 2.1, a convex n-vertex polygon P can be decomposed into its left MPS P_L and its right MPS P_R in $O(\log n/(1 + \log k))$ time using k CREW processors.

We maintain MPS P_L with its vertices indexed by increasing y-coordinate as $P_L[p_{lower\_P}, p_{upper\_P}]$. This denotes the convex region delimited by the sequence of edges of P from vertex p_{lower_P} to vertex p_{upper_P} augmented by infinite rays at the upper and lower bounds. Initially, the upper and lower bounds correspond to the maximum and minimum y-coordinates respectively. Within the general design of our algorithms, each iteration reduces the size of a polygonal chain under consideration by adjusting the upper and lower bounds of an MPS. When the upper and lower bounds are adjusted, the resulting MPS has infinite rays extending from the adjusted bounds. The polygon hierarchy extends in the natural way to an MPS: the MPS $\tilde{P}^t_L[\tilde{p}_{lower\_P}, \tilde{p}_{upper\_P}]$ consists of the portion of hierarchy element $\tilde{P}^t$ between (and including) $\tilde{p}_{lower\_P}$ and $\tilde{p}_{upper\_P}$. When the context is clear, we omit the range specification and hierarchy element index by referring to $P_L[p_{lower\_P}, p_{upper\_P}]$ (respectively, $\tilde{P}_L[\tilde{p}_{lower\_P}, \tilde{p}_{upper\_P}]$) as P_L (respectively, $\tilde{P}_L$).

3.0 Polygon Separation

For our polygon separation algorithms, we assume that each input convex polygon P has been decomposed into its left and right MPSs P_L and P_R. We state without proof several useful separation properties of monotone polygonal sectors [DK1][1]:

MPS Property 1: Convex polygons P and Q intersect iff P_L intersects Q_R and P_R intersects Q_L.

---

[1] Note that a line segment can be considered a degenerate polygon and hence these properties can be applied to polygon/line segment separation.

MPS Property 2: If P and Q do not intersect then their separation is realized either by the separation between P_L and Q_R or the separation between Q_L and P_R.

MPS Property 3: Ignoring degeneracies, the boundaries of two oppositely-oriented MPSs can intersect in either 0 or 2 points.

First, consider the problem of finding the separation of an n-vertex convex polygon P and a line segment r. By MPS Properties 1 and 2, it is sufficient to describe an algorithm for MPS/line segment separation. For simplicity, we assume that the MPS and line segment do not intersect; the algorithm can be trivially modified to report an intersection if one is detected.

Without loss of generality, we consider the problem of determining the separation of MPS P_L and line segment r. The following k processor algorithm to find a point p* of P which realizes the separation of P_L and r is essentially the algorithm presented by Atallah and Goodrich [AG1]: Initialize $\tilde{P}_L$ to be the last element of P_L's polygon hierarchy and repeat the following. If $\tilde{P}_L^t$ is reduced to a constant number of segments then determine exhaustively a point p* of $\tilde{P}_L^t$ (not necessarily a vertex) which realizes the separation with r. Otherwise, assign a processor to each vertex $\tilde{p}_i$ of $\tilde{P}_L^t$. In constant time, determine if vertex $\tilde{p}_i$ is closer to segment r than vertices $\tilde{p}_{i-1}$ or $\tilde{p}_{i+1}$. If so, then by Observation 2, all points of P_L in $P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i-1}]$ or in $P_L[\tilde{p}_{i+1}, \tilde{p}_{upper\_P}]$ lie interior to the wedge formed by rays $\tilde{p}_i\tilde{p}_{i+1}$ and $\tilde{p}_i\tilde{p}_{i-1}$. Therefore, p* must lie in $P_L[\tilde{p}_{i-1}, \tilde{p}_{i+1}]$. Replace $\tilde{P}_L^t$ with the portion of hierarchy element $\tilde{P}_L^{t-1}$ between vertices $\tilde{p}_{i-1}$ and $\tilde{p}_{i+1}$.

Each iteration of this procedure takes a single step through the polygon hierarchy while maintaining the invariant that the remaining range of edges contains a point which realizes the desired separation. This algorithm can be used to find line/polygon separation by regarding the line as a segment with its endpoints extended to infinity. Similarly, this algorithm can be used to find point/polygon separation by regarding the point as a degenerate segment with zero length.[1] This last

---

[1] Note that due to the non-unimodality of distances in a convex polygon [E], this technique cannot be used directly for the farthest point problem.

application can be thought of as a table-lookup routine: finding the minimum in a virtual table of distances from a fixed point. We summarize these results as:

Lemma 3.1: The separation of an n-vertex convex polygon with a line-segment, a line or a point can be determined in O(log n/(1 + log k)) time using k CREW processors.

We turn to the problem of finding the separation of arbitrary convex polygons P and Q. If one of the polygons, say Q, has a constant number of facets then using Lemma 3.1 it is possible to determine the separation of P and Q by testing each facet of Q against P. If P is an n-vertex convex polygon, this would determine the polygon/polygon separation in O(log n/(1 + log k)) time using k CREW processors. However, if both P and Q are n-vertex polygons then this naive approach would not produce an optimal algorithm.

Therefore, consider the case in which both P and Q have n-vertices. The MPS properties simplify several aspects of determining polygon/polygon separation. By Properties 1 and 2, it suffices to compute MPS separations; without loss of generality, we describe an algorithm to determine the separation of P_L and Q_R. By Property 3, if the polygons P and Q do intersect, then their corresponding (oppositely-oriented) MPS boundaries have a constant number of intersections and and a witness to the intersection of P and Q can be determined from the corresponding MPS intersections.

Within each iteration of the algorithm, we maintain the invariant that the MPSs $\tilde{P}_L[\tilde{p}_{lower\_P}, \tilde{p}_{upper\_P}]$ and $\tilde{Q}_R[\tilde{q}_{lower\_Q}, \tilde{q}_{upper\_Q}]$ contain either a pair of points realizing the minimum distance separating P_L and Q_R or a point in the intersection of P_L and Q_R. Thus, the algorithm can be viewed as a search routine to identify p^* and q^*, points of P_L and Q_R respectively, which realize the MPS separation. Each iteration of the algorithm finds the separation of the current hierarchy elements $\tilde{P}_L$ and $\tilde{Q}_R$. If $\tilde{P}_L$ and $\tilde{Q}_R$ intersect then a witness to their intersection is returned as a witness to the intersection of P_L and Q_R. Alternatively, if $\tilde{P}_L$ and $\tilde{Q}_R$ do not intersect, their separation is realized by a vertex-vertex or vertex-segment pair.

We maintain the segment T with endpoints $\tilde{p}_i$ and $\tilde{q}_j$ defining the separation between hierarchy elements $\tilde{P}_L$ and $\tilde{Q}_R$. For ease of exposition in the vertex-segment case, we assume the existence of a pseudo-vertex on the segment in question which realizes the separation and assume that it has been labeled to incorporate it into the vertex ordering. This reduces the vertex-segment case to the vertex-vertex case.

Given $\tilde{P}_L$ and $\tilde{Q}_R$, hierarchy element MPSs of P_L and Q_R respectively, and the points $\tilde{p}_i$ and $\tilde{q}_j$ which realize the separation of $\tilde{P}_L$ and $\tilde{Q}_R$, the following two lemmas show how the range of vertices under consideration for realizing the separation of P_L and Q_R can be reduced so that a step can be taken in one of their polygon hierarchies. This is done by demonstrating that there exist portions of $\tilde{P}_L$ and $\tilde{Q}_R$ which can be replaced by horizontal rays such that points p* and q* realizing the separation are retained within the respective reduced MPSs.

Lemma 3.2 (The kitty-corner lemma)**:** Given hierarchy element MPSs $\tilde{P}_L$ and $\tilde{Q}_R$ and vertices $\tilde{p}_i$ and $\tilde{q}_j$ realizing their separation, then

(i) either $p^* \in P_L[\tilde{p}_{i-2}, \tilde{p}_{upper\_P}]$ or $q^* \in Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$

and (ii) either $p^* \in P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i+2}]$ or $q^* \in Q_R[\tilde{q}_{j-2}, \tilde{q}_{upper\_Q}]$.

Furthermore, the valid reduced range in each of (i) and (ii) can be determined by a single processor in constant time.

Proof: If one of the MPSs, say $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{upper\_Q}]$, is a single vertex[1] then, by Observation 2, $p^* \in P_L[\tilde{p}_{i-1}, \tilde{p}_{i+1}]$. As well, if $\tilde{p}_{i-2}$ or $\tilde{q}_{j+2}$ (respectively, $\tilde{p}_{i+2}$ or $\tilde{q}_{j-2}$) do not exist then (i) (respectively, (ii)) is true trivially.

Otherwise, each of statements (i) and (ii) offers a possibility of two MPS reductions. We ignore for the moment the question of determining which reduction is applicable, and begin by showing

---

[1] That is $\tilde{q}_{lower\_Q} = \tilde{q}_{upper\_Q}$.

that statements stronger than (i) and (ii) respectively are true. That is, we prove that the separation cannot be realized by a point in $P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i-1}]$ *and* a point in $Q_R[\tilde{q}_{j+1}, \tilde{q}_{upper\_Q}]$ (respectively, a point in $P_L[\tilde{p}_{i+1}, \tilde{p}_{upper\_P}]$ *and* a point in $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j-1}]$). Observe that the "kitty-corner" MPSs $P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i-1}]$ and $Q_R[\tilde{q}_{j+1}, \tilde{q}_{upper\_Q}]$ (respectively, $P_L[\tilde{p}_{i+1}, \tilde{p}_{upper\_P}]$ and $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j-1}]$) are contained in halfplanes on opposite sides of the parallel lines of support passing through $\tilde{p}_i$ and $\tilde{q}_j$ perpendicular to the separation segment $\tilde{p}_i\tilde{q}_j$. It follows that their minimum separation must be greater than the distance between $\tilde{p}_i$ and $\tilde{q}_j$ and hence (at least) one of them can be discarded while retaining a pair of points which realize the separation of P_L and Q_R.

The difficulty lies in the determination of *which* portions of P_L and Q_R can be discarded while retaining a pair of points which realize the true separation. To ensure that a correct reduced range can be determined by a single processor in constant time, it seems necessary to restrict consideration to which of $P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i-2}]$ and $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ (respectively, which of $P_L[\tilde{p}_{i+2}, \tilde{p}_{upper\_P}]$ and $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j-2}]$) can be replaced by a single ray.

We turn to the determination of a correct reduced range corresponding to statement (i). Consider the line $\tilde{p}_{i-1}\tilde{q}_{j+1}$. At least one of **(a)**, **(b)** and **(c)** must occur[1]:

(a) Vertex $\tilde{q}_{j+2} \in H^{+}_{\tilde{p}_{i-1}\tilde{q}_{j+1}}$ (see Figure 3.1). Note that the case condition implies that $\tilde{q}_{j+1} \in H^{-}_{\tilde{p}_{i-1}\tilde{q}_{j+2}}$ and equivalently, $\tilde{p}_{i-1} \in H^{+}_{\tilde{q}_{j+1}\tilde{q}_{j+2}}$. As well, by the monotonicity of Q_R, the y-coordinate of $\tilde{q}_{j+1}$ is less than the y-coordinate of $\tilde{q}_{j+2}$, therefore the case condition implies that the y-coordinate of $\tilde{p}_{i-1}$ is less than the y-coordinate of $\tilde{q}_{j+1}$.

Let w be the horizontal ray originating at $\tilde{q}_{j+2}$ and extending to positive infinity. Let v be the ray originating at $\tilde{q}_{j+2}$, at right angles to $\tilde{q}_{j+1}\tilde{q}_{j+2}$, such that v does not intersect the interior of Q_R. Denote by p' and q' points which realize the separation of P_L and $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ respectively. By Observation 1 and monotonicity, $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ is contained in the wedge

---

[1] It is possible that both **(a)** and **(b)** occur simultaneously.

$W_Q = H^+_{\tilde{q}_{j+1}\tilde{q}_{j+2}} \cap H^-_w$ and therefore $q' \in W_Q$. The following case analysis demonstrates that wherever p' is located, there must exist a point on w which realizes the separation of P_L and $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$:

1) $p' \in H^+_w \cup H^-_v$. Any circle centred at p' which contains a point in $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ must also contain a point of the ray w. That is, if $q' \notin w$ then it is possible to find a point $q_w \in w$ such that the angle $\angle p'q_wq' \geq \frac{\pi}{2}$ and therefore, the distance between p' and q_w must be less than the distance between p' and q'. Therefore, the separation of p' with $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ must be realized by a point on w and hence in $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$.

2) $p' \in H^+_v \cap H^-_w$. Since the y-coordinate of $\tilde{p}_{i-1}$ is less than the y-coordinate of $\tilde{q}_{j+1}$ then $\tilde{p}_{i-1} \notin H^+_v \cap H^-_w$. If $p' \in H^+_{\tilde{p}_{i-1}\tilde{q}_{j+2}}$ then the segment $\tilde{p}_{i-1}p'$ must intersect w and a witness to the intersection must exist in $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$. Otherwise, $p' \in H^-_{\tilde{p}_{i-1}\tilde{q}_{j+2}}$. Denote by p" the intersection of segment $\tilde{p}_{i-1}p'$ and ray v and note that by Case 1, $p' \neq p''$. Assume that $q' \notin w$. If the distance between p' and q' were less than the distance between p" and $\tilde{q}_{j+2}$, then ray p"p' would have to intersect ray $\tilde{q}_{j+2}q'$. But ray $p''p' \subset$ ray $\tilde{p}_{i-1}p' \subset H^-_{\tilde{p}_{i-1}\tilde{q}_{j+2}}$ and ray $\tilde{q}_{j+2}q' \subset W_Q$, and clearly $H^-_{\tilde{p}_{i-1}\tilde{q}_{j+2}}$ and W_Q are disjoint except for the vertex $\tilde{q}_{j+2}$. Therefore, the distance between p' and q' must be greater than the distance between p" and $\tilde{q}_{j+2}$ and the separation of P_L with $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ must be realized by a point in $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$.

These cases are exhaustive and therefore $q^* \in Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$. Note that the analysis does not depend on the structure of P_L or the location of $\tilde{p}_{i-1}$ other than what is implied by the initial case condition that $\tilde{q}_{j+2} \in H^+_{\tilde{p}_{i-1}\tilde{q}_{j+1}}$. This allows us to reuse this argument in the proof of Lemma 3.3.

(b) Vertex $\tilde{p}_{i-2} \in H^-_{\tilde{p}_{i-1}\tilde{q}_{j+1}}$. By symmetry with **(a)**, $p^* \in P_L[\tilde{p}_{i-2}, \tilde{p}_{upper\_P}]$.

(c) Vertex $\tilde{q}_{j+2} \in H^-_{\tilde{p}_{i-1}\tilde{q}_{j+1}}$ and vertex $\tilde{p}_{i-2} \in H^+_{\tilde{p}_{i-1}\tilde{q}_{j+1}}$ (see Figure 3.2). Denote line $\tilde{p}_{i-1}\tilde{p}_{i-2}$ by L and line $\tilde{q}_{j+1}\tilde{q}_{j+2}$ by R. Without loss of generality, assume that the intersection of L and R lies in $H^+_{\tilde{p}_i\tilde{q}_j}$ and therefore, that the horizontal separation of L and R in $H^-_{\tilde{p}_i\tilde{q}_j}$ increases with increasing

y-coordinate. By the case condition, $\tilde{q}_{j+2} \in H^-_{\tilde{p}_{i-1}\tilde{q}_{j+1}}$ (respectively, $\tilde{p}_{i-2} \in H^+_{\tilde{p}_{i-1}\tilde{q}_{j+1}}$), the intersection of L and R must lie below $\tilde{p}_{i-1}$ (respectively, below $\tilde{q}_{j+1}$).

Define rays w and v as in (**a**). Let u be the horizontal ray originating at $\tilde{p}_{i-1}$ and extending to negative infinity. Again, denote by p' and q' points which realize the separation of P_L and $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ respectively and, as in (**a**), note that $q' \in W_Q = H^+_{\tilde{q}_{j+1}\tilde{q}_{j+2}} \cap H^-_w$. Recall that the separation of P_L and Q_R cannot be realized by a point in $P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i-1}]$ *and* a point in $Q_R[\tilde{q}_{j+1}, \tilde{q}_{upper\_Q}]$ and therefore we assume that $p' \in P_L[\tilde{p}_{i-1}, \tilde{p}_{upper\_P}]$. By Observation 1 and monotonicity, $P_L[\tilde{p}_{i-1}, \tilde{p}_{upper\_P}]$ is contained in the wedge $W_P = H^-_{\tilde{p}_{i-2}\tilde{p}_{i-1}} \cap H^+_u$ and therefore $p' \in W_P$. Since the intersection of L and R lies below $\tilde{p}_{i-1}$ and $\tilde{q}_{j+1}$, the wedges W_P and W_Q are disjoint.

Let $\alpha = \angle\tilde{q}_{j+1}\tilde{q}_{j+2}\tilde{p}_{i-1}$ and $\beta = \angle\tilde{q}_j\tilde{q}_{j+1}\tilde{p}_{i-1}$. Before proceeding with a case analysis, we show that $\alpha < \frac{\pi}{2}$. Since $\tilde{q}_{j+2} \in H^+_{\tilde{q}_j\tilde{q}_{j+1}}$ (by Observation 1) and $\tilde{q}_{j+2} \in H^-_{\tilde{p}_{i-1}\tilde{q}_{j+1}}$ (by the case condition), therefore $\alpha < \beta$. We assume that $\alpha \geq \frac{\pi}{2}$ (and, therefore $\beta > \frac{\pi}{2}$) and derive a contradiction. If $\beta > \frac{\pi}{2}$, then $\tilde{p}_i \in H^+_{\tilde{p}_{i-1}\tilde{q}_{j+1}}$ (since $\angle\tilde{p}_i\tilde{q}_j\tilde{q}_{j+1} \geq \frac{\pi}{2}$). Therefore, $\tilde{p}_{i-2}$ must be below the horizontal line through $\tilde{p}_{i-1}$ (by monotonicity), $\tilde{p}_{i-2} \in H^+_{\tilde{p}_i\tilde{p}_{i-1}}$ (by Observation 1), and $\tilde{p}_{i-2} \in H^+_{\tilde{p}_{i-1}\tilde{q}_{j+1}}$ (by the case condition). But these three regions are disjoint (except for vertex $\tilde{p}_{i-1}$). Therefore, $\beta \leq \frac{\pi}{2}$ and hence $\alpha < \frac{\pi}{2}$ and $\tilde{p}_{i-1} \in H^-_v$.

The following case analysis demonstrates that wherever p' is located there must exist a point on w which realizes the separation of P_L and $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$:

1) $p' \in H^+_{\tilde{q}_{j+1}\tilde{q}_{j+2}} \cap H^-_w$ ($= W_Q$). As noted above, $p' \in W_P$ and $W_P \cap W_Q = \varnothing$. Therefore this case cannot occur.

2) $p' \in H^+_w \cup H^-_v$. As in (**a**) Case 1, any circle centred at p' which contains a point in $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ must also contain a point of the ray w. Therefore, the separation of p' with $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ must be realized by a point in $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$.

3) $p' \in H_v^+ \cap H^-_{\tilde{q}_{j+1}\tilde{q}_{j+2}}$. Since $\tilde{p}_{i-1} \in H_v^-$, p' and $\tilde{p}_{i-1}$ lie on opposite sides of ray v. Denote by p'' the intersection of segment $\tilde{p}_{i-1}p'$ and ray v and note that by Case 1, $p' \neq p''$. Assume that $q' \notin w$. If the distance between p' and q' were less than the distance between p'' and $\tilde{q}_{j+2}$, then ray $p''p'$ would have to intersect ray $\tilde{q}_{j+2}q'$. But ray $p''p' \subset$ ray $\tilde{p}_{i-1}p' \subset W_P$ and ray $\tilde{q}_{j+2}q' \subset W_Q$, and $W_P \cap W_Q = \emptyset$. Therefore, the distance between p' and q' must be greater than the distance between p'' and $\tilde{q}_{j+2}$ and the separation of P_L with $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ must be realized by a point in $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$.

These cases are exhaustive and therefore $q^* \in Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$. Similarly, if the intersection of L and R lies in $H^-_{\tilde{p}_i\tilde{q}_j}$ then $p^* \in P_L[\tilde{p}_{i-2}, \tilde{p}_{upper\_P}]$. If L and R are parallel, then $q^* \in Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$ and $p^* \in P_L[\tilde{p}_{i-2}, \tilde{p}_{upper\_P}]$.

The arguments used in **(a)**, **(b)** and **(c)** prove (i); using symmetry in the y direction the same arguments prove (ii). The tests employed above to determine correct reduced ranges are line intersection and point containments in half-planes. These tests can be easily implemented to run in constant time using a single processor. ♦

To take a step in one of the polygon hierarchies within a given iteration, it is necessary to ensure that one of the $\tilde{P}_L$ and $\tilde{Q}_R$ subsequences under consideration be reduced to a constant length. This is not guaranteed by Lemma 3.2. It is possible, for example, that only the upper portion of $\tilde{P}_L$ and the upper portion of $\tilde{Q}_R$ are eliminated. The following Lemma shows that this reduction is always possible for at least one of the hierarchy element approximations.

Lemma 3.3 (The same side lemma): Given hierarchy element MPSs $\tilde{P}_L$ and $\tilde{Q}_R$ and vertices $\tilde{p}_i$ and $\tilde{q}_j$ realizing their separation, then

(i) either $p^* \in P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i+2}]$ or $q^* \in Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$

and (ii) either $p^* \in P_L[\tilde{p}_{i-2}, \tilde{p}_{upper\_P}]$ or $q^* \in Q_R[\tilde{q}_{j-2}, \tilde{q}_{upper\_Q}]$.

Furthermore, the valid reduced range in each of (i) and (ii) can be determined by a single processor in constant time.

Proof: If one of the MPSs, say $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{upper\_Q}]$, is a single vertex then, by Observation 2, $p^* \in P_L[\tilde{p}_{i-1}, \tilde{p}_{i+1}]$. As well, if $\tilde{p}_{i+2}$ or $\tilde{q}_{j+2}$ (respectively, $\tilde{p}_{i-2}$ or $\tilde{q}_{j-2}$) do not exist then (i) (respectively, (ii)) is true trivially.

Otherwise, as in Lemma 3.2, we ignore for the moment the question of determining which reduction is applicable, and begin by showing that statements stronger than (i) and (ii) respectively are true. Observe that the "same side" MPSs $P_L[\tilde{p}_{i+1}, \tilde{p}_{upper\_P}]$ and $Q_R[\tilde{q}_{j+1}, \tilde{q}_{upper\_Q}]$ (respectively, $P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i-1}]$ and $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j-1}]$) are contained in halfplanes on opposite sides of the parallel lines of support passing through $\tilde{p}_i$ and $\tilde{q}_j$ perpendicular to the separation segment $\tilde{p}_i\tilde{q}_j$. It follows that their minimum separation must be greater than or equal to the distance between $\tilde{p}_i$ and $\tilde{q}_j$ and that (at least) one of them can be discarded while retaining a pair of points which realize the separation of P_L and Q_R.

Again, it seems necessary to restrict consideration to which of $P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i-2}]$ and $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j-2}]$ (respectively, which of $P_L[\tilde{p}_{i+2}, \tilde{p}_{upper\_P}]$ and $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$) can be discarded to ensure that a correct reduced range can be determined by a single processor in constant time.

We turn to the determination of a correct reduced range corresponding to statement (i). Consider line $\tilde{p}_{i+1}\tilde{q}_{j+1}$. At least one of **(a)**, **(b)** and **(c)** must occur[1]:

(a) Vertex $\tilde{q}_{j+2} \in H^+_{\tilde{p}_{i+1}\tilde{q}_{j+1}}$. Substituting $\tilde{p}_{i+1}$ for $\tilde{p}_{i-1}$, the same case analysis used in Lemma 3.2 **(a)** shows that $q^* \in Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$.

(b) Vertex $\tilde{p}_{i+2} \in H^+_{\tilde{p}_{i+1}\tilde{q}_{j+1}}$. By symmetry with **(a)**, $p^* \in P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i+2}]$.

(c) Vertex $\tilde{q}_{j+2} \in H^-_{\tilde{p}_{i+1}\tilde{q}_{j+1}}$ and vertex $\tilde{p}_{i+2} \in H^-_{\tilde{p}_{i+1}\tilde{q}_{j+1}}$ (see Figure 3.3). Define rays w and v as in Lemma 3.2 **(a)**.

---

[1] It is possible that both **(a)** and **(b)** occur simultaneously.

Consider the angles $\alpha = \angle\tilde{p}_{i+1}\tilde{p}_{i+2}\tilde{q}_{j+2}$ and $\beta = \angle\tilde{q}_{j+1}\tilde{q}_{j+2}\tilde{p}_{i+2}$. Note that $\alpha + \beta < \pi$ (otherwise, rays $\tilde{p}_{i+1}\tilde{p}_{i+2}$ and $\tilde{q}_{j+1}\tilde{q}_{j+2}$ would intersect or would be parallel) and therefore at least one of the angles α or β must be less than $\frac{\pi}{2}$. Without loss of generality, assume that $\beta < \frac{\pi}{2}$ and therefore that $\tilde{p}_{i+2} \in H_v^-$.

Again, denote by p' and q' points which realize the separation of P_L and $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ respectively and, as in Lemma 3.2 **(a)**, note that $q' \in W_Q = H^+_{\tilde{q}_{j+1}\tilde{q}_{j+2}} \cap H_w^-$. We demonstrate by a case analysis that wherever p' is located there must exist a point on w which realizes the separation of P_L and $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$:

1) $p' \in H^-_{\tilde{q}_{j+1}\tilde{q}_{j+2}} \cap H_w^-$ $(= W_Q)$. In the halfplane $H^-_{\tilde{p}_{i+1}\tilde{q}_{j+2}}$, $P_L \subset H^-_{\tilde{p}_i\tilde{p}_{i+1}}$ (by Observation 1), $W_Q \subset H^+_{\tilde{q}_j\tilde{q}_{j+1}}$ (by Observation 1), and rays $\tilde{p}_i\tilde{p}_{i+1}$ and $\tilde{q}_j\tilde{q}_{j+1}$ do not intersect. In the halfplane $H^+_{\tilde{p}_{i+1}\tilde{q}_{j+2}}$, $P_L \subset H^+_{\tilde{p}_{i+2}\tilde{p}_{i+1}}$ (by Observation 1), $W_Q \subset H_w^-$ and rays $\tilde{p}_{i+2}\tilde{p}_{i+1}$ and w do not intersect. Therefore, P_L and W_Q are disjoint and this case cannot occur.

2) $p' \in H_w^+ \cup H_v^-$. As in Lemma 3.2 **(a)** Case 1, any circle centred at p' which contains a point in $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ must also contain a point of the ray w. Therefore, the separation of p' with $Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ must be realized by a point in $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$.

3) $p' \in H_v^+ \cap H^-_{\tilde{q}_{j+1}\tilde{q}_{j+2}}$. Since $\beta < \frac{\pi}{2}$, then $\tilde{p}_{i+2} \in H_v^-$ and $p' \in P_L[\tilde{p}_{i+2}, \tilde{p}_{upper\_P}]$. But, as noted above, the separation of P_L and Q_R cannot be realized by a point in $P_L[\tilde{p}_{i+1}, \tilde{p}_{upper\_P}]$ *and* a point in $Q_R[\tilde{q}_{j+1}, \tilde{q}_{upper\_Q}]$. Therefore, no q' arising in this case can be a point realizing the minimum separation of P_L and Q_R.

These cases are exhaustive and therefore $q^* \in Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{j+2}]$. Similarly, if $\alpha < \frac{\pi}{2}$ then $p^* \in P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i+2}]$.

The arguments used in **(a)**, **(b)** and **(c)** prove (i); using symmetry in the y direction the same arguments prove (ii). The tests employed above to determine the valid reduced ranges are point containments in half-planes and angle comparisons. These tests can be implemented to run in constant time using a single processor. ♦

Without loss of generality, assume that $k \geq 25$. We outline a k CREW processor algorithm for determining the separation between P_L and Q_R which follows from Lemmas 3.2 and 3.3 above: Initialize $\tilde{P}_L$ and $\tilde{Q}_R$ to be the last elements of the $\sqrt{k}$-polygon hierarchies[1] of P_L and Q_R respectively and repeat the following. If $\tilde{P}_L^t$ or $\tilde{Q}_R^s$ are of constant size then use the polygon/segment separation algorithm to determine their separation and halt. Otherwise, assign a processor to each pairing of a vertex from $\tilde{P}_L^t$ with a vertex from $\tilde{Q}_R^s$. The processor which determines that the edges incident on its pair of vertices realize the separation of the current hierarchy elements then restricts the sequence of vertices under consideration according to Lemmas 3.2 and 3.3 by adjusting lower_P and upper_P, or lower_Q and upper_Q. Depending on which MPS bounds are adjusted, replace $\tilde{P}_L^t$ (respectively, $\tilde{Q}_R^s$) with the portion of $\tilde{P}_L^{t-1}$ (respectively, $\tilde{Q}_R^{s-1}$) between vertices $\tilde{p}_{lower\_P}$ and $\tilde{p}_{upper\_P}$ (respectively, $\tilde{q}_{lower\_Q}$ and $\tilde{q}_{upper\_Q}$).

Theorem 3.1: The separation of two convex n-vertex polygons P and Q can be determined in $O(\log n/(1 + \log k))$ time using k CREW processors.

Proof: Given two convex n-vertex polygons, the MPS preprocessing is performed in $O(\log n/(1 + \log k))$ time using k CREW processors. Within each iteration, the vertices realizing the separation of $\tilde{P}_L$ and $\tilde{Q}_R$ (or $\tilde{P}_R$ and $\tilde{Q}_L$) can be determined in constant time. Therefore by Lemmas 3.2 and 3.3, each iteration takes constant time and results in a step in (at least) one of the polygon hierarchies. Therefore, a total of $O(\log n/(1 + \log k))$ iterations will suffice to determine MPS separation. Running the algorithm twice for the two MPS pairs will suffice to determine the convex polygon separation. ♦

The use of the $\sqrt{k}$-polygon hierarchy (an instance of the $\sqrt{k}$-subdivision technique [KR]) provides a logarithmic sequential algorithm and a constant time algorithm using a quasi-linear

---

[1] Note that an integer within a constant factor of $\sqrt{k}$ will suffice for the subdivision size and that with k processors an integral approximation of $\sqrt{k}$ can be computed in constant time. Each processor squares its predecessor's, its successor's and its own index. The unique processor whose squared index is closest to k, writes its index into a predetermined location and the other processors read it using the concurrent read facility.

number of processors for computing convex polygon separation. The polygon/polygon separation case analysis provided in Edelsbrunner [E] produces a sequential O(log n) time separation algorithm. If concurrent write is available, it is possible to use Edelsbrunner's case analysis to design a polygon/polygon separation algorithm which runs in O(log n/(1 + log k)) time using k CRCW processors. In the absence of a concurrent write facility, it is not straightforward to coordinate the $\sqrt{k}$ case results at each vertex of the hierarchy elements under consideration. Our algorithm avoids this coordination problem by "electing" the processor which corresponds to the hierarchy element minimum separation to apply the case analysis.

3.0 Polygon Separating and Common Tangents

For the problem of computing convex polygon tangents, we add a preprocessing step. To ensure that the polygons in question do not intersect, we apply the appropriate separation algorithm presented in the last section. With a pair of points realizing the separation, we construct a separating line S perpendicular to the segment realizing the separation. Without loss of generality, we assume that this separating line S is oriented horizontally with polygon P above and polygon Q below. Given this orientation, it is sufficient to consider MPS common/separating tangents: The two separating tangents of P and Q can be constructed from oppositely-oriented MPSs P_L and Q_R, or P_R and Q_L. The two common tangents can be constructed from similarly-oriented MPSs P_R and Q_R or P_L and Q_L.

Therefore, we assume that each polygon P has been preprocessed into MPSs P_L and P_R which are monotone with respect to the perpendicular of the separating line S as constructed above. Note that the rays augmenting the left and right monotone polygonal chains of P are parallel to S.

We address the problem of finding separating tangents. For a convex polygon and vertex pair, the separating tangents are the same as the common tangents. For two convex polygons, minor modifications to the separating tangents algorithm produce an algorithm for finding common tangents; we discuss the necessary modifications at the end of the section.

We first consider the problem of constructing the tangent of an n-vertex MPS P_L passing through a vertex r outside P_L. We assume that P_L and r are presented with a horizontally-oriented separating line S with P_L above S and r below S. The following k processor algorithm to find a point p* of P which realizes the tangent of P_L passing through r is essentially the algorithm presented by Atallah and Goodrich [AG1]: Initialize $\tilde{P}_L$ to be the last element of P_L's polygon hierarchy and repeat the following. If $\tilde{P}_L^t$ is reduced to a constant number of segments then determine exhaustively a vertex p* of $\tilde{P}_L^t$ which realizes the tangent passing through r. Otherwise, assign a processor to each vertex $\tilde{p}_i$ of $\tilde{P}_L^t$. In constant time, for vertex $\tilde{p}_i$ determine if $\tilde{p}_{i-1}$ and $\tilde{p}_{i+1}$, the neighbours of $\tilde{p}_i$ on $\tilde{P}_L^t$, both lie on the same side of line $r\tilde{p}_i$. If so, then by Observation 2, all points of P_L in $P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i-1}]$ or in $P_L[\tilde{p}_{i+1}, \tilde{p}_{upper\_P}]$ lie interior to the wedge formed by rays $\tilde{p}_i\tilde{p}_{i+1}$ and $\tilde{p}_i\tilde{p}_{i-1}$. Therefore, p* must lie in $P_L[\tilde{p}_{i-1}, \tilde{p}_{i+1}]$. Set $\tilde{p}_{lower\_P}$ to $\tilde{p}_{i-1}$ and $\tilde{p}_{upper\_P}$ to $\tilde{p}_{i+1}$ and replace $\tilde{P}_L^t$ with the portion of hierarchy element $\tilde{P}_L^{t-1}$ between vertices $\tilde{p}_{i-1}$ and $\tilde{p}_{i+1}$.

Each iteration of this procedure takes a single step through the polygon hierarchy while maintaining the invariant that the range of edges under consideration contains a point which realizes the desired tangent. Therefore, the tangent of P_L passing through r can be constructed using k CREW processors in $O(\log n/(1 + \log k))$ time. The separation and MPS preprocessing are performed within the same resource bounds.

Now, consider the problem of constructing the separating tangents of n-vertex convex polygons P and Q. As noted above, it suffices to construct the separating tangents of oppositely-oriented MPSs; without loss of generality, we describe an algorithm to construct the separating tangent of P_L and Q_R. We assume that P_L and Q_R have been preprocessed with respect to the horizontally-oriented separating line S as described above and that they are presented with P_L above S and Q_R below S. Since this preprocessing step uses the separating line perpendicular to the minimum separation of P and Q, the vertex with minimum y-coordinate on P_L and the vertex with the maximum y-coordinate on Q_R have the same x-coordinate. Therefore, we are searching for the separating tangent of maximum positive slope.

Denote the vertices of P_L and Q_R which realize the separating tangent as p^* and q^* respectively; recall that the vertices are indexed in order of increasing y-coordinate. Given $\tilde{P}_L$ and $\tilde{Q}_R$, hierarchy element MPSs of P_L and Q_R respectively, and the current separating tangent line T defined by vertices $\tilde{p}_i$ and $\tilde{q}_j$ of $\tilde{P}_L$ and $\tilde{Q}_R$ respectively, the following two lemmas show how the range of vertices under consideration for realizing the separating tangent of P_L and Q_R can be reduced so that a step can be taken in one of their polygon hierarchies. This is done by demonstrating that there exist portions of $\tilde{P}_L$ and $\tilde{Q}_R$ which can be replaced by horizontal rays such that points p* and q* realizing the separating tangent are retained within the respective reduced MPSs.

Lemma 4.1 (See Figure 4.1): Given hierarchy element MPSs$\tilde{P}_L$ and $\tilde{Q}_R$ with vertices $\tilde{p}_i$ and $\tilde{q}_j$ realizing their separating tangent T then $p^* \in P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i+1}]$ and $q^* \in Q_R[\tilde{q}_{j-1}, \tilde{q}_{upper\_Q}]$.

Proof: If one of the MPSs, say $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{upper\_Q}]$, is a single vertex then, by Observation 2, $p^* \in P_L[\tilde{p}_{i-1}, \tilde{p}_{i+1}]$.

Otherwise, $p^* \in P_L[\tilde{p}_{i+1}, \tilde{p}_{upper\_P}]$ iff a vertex in that range lies in $H^+_{\tilde{q}_j\tilde{p}_i}$. By Observation 1, all vertices in $P_L[\tilde{p}_{i+1}, \tilde{p}_{upper\_P}]$ lie in $H^-_{\tilde{p}_i\tilde{p}_{i+1}}$ and hence in $H^-_{\tilde{q}_j\tilde{p}_i}$. Therefore, $p^* \in P_L[\tilde{p}_{lower\_P}, \tilde{p}_{i+1}]$ and, by a symmetric argument, $q^* \in Q_R[\tilde{q}_{j-1}, \tilde{q}_{upper\_Q}]$. ♦

Lemma 4.2 (See Figure 4.1): Given hierarchy element MPSs $\tilde{P}_L$ and $\tilde{Q}_R$ with vertices $\tilde{p}_i$ and $\tilde{q}_j$ realizing their separating tangent T then either $p^* \in P_L[\tilde{p}_{i-2}, \tilde{p}_{i+1}]$ or $q^* \in Q_R[\tilde{q}_{j-1}, \tilde{q}_{j+2}]$. Furthermore, the correct containment can be determined in constant time using a single processor.

Proof: If one of the MPSs, say $Q_R[\tilde{q}_{lower\_Q}, \tilde{q}_{upper\_Q}]$, is a single vertex then, by Observation 2, $p^* \in P_L[\tilde{p}_{i-1}, \tilde{p}_{i+1}]$. As well, if $\tilde{p}_{i-2}$ or $\tilde{q}_{j+2}$ do not exist then the result follows trivially from Lemma 4.1.

Otherwise, denote line $\tilde{p}_{i-2}\tilde{p}_{i-1}$ by L, and line $\tilde{q}_{j+1}\tilde{q}_{j+2}$ by R. Note that the slopes of L and R are both positive and both less than the slope of T. Without loss of generality, assume that L and R intersect below S and denote line $\tilde{p}_i\tilde{p}_{i-1}$ by L_2. Note that L_2 must also intersect R below S since the slope of L_2 lies between the slopes of L and T. If $q^* \in Q_R[\tilde{q}_{j+2}, \tilde{q}_{upper\_Q}]$ then part of P_L above S

must extend below R. By Observation 1, all vertices on P_L which are outside $P_L[\tilde{p}_{i-2}, \tilde{p}_{i-1}]$ (respectively, outside $P_L[\tilde{p}_{i-1}, \tilde{p}_i]$) are above L (respectively, above L_2). Therefore, no part of P_L below T can extend below R and therefore, with the above argument, $q^* \in Q_R[\tilde{q}_{j-1}, \tilde{q}_{j+2}]$[1]. Similarly, if L intersects R above S then $p^* \in P_L[\tilde{p}_{i-2}, \tilde{p}_{i+1}]$.

This test can be easily implemented to run in constant time using a single processor. ♦

The polygon/polygon separating tangent algorithm uses the $\sqrt{k}$-subdivision technique (in the $\sqrt{k}$-polygon hierarchy) in the same way as the polygon separation algorithm described in Section 3. Within each iteration the separating tangent between the current hierarchy elements is determined and the case analysis described in the above lemmas enables the algorithm to take a step in (at least) one of the polygon hierarchies.

Theorem 4.1: The separating tangents of two convex n-vertex polygons P and Q can be determined in $O(\log n/(1 + \log k))$ time using k CREW processors.

Proof: The separation preprocessing and decomposition into monotone polygonal sectors is performed within the same resource bounds. Similar to Theorem 3.1, by Lemmas 4.1 and 4.2, each iteration takes constant time and results in a step through one of the polygon hierarchies. Therefore, a total of $O(\log n/(1 + \log k))$ iterations will suffice to determine MPS separating tangents. Running the algorithm twice with the appropriate (symmetric) changes will produce both separating tangents. ♦

The case analysis presented here is similar in spirit to that provided by Overmars and Van Leeuwen [OvL]. They describe an analysis for a sequential O(log n) time algorithm to find the upper common tangents of two n-vertex convex polygons; their analysis can be modified to provide an equivalent separating tangent result. Although the analysis for their common tangent algorithm is sufficient for the sequential result, it only considers consecutive vertices and thus does not

---

[1] Note that if line $\tilde{q}_j\tilde{q}_{j+1}$ intersects L above S then it is possible that $q^* \in [\tilde{q}_{j+1}, \tilde{q}_{j+2}]$.

immediately provide a test which will enable the algorithm to take a step in one of the polygon hierarchies.

The separating tangents algorithm can be modified easily to compute common tangents within the same bounds. For the separating tangent case analysis presented in Lemma 4.2, the case reduction is predicated on whether R and L intersect above or below the separating line S. In fact, within Lemma 4.2, the same case reduction could have been predicated on whether R and L intersect to the left or to the right of the current separating tangent T. However, the analog of Lemma 4.2 used in the common tangent algorithm requires a separating line for its case reduction (as does the case reduction of Overmars and Van Leeuwen [OvL]). Therefore, as for the separating tangents algorithm, given two convex polygons P and Q, we apply the appropriate separation algorithm, construct a line S between them perpendicular to their separation and decompose P and Q into their respective MPSs with respect to line S .

We describe the modifications to find the common tangent of P_L and Q_L (see Figure 4.2); the common tangent algorithm for P_R and Q_R is symmetric. The $\sqrt{k}$-hierarchies technique are used for each of P_L and Q_L. Within each iteration, the tangent of $\tilde{P}_L$ and $\tilde{Q}_L$ is found and analyses analogous to those in Lemmas 4.1 and 4.2 are applied to take a step in one of $\tilde{P}_L$'s or $\tilde{Q}_L$'s polygon hierarchies.

Denote the vertices of P_L and Q_L which realize the common tangent as p^* and q^* respectively. Assume that we have vertices$\tilde{p}_i$ and $\tilde{q}_j$ which realize the common tangent T of $\tilde{P}_L[\tilde{p}_{lower\_P}, \tilde{p}_{upper\_P}]$ and $\tilde{Q}_L[\tilde{q}_{lower\_Q}, \tilde{q}_{upper\_Q}]$, hierarchy elements of P_L and Q_L respectively.

For the analog of Lemma 4.1, consider the horizontal slab delimited by the y-coordinate of $\tilde{p}_i$ above and the y-coordinate of $\tilde{q}_j$ below. The true common tangent support vertices p^* and q^* must lie on or to the right of where the current common tangent segment $\tilde{p}_i\tilde{q}_j$ intersects this slab. Note that $\tilde{p}_{i-1}$ and $\tilde{q}_{j+1}$ lie to the left of the current common tangent. By Observation 1, all vertices on P_L below $\tilde{p}_{i-1}$ lie to the left of line $\tilde{p}_{i-1}\tilde{p}_i$ and therefore lie to the left of the current common tangent. Therefore, p^* cannot be below $\tilde{p}_{i-1}$ on P_L. Similarly, q^* cannot be above $\tilde{q}_{j+1}$ on Q_L.

For the analog of Lemma 4.2, denote ray $\tilde{p}_{i+2}\tilde{p}_{i+1}$ by L, and ray $\tilde{q}_{j-2}\tilde{q}_{j-1}$ by R. Denote ray $\tilde{p}_{i+1}\tilde{p}_i$ by L_2 and ray $\tilde{q}_{j-1}\tilde{q}_j$ by R_2. Without loss of generality, assume that L and R intersect below the given separator S. Note that L_2 must also intersect R below S since L_2 lies to the left of L. For q^* to exist below $\tilde{q}_{j-2}$ on Q_L, part of P_L must extend to the right of R. By Observation 1, all vertices on P_L which are outside $P_L[\tilde{p}_{i+2}, \tilde{p}_{i+1}]$ (respectively, outside $P_L[\tilde{p}_{i+1}, \tilde{p}_i]$) are to the left of L (respectively, to the left of L_2) and above S. Therefore, no part of P can extend to the right of R and therefore, combined with the above reduction $q^* \in Q_L[\tilde{q}_{j-2}, \tilde{q}_{j+1}]$[1]. Similarly, if L intersects R above the given separator S then $p^* \in P_L[\tilde{p}_{i-1}, \tilde{p}_{i+2}]$.

Therefore, using the $\sqrt{k}$-polygon hierarchies and the case analysis above, we have immediately:

Theorem 4.2: The common tangents of two convex n-vertex polygons P and Q can be determined in $O(\log n/(1 + \log k))$ time using k CREW processors.

This technique provides a logarithmic sequential algorithm and a constant time algorithm using a quasi-linear number of processors for computing upper common tangents. With n processors, using standard recursive subdivision, the common tangents algorithm can be incorporated into another optimal 2-d convex hull algorithm. Earlier optimal 2-d convex hull algorithms used a $\sqrt{k}$-subdivision for the divide and conquer step [ACGOY] [AG2]; this can be thought of as using the power of this technique within a different level of the algorithm. This $\sqrt{k}$-subdivision technique has also been used for list merging and other applications [KR]. Interestingly, Cole & Goodrich [CG] use a cascading routine based on Cole's parallel merge sort [C] to construct the convex hull without using the $\sqrt{k}$-subdivision technique.

---

[1] Note that if R_2 intersects L and L_2 above S then it is possible that $q^* \in [\tilde{q}_{j-1}, \tilde{q}_{j-2}]$.

5.0 Lower Bounds

We have repeatedly stressed the link between the common tangents/separation problems and sorted table lookup. We make this precise with a reduction.

Recall that the sorted table look-up problem is defined as follows: given a sorted table of n real numbers $x_1, x_2, \ldots x_n$ and a search key x_s, return the index i such that $x_s \in [x_i, x_{i+1}]$ (with $x_0 = -\infty$ and $x_{n+1} = \infty$). To deal simply with the extreme table values, we consider the normalized sorted table look-up problem where $x_1 = 0$, $x_n = 1$, and $x_s \in (0, 1)$.

Theorem 5.1: The sorted table-lookup problem can be reduced to convex polygon common tangents and separation.

Proof: Given a sorted table $X = \{x_1, x_2, \ldots x_n\}$ such that $x_1 = 0$ and $x_n = 1$, and a search key $x_s \in (0, 1)$, associate with each x_i the coordinate $P(x_i) = (x_i, -x_i^2)$. This associates with each table entry a vertex on a parabola opening downwards. Therefore, X defines the vertices of a convex polygon P(X). Consider the key x_s; $P(x_s)$ is also a vertex on this parabola. It is easy to see that,

(i) $x_s \in [x_k, x_{k+1}]$ iff $P(x_s)$ has common tangents with $P(x_k)$ and $P(x_{k+1})$; and

(ii) $x_s \in [x_k, x_{k+1}]$ iff the separation of $P(x_s)$ with P(X) is realized by a point on the segment $P(x_k)P(x_{k+1})$.

Therefore, a solution to either of these problems provides an equivalent solution to the sorted table-lookup problem. ♦

Snir [S] has shown that in the CREW model of computation, with k processors available, the sorted table look-up problem has a time bound of $\Theta(\log n/(1 + \log k))$. In the absence of concurrent read, the sorted table look-up problem has time bound $\Theta(1+ \log n - \log k)$. (This is a key result which demonstrates that processors with a concurrent read facility are strictly more powerful than processors without concurrent reads.) This result together with Theorems 3.1, 4.1 and 4.2 immediately yields:

Corollary 5.1: Finding the common or separating tangents of a vertex with an n-vertex convex polygon with k CREW processors has time complexity $\Theta(\log n/(1+\log k))$.

Corollary 5.2: Finding the separation of a vertex with an n-vertex convex polygon with k CREW processors has time complexity $\Theta(\log n/(1+\log k))$.

The algorithms presented in Sections 3 and 4 demonstrate that the tight bounds stated in Corollaries 5.1 and 5.2 for the vertex/polygon problems in the CREW model extend to the polygon/polygon versions of those problems[1]. The reduction of Theorem 5.1 has implications for the EREW model as well:

Corollary 5.3: Finding the common or separating tangents of a vertex with an n-vertex convex polygon with k EREW processors has time complexity $\Theta(1 + \log n - \log k)$.

Corollary 5.4: Finding the separation of a vertex with an n-vertex convex polygon with k EREW processors has time complexity $\Theta(1 + \log n - \log k)$.

The algorithms providing the upper bound for Corollaries 5.3 and 5.4 are the obvious extensions of the EREW k-way search algorithm: the k processors are used in the first step to reduce the length of the input sequence by a factor of k and the (unique) winning processor whose subsequence contains the solution applies the sequential algorithm to obtain the final solution.

However, it is not clear how to design an $O(1 + \log n - \log k)$ algorithm for the polygon/polygon versions of those problems using the EREW model of computation. In the CREW model, the $\sqrt{k}$-subdivision technique introduces a constant factor which does not affect the asymptotic complexity of the solution. This technique does not seem to be useful in the EREW model. In fact, given two n-vertex polygons, it is not clear how to obtain an asymptotic speedup for n^{α} EREW processors for any $\alpha \leq 1$.

---

[1] Note that within the vertex-polygon version of the algorithms described a full concurrent read facility is not necessary, a facility which allows one processor to broadcast to all other processors would suffice.

6.0 Discussion

Employing a hierarchical representation of convex polygons, we have demonstrated tight time bounds of $\Theta(\log n/(1+\log k))$ for the separation and separating/common tangents problems using the CREW model of computation. The algorithms and lower bounds have been derived by regarding the exterior of a convex polygon as being related to a sorted table and extending table lookup results to the separation and tangent problems.

As reported elsewhere [DaK3], it is possible to augment the subdivision hierarchies of Kirkpatrick [K] [DaK1] [DaK2] to produce a linear space data structure which will support optimal cooperative algorithms for planar subdivision point location. By regarding the exterior of a convex polyhedron as topologically equivalent to a planar subdivision, it is possible to use these point location algorithms to design efficient cooperative algorithms for convex polyhedron separation.

References

[ACGOY] Aggarwal, A., Chazelle, B., Guibas, L., O'Dunlaing, C., and Yap, C., "Parallel computational geometry", *Algorithmica* 3, 1988, pp. 293-327.

[AES] Avis, D., El Gindy, H. and Seidel, R., "Simple on-line algorithms for convex polygons", in Computational Geometry, G.T. Toussaint (editor), North-Holland, 1985.

[AG1] Atallah, M.J. and Goodrich, M.T., "Parallel algorithms for some functions of two convex polygons", *Algorithmica* 3, 1988, pp. 535-548.

[AG2] Atallah, M.J. and Goodrich, M.T., "Efficient parallel solutions to some geometric problems", *Journal of Parallel and Distributed Computing*, 1986, pp. 492-507.

[C] Cole, R., "Parallel merge sort", *Proc. 27th Annual IEEE Symp. on Foundations of Comp. Sci.*, 1986, pp. 511-516.

[CG] Cole, R. and Goodrich, M., "Optimal parallel algorithms for polygon and point-set problems (preliminary version)", *Proc. 20th ACM Symp. on Theory of Computing*, 1988, pp. 201-210.

[DaK1] Dadoun, N. and Kirkpatrick, D.G. "Parallel processing for efficient subdivision search", *Proc. of the 3rd ACM Symposium on Computational Geometry*, 1987, pp. 204-214.

[DaK2] Dadoun, N. and Kirkpatrick, D.G., "Parallel construction of subdivision hierarchies", University of British Columbia Computer Science Dept. Technical Report, TR 87-15, 1987; also To Appear in *Journal of Parallel and Distributed Computing*.

[DaK3] Dadoun, N. and Kirkpatrick, D.G., "Cooperative algorithms for planar point location and convex polyhedron separation", In preparation, 1989.

[DaK4] Dadoun, N. and Kirkpatrick, D.G., "Cooperative subdivision search algorithms with applications", *Proc. 27th Allerton Conference on Communication, Control and Computing*, 1989.

[DK1] Dobkin, D.P. and Kirkpatrick, D.G., "Fast detection of polyhedral intersections", *Proceedings of the International Colloquium on Automata, Languages and Programming*, 1982, pp. 154-165.

[DK2] Dobkin, D.P. and Kirkpatrick, D.G., "Fast detection of polyhedral intersection", *Theoretical Computer Science* 27, 1983, pp. 241-253.

[DK3] Dobkin, D.P. and Kirkpatrick, D.G., "A linear time algorithm for determining the separation of convex polyhedra", *Journal of Algorithms* 6, 3, 1985, pp. 381-392.

[DK4] Dobkin, D.P. and Kirkpatrick, D.G., "Determining the separation of preprocessed polyhedral - A unified approach", manuscript.

[E] Edelsbrunner, H., "Computing the extreme distances between two convex polygons", *Journal of Algorithms* 6, 1985, pp. 213-224.

[K] Kirkpatrick, D.G., "Optimal search in planar subdivisions", *SIAM Journal of Computing* 12,1, January 1983, pp. 28-35.

[KR] Karp, R.M., and Ramachandran, V. "A survey of parallel algorithms for shared-memory machines", Report No. UCB/CSD 88/408, Comp. Sci. Division, UC Berkeley, March, 1988.

[OvL] Overmars, M.H., and van Leeuwen, J., "Maintenance of configurations in the plane", *Journal of Computer and Systems Sciences* 23, 1981, pp. 166-204.

[S] Snir, M., "On parallel searching", *SIAM Journal of Computing* 14, 3, August 1985, pp. 688-708.

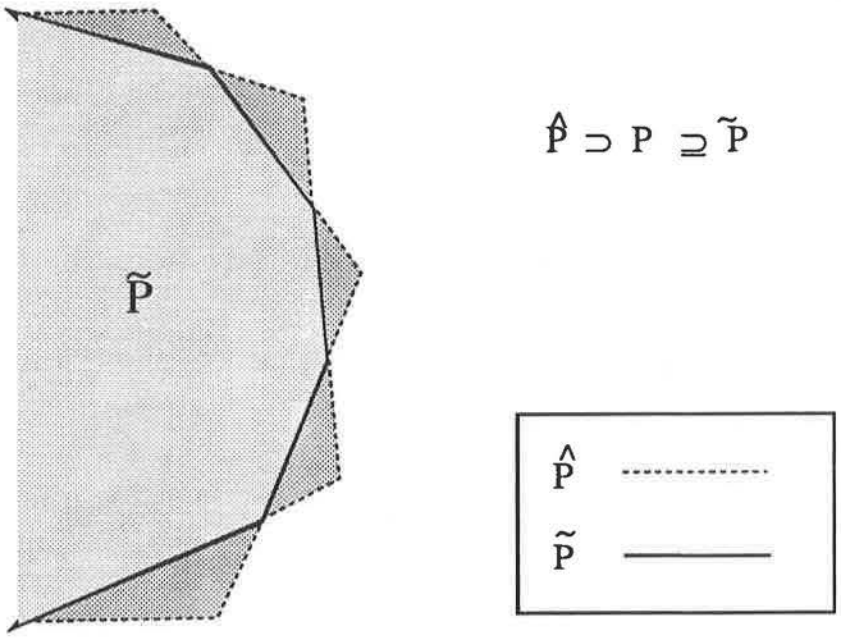

Figure 2.1: The envelope between $\hat{P}$ and $\tilde{P}$ containing the boundary of P

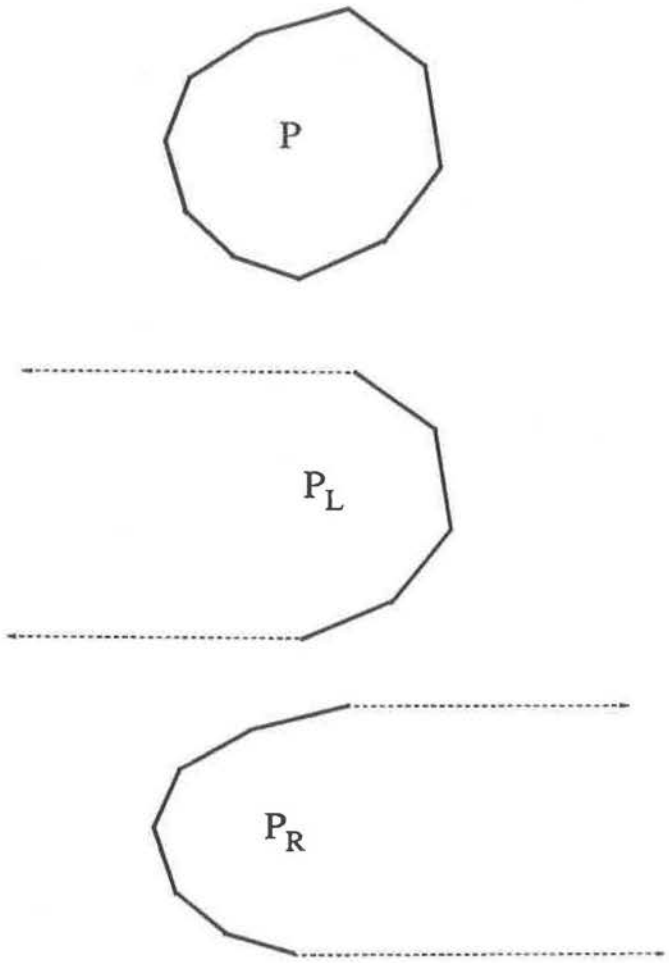

Figure 2.2: Monotone Polygonal Sectors

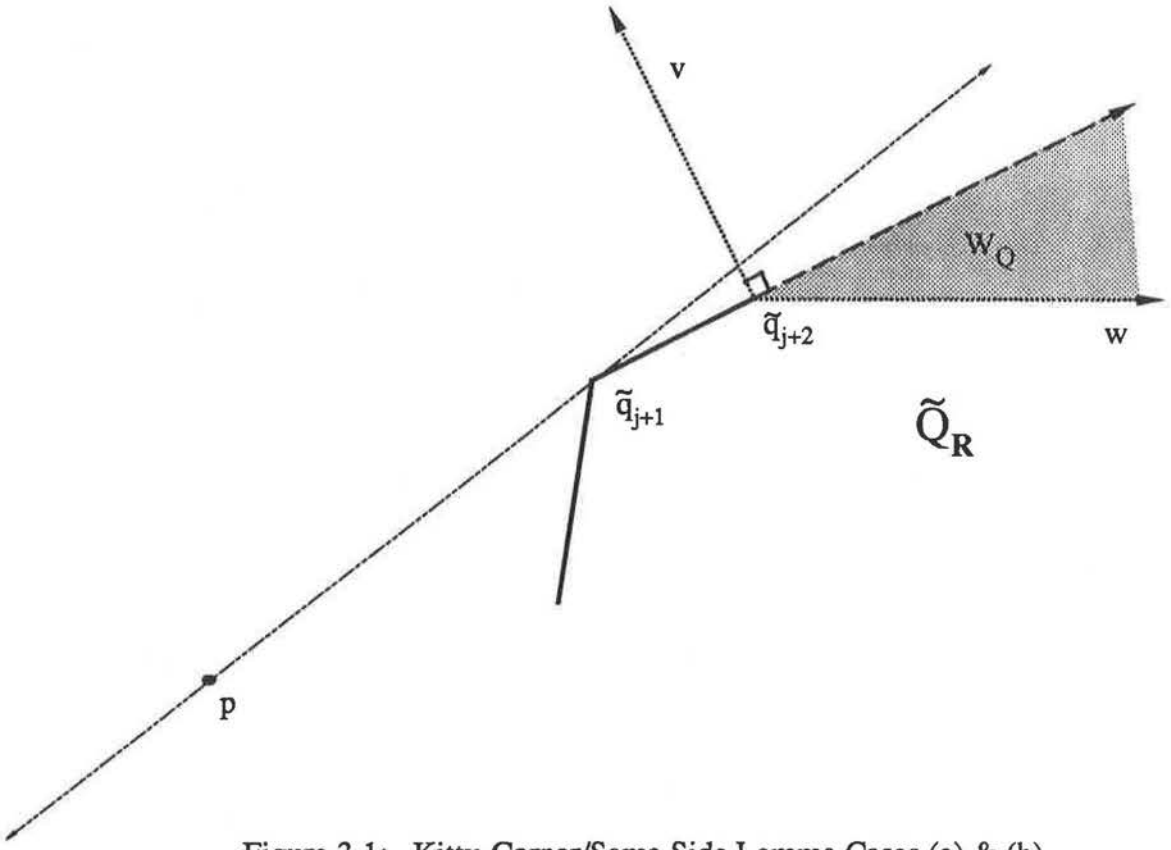

Figure 3.1: Kitty-Corner/Same Side Lemma Cases (a) & (b)

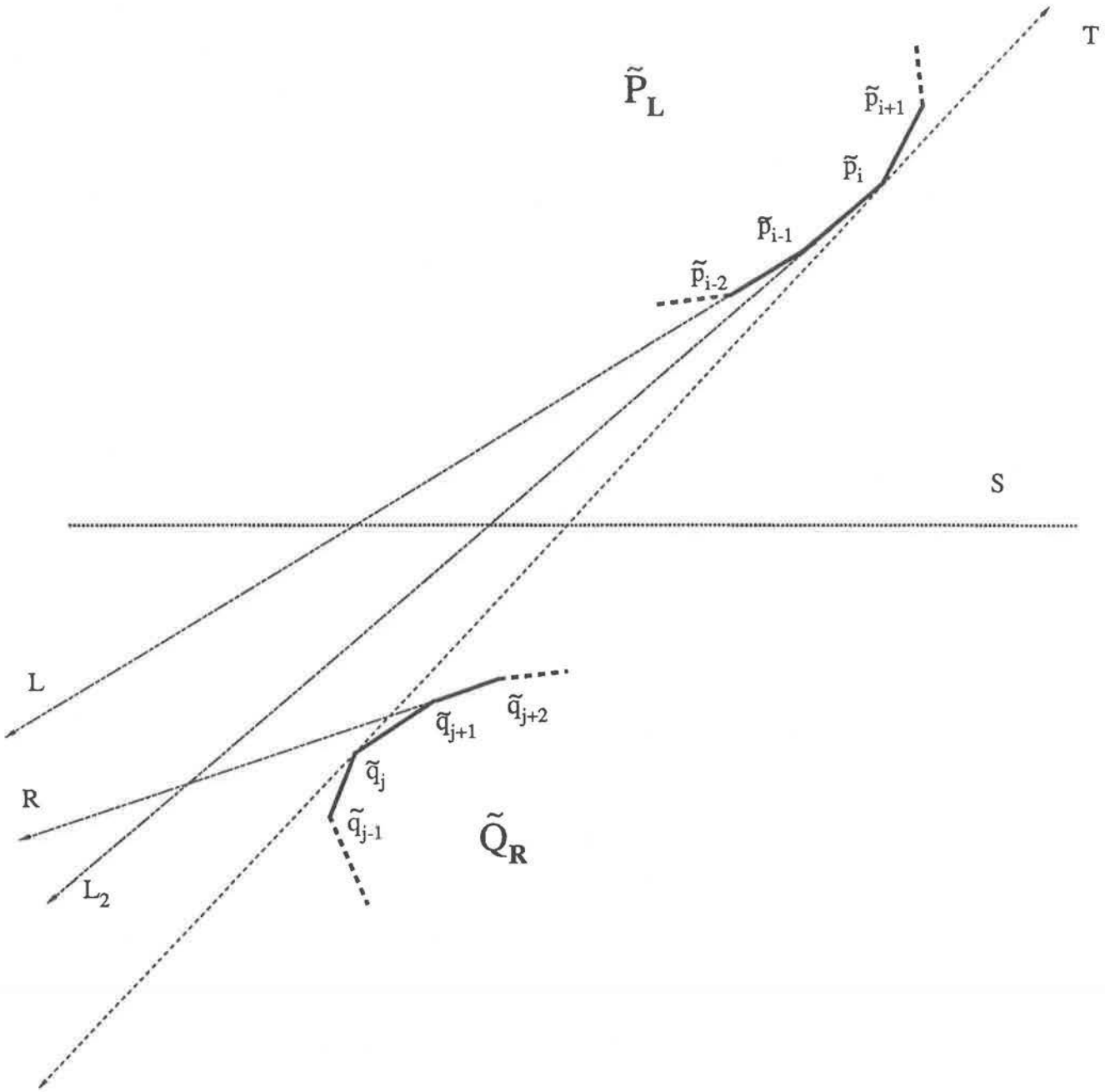

Figure 4.1: Separating Tangent Diagram

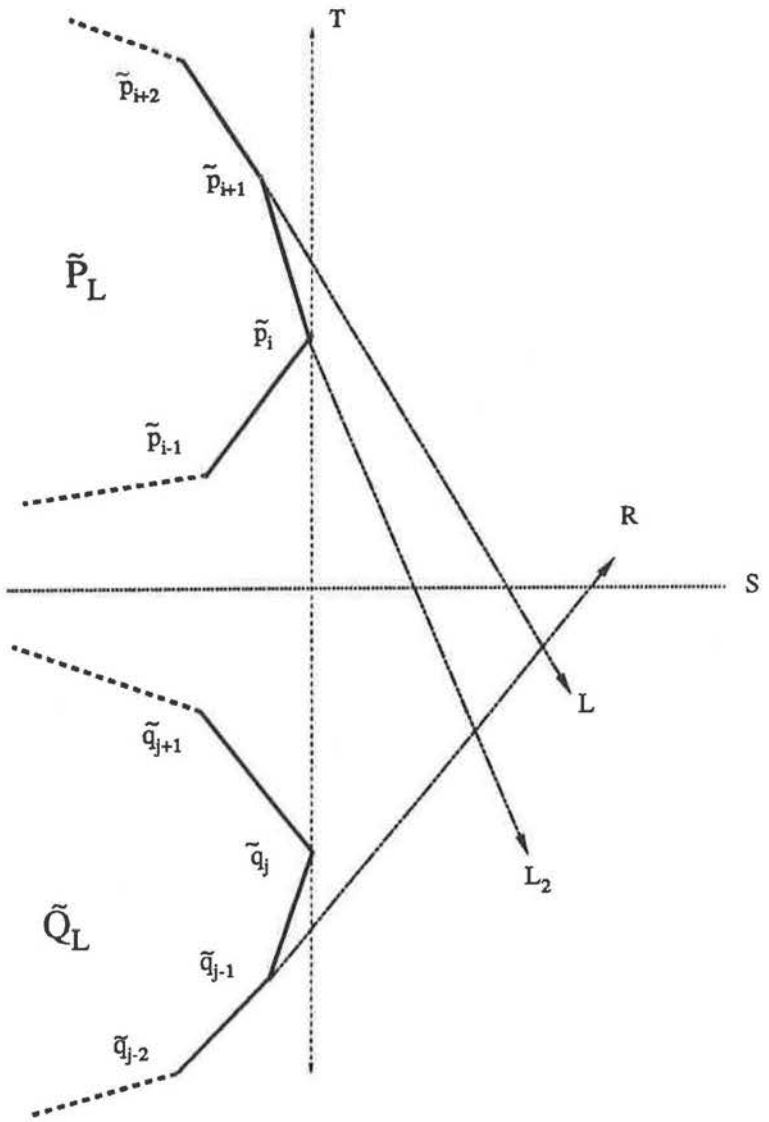

Figure 4.2: Common Tangent Diagram