THE PARALLEL 3D CONVEX-HULL PROBLEM REVISITED—REVISION 1

N. Amato
F. P. Preparata

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

Approved for Public Release. Distribution Unlimited.
In this paper we prove the correctness of a "local" criterion for computing the convex hull of the union ("merging") of two disjoint convex polyhedra. This criterion is structural. Therefore it can be algorithmically tested in several ways, not necessarily involving the determination of support (tangent) planes; indeed, it can be implemented by just testing for the intersection of certain planes and lines with convex polytopes. This criterion is amenable to parallel implementation and leads to a provably correct algorithm that computes the convex hull of any point set in three-dimensional space in $O(\log^2 n)$ time using $O(n)$ CREW PRAM processors.
The Parallel 3D Convex-Hull Problem Revisited - 
Revision 1*

N. Amato and F. P. Preparata
University of Illinois at Urbana-Champaign

Abstract

In this paper we prove the correctness of a "local" criterion for computing the convex hull of the union ("merging") of two disjoint convex polyhedra. This criterion is structural. Therefore it can be algorithmically tested in several ways, not necessarily involving the determination of support (tangent) planes; indeed, it can be implemented by just testing for the intersection of certain planes and lines with convex polytopes. This criterion is amenable to parallel implementation and leads to a provably correct algorithm that computes the convex hull of any point set in three-dimensional space in $O(\log^2 n)$ time using $O(n)$ CREW PRAM processors.

1 Introduction

Computing the convex hull of a set of $n$ points in three-dimensional space is a fundamental problem in Computational Geometry, which has received considerable attention since the early days of this discipline. About ten years ago technological innovations prompted some interest in parallel algorithms for this problem, and A. Chow [5] proposed the first algorithm in this class. Other ingenious parallel convex-hull algorithms have been proposed since, notable among them the CREW PRAM algorithms by Aggarwal et al. [1, 3]. Dadoun and Kirkpatrick [6] made use of simple data structures (the hierarchical representations of Dobkin and Kirkpatrick [8, 9, 10] also adopted in this paper) to improve the efficiency of the techniques of Aggarwal et al.. Lu [16] proposed a mesh connected computer (MCC) algorithm for computing the convex-hull in the case in which the point set is restricted to the surface of a sphere. A MCC algorithm for arbitrary point sets, which avoids some shortcomings of [1, 3, 5, 6], is given by Dehne et al. in [7].

*This work was supported in part by NSF Grant CCR89-06419
The previous algorithms for the three-dimensional convex hull problem are based on the serial divide-and-conquer algorithm of Preparata and Hong [17], whose crucial operation is the merging of the convex hulls of two linearly separated point sets. With the exception of the algorithm of Chow [5], all of the above algorithms use similar criteria for merging separable convex hulls $H_1$ and $H_2$. Namely, for every edge $e$ of $H_1$, the tangent planes through $e$ to $H_2$ are used to determine if the edge $e$ is part of the convex hull of $H_1 \cup H_2$. Furthermore, the criterion in the above algorithms [1, 3, 6, 5, 7], which each edge $e$ and its respective tangent plane(s) must meet in order for $e$ to be classified as an edge of $CH(H_1 \cup H_2)$, is not necessary and sufficient for all problem instances. (Although, in all cases a relatively simple addition to the stated criterion would do the trick.)

It is the purpose of this paper to evidence and rectify the inaccuracies in the referenced algorithms [1, 3, 5, 6, 7] and to present a self-contained 3D parallel convex-hull algorithm. We develop a criterion for an edge $e$ of $H_1$ to belong to the convex hull of $H_1 \cup H_2$, whose verification does not necessarily resort to the computation of the tangent planes through $e$ to $H_2$, but instead can be achieved by the potentially simpler operation of testing for intersection of $CH(H_2)$ with appropriate lines and planes. The main contribution of this paper is therefore the statement of a structural criterion for merging disjoint convex hulls $H_1$ and $H_2$, yielding a provably correct algorithm for computing the convex hull of a three-dimensional point set in $O(\log^2 n)$ time using $O(n)$ processors on a CREW PRAM. It must be emphasized, however, that many of the techniques implementing the introduced "merge criterion" are either derived from or inspired by ideas appearing in the papers quoted above [8, 9, 10], which should receive all the appropriate credit.

2 A Convex-Hull Merge Criterion

The algorithms of Chow [5], Aggarwal et al. [1, 3], Lu [16], and Dehne et al. [7], are all inspired by the serial divide-and-conquer algorithm of Preparata and Hong [17]. Let $CH(X)$ denote the convex hull of the point set $X$. The serial algorithm for computing the convex hull of a point set $S$ can be outlined as follows: the set $S$ is evenly divided into two sets $P$ and $Q$ such that the $z$-value of each vertex in $P$ is greater than the $z$-value of every vertex in $Q$; $CH(P)$ and $CH(Q)$ are recursively computed; the cycle of supporting faces that are tangent to $CH(P)$ and $CH(Q)$ is computed; finally, $CH(P)$ and $CH(Q)$ are merged along the cycle of supporting faces just computed to form $CH(P \cup Q)$.

Note that the edges of $CH(P)$ ($CH(Q)$) that are incident to the cycle of supporting faces of $CH(P)$ and $CH(Q)$ will form an Eulerian circuit which will be referred to as the upper seam (lower seam) as in [1, 3], in which vertices may be visited more than once and the same edge may occur with both orientations along the seam. The criteria used to identify the seam edges in [1, 5] do not take into account the fact that the same edge may occur with
both orientations along the seam. It is correctly noted in [3] that the crucial operation is the efficient computation of the seam edges at each stage:

*Once the edges in the seam have been calculated, their cyclic connection order can be ascertained in logarithmic time by a list-ranking process. The full structure of the sleeve can be deduced once both seams have been constructed by implementing what is essentially a merging process, easily accomplished in logarithmic time.*

The following criterion is used in [3] to determine which edges belong to the upper seam (an analogous criterion is used for the lower seam).

*Consider a fixed edge of CH(P); let L be the line containing it. If the edge belongs to the upper seam it is necessary (a) that L should not meet the interior of CH(Q), and (b) one (or conceivably both) of the two planes through L tangent to CH(Q) should not intersect the interior of CH(P).*

It is not difficult to verify that the above criterion is necessary, although it is not sufficient, as illustrated in Figure 1. Here edge $\overline{ab}$ is clearly not a seam edge, although it satisfies the above criterion for inclusion in the seam: The line containing $\overline{ab}$ does not intersect CH(Q) and the plane tangent to CH(Q) passing through $\overline{ab}$, defined by the points $a$, $b$ and $c$, does not intersect $P$.

Unlike the above algorithms [3, 1, 6], which are concerned with determining the seam edges, Dehne et al. [7] are concerned with identifying all edges of CH(P) that are edges of CH(P U Q). Let $f_r(e)$ and $f_l(e)$ be the two faces of CH(P) incident to edge $e$. Let $T_l$ and $T_r$ be the two tangent planes from $e$ to CH(Q), and let $v_l$ and $v_r$ be the vertices of CH(Q) supporting $T_l$ and $T_r$, respectively. The edge $e$ and the vertices $v_l$ and $v_r$, respectively, form potential faces $f_l$, and $f_r$, of CH(P U Q). The condition used in [7] to classify an edge $e$ of CH(P) as an edge of CH(P U Q) is as follows: $e$ is an edge of CH(P U Q) if and only if $f_r$ (resp. $f_l$), $f_r(e)$, and $f_l(e)$ are contained in the same halfspace defined by $T_r$ (resp. $T_l$). It is not difficult to see that all seam edges will be correctly identified by this criterion; however,
edges that are incident only to faces of $CH(P)$ that are also faces of $CH(P \cup Q)$ will not be identified (such edges occur when both $f_r$ and $f_l$ intersect the interior of $CH(P)$).

Considering without loss of generality the upper seam, we now present a structural necessary and sufficient condition for the classification of an edge $e$ of $CH(P)$ as a seam edge. For a point set $U$, $\text{interior}(U)$ denotes the set of points in the interior of $CH(U)$. For an edge $e$ of $CH(P)$, the planes containing the two faces of $CH(P)$ incident to $e$ partition the space into four quadrants. (See Figure 2, where $CH(P)$ is projected onto a plane orthogonal to $e$.) Specifically, edge $e$ is given an arbitrary orientation, which permits us to identify as $T_l(e)$ and $T_r(e)$ the planes respectively containing faces $f_l(e)$ and $f_r(e)$ of $CH(P)$ incident to $e$. The quadrant containing $P$ is referred to as $P(e)$; the two quadrants adjacent to $P(e)$ are referred to as $L(e)$ and $R(e)$, separated from $P(e)$ by $T_l(e)$ and $T_r(e)$ respectively; finally, the remaining quadrant, vertically opposed to $P(e)$, is called $\text{wedge}(e)$.

Definition: A point $p$ is $P$-visible from a point $q$ if the segment $pq$ does not intersect $CH(P)$. An edge $e$ is $P$-visible from a point $q$ if every point of $e$ is $P$-visible from $q$. (Similarly, a face $f$ is $P$-visible from a point $q$ if every point of $f$ is $P$-visible from $q$.)

The following lemma characterizes the edges of $CH(P)$ belonging to the upper seam of $CH(P \cup Q)$.

Lemma 1: An edge $e$ of $CH(P)$ belongs to the upper seam of $CH(P \cup Q)$ if and only if
(i) $Q \nsubseteq P(e)$, and
(ii) $\text{interior}(Q) \cap \text{wedge}(e) = \emptyset$.

Proof: If an edge $e$ of $CH(P)$ belongs to $CH(P \cup Q)$ but is not a seam edge we will say that $e$ "lies above" the seam. If an edge $e$ of $CH(P)$ is not part of $CH(P \cup Q)$ we will say that $e$ "lies below" the seam. It is immediate to verify that (i) an edge $e$ of $CH(P)$ lies above the seam if and only if $e$ is not $P$-visible from any point $q$ of $CH(Q)$ and (ii) an edge $e$ of
Figure 3: Edge \( e \) of \( CH(P) \) is not part of the upper seam if \( Q \subset P(e) \) or \( \text{interior}(Q) \cap \text{wedge}(e) \neq \emptyset \), (a) and (b), respectively, otherwise \( e \) is part of the upper seam, (c) and (d).

\( CH(P) \) lies below the seam if and only if there exists a point \( q \) of \( Q \) such that both \( f_l(e) \) and \( f_r(e) \) are \( P \)-visible from \( q \). If neither of the above conditions is met then \( e \) is a seam edge.

We now note that \( e \) is not \( P \)-visible from exactly the points which lie in \( P(e) - CH(P) \) and both \( f_l(e) \) and \( f_r(e) \) are \( P \)-visible from exactly the points which lie in the interior of \( \text{wedge}(e) \). Thus, if \( Q \subset P(e) \) or \( \text{interior}(Q) \cap \text{wedge}(e) \neq \emptyset \) we can establish that \( e \) is not part of the seam (see Figure 3, cases (a) and (b), in which \( P \) and \( Q \) are projected onto a plane orthogonal to \( e \)). In all other cases, \( e \) is part of the seam (see Figure 3, cases (c) and (d)); thus, negating the statement "\( Q \subset P(e) \) or \( \text{interior}(Q) \cap \text{wedge}(e) \neq \emptyset \)" yields the theorem. □

3 The Algorithm

As a consequence of Lemma 1, we see that a technique that tests the conditions \( Q \not\subset P(e) \) and \( \text{interior}(Q) \cap \text{wedge}(e) = \emptyset \) for each edge \( e \) of \( CH(P) \) could be used to determine the upper seam edges of \( CH(P \cup Q) \). We will next develop such a technique.

We first illustrate that the above conditions can be tested by executing a particular combination of the following three types of primitives.

**Plane_query**\((T, Q)\): Let \( T \) be a plane and \( Q \) a convex polyhedron. A **plane_query**\((T, Q)\) will return a point of \( T \cap \text{interior}(Q) \) if an intersection occurs, otherwise it will return
the empty set.

**Line_query**(*l*, *Q*): Let *l* be a line and *Q* a convex polyhedron. A line_query(*l*, *Q*) will return "YES" if *l* intersects interior(*Q*) and "NO" otherwise.

**Halfspace_query**(*p*, *T*): Let *T* be a plane and *p* a point not contained in *T*. A halfspace_query(*p*, *T*) will determine in which of the halfspaces defined by *T* the point *p* lies.

We begin with Condition (i): *Q* ∉ *P*(*e*). Clearly, if either plane_query(*T*(_i_)*e*, *Q*) or plane_query(*T*(_r_)*e*, *Q*) find a point of intersection, then *Q* ∉ *P*(*e*). Otherwise, if neither plane_query(*T*(_i_)*e*, *Q*) nor plane_query(*T*(_r_)*e*, *Q*) find a point of intersection then *Q* is contained in exactly one of the four quadrants *P*(*e*), *L*(*e*), *R*(*e*) or wedge(*e*). Let *q* be any point of interior(*Q*) (such a point can be determined in O(1) time from any four vertices of CH(*Q*)). Then halfspace_query(*q*, *T*(_i_)*e*) and halfspace_query(*q*, *T*(_r_)*e*) will determine which quadrant contains *Q*, and consequently whether *Q* ∈ *P*(*e*).

Next we consider Condition (ii): interior(*Q*) ∩ wedge(*e*) = ∅. Let *l*(*e*) be the line containing *e*. Clearly, if line_query(*l*(**e**), *Q*) returns YES, then interior(*Q*) ∩ wedge(*e*) ≠ ∅ and we are done. Otherwise, we perform plane_query(*T*(_l_)*e*, *Q*) and plane_query(*T*(_r_)*e*, *Q*). If neither plane_query(*T*(_i_)*e*, *Q*) nor plane_query(*T*(_r_)*e*, *Q*) find a point of intersection then *Q* is contained in exactly one of the four quadrants *P*(*e*), *L*(*e*), *R*(*e*) or wedge(*e*), and the quadrant containing *Q* can be determined by the same halfspace_queries described above for testing the condition *Q* ∉ *P*(*e*). The remaining case to consider is when plane_query(*T*(_i_)*e*, *Q*) and/or plane_query(*T*(_r_)*e*, *Q*) find a point of intersection. In this case we use the points of intersection returned by the queries to determine if interior(*Q*) ∩ wedge(*e*) = ∅. The line containing *e* divides *T*(_i_)*e* into two half-planes; we will denote the half-plane of *T*(_i_)*e* which bounds wedge(*e*) as T(^w_)*i*(e) and the other as T(^f_)*i*(e) (T(^w_)*r*(e) and T(^f_)*r*(e) are defined analogously). Note that it is not possible for *Q* to intersect either (i) both T(^w_)*i*(e) and T(^f_)*i*(e), or (ii) both T(^w_)*r*(e) and T(^f_)*r*(e), because the initial test, i.e. line_query(*l*(**e**), *Q*), established that *l*(**e**) does not intersect interior(*Q*). Thus, halfspace_query(*p*(, *T*(_l_)*e*)) and halfspace_query(*p*(, *T*(_r_)*e*)) will determine if interior(*Q*) ∩ wedge(*e*) ≠ ∅, where *p*( and *p*( are the points of intersection returned by plane_query(*T*(_i_)*e*, *Q*) and plane_query(*T*(_r_)*e*, *Q*), respectively.

Thus, we see that we can determine if an edge *e* ∈ CH(*P*) belongs to the seam by answering line, plane and halfspace queries. Whereas the technique of [1, 3, 6, 7] required the computation of the tangent planes from *e* to CH(*Q*), as we have seen, it is enough to perform the potentially simpler detection of the intersection of CH(*Q*) with certain lines and planes.

We will next show how each of the above queries can be implemented in parallel. Clearly the halfspace queries can be answered in O(1) time using a single CREW PRAM processor. The techniques used to answer line and plane queries are patterned after analogous more
general techniques sketched in [8, 9, 10]; their adaptation to the present situation, however, considerably simplifies the implementation.

The data structure for \( CH(Q) \) we will use is the very versatile hierarchical representation of Dobkin and Kirkpatrick [8, 9, 10]. Loosely following the notation used in [10], the hierarchical representation (HR) of a convex polyhedron (and, analogously, of a convex polygon) can be described as a sequence of polytopes \( P_1, P_2, ..., P_k \) with the following properties. Let \( V(P) \) denote the vertex set of the given polytope \( P \). An independent set of vertices in \( P \) is a subset of \( V(P) \) no two elements of which are joined by an edge. In three dimensions, polyhedron \( P \) is assumed to be triangulated.

1. \( P_1 = P \) and \( P_k \) is a tetrahedron.
2. \( P_{i+1} \subseteq P_i \), for \( 1 \leq i < k \).
3. \( V(P_{i+1}) \subseteq V(P_i) \), for \( 1 \leq i < k \).
4. The vertices of \( V(P_i) - V(P_{i+1}) \) form an independent set in \( P_i \), for \( 1 \leq i < k \).
5. Each facet \( f \) of \( P_{i+1} \) that is not a facet of \( P_i \) has associated with it a pointer to the unique vertex of \( P_i \) that lies in the halfspace not containing \( P_{i+1} \), with respect to the plane containing \( f \). (The fact that there is a unique such vertex follows from Property 4.)

The HR of a convex polytope \( P \) with a triangulated surface can be constructed by a process reminiscent of the preprocessing of a planar triangulation occurring in Kirkpatrick’s planar point-location-technique [14], in which a maximal independent set of vertices, each of degree less than 7, is removed from \( P_i \) to form \( P_{i+1} \). In fact, the HR of \( P \) can be constructed by applying this same process to the planar subdivision obtained as a stereographic projection of the surface of polyhedron \( P \) on a plane. A remarkable feature of the approach is that the cardinality of the maximal independent set of vertices, each of degree less than some fixed integer \( \mu \), removed at each stage will be large enough to ensure that \( k = O(\log n) \) and \( \sum_{i=1}^{k} |V(P_i)| = O(|V(P)|) \). Kirkpatrick [14] selected \( \mu = 12 \); Lipton and Miller [15] (and independently Edahiro et al. [11]) showed that \( \mu = 7 \) is equally applicable.

Intersection is preserved through projection, so that plane/polyhedron and line/polyhedron intersections can be tested on their projections onto a plane orthogonal to the given plane or line, respectively. We let \( P^{(H)} \) denote the orthogonal projection (a polygon) of polyhedron \( P \) onto an arbitrary plane \( H \). If we can easily construct \( P^{(H)} \) – where \( H \) is now the plane orthogonal to the query line or plane – then line_queries and plane_queries become much simpler two-dimensional problems. In reality, \( P^{(H)} \) need not be explicitly constructed in its entirety; all that is needed is the portion of \( P^{(H)} \) relevant to the intersection detection. An important property of the hierarchical representation of \( P \) is that it enables us to efficiently
construct the "relevant portion" of $P^H$; in other words, an HR of $P$ implicitly contains $P^H$ for an arbitrary plane $H$.

Specifically, let $P_1, P_2, \ldots, P_k$ be an HR of a polyhedron $P$. The problem of detecting the intersection of $P$ with a linear variety $S$ (a line or a plane) is transformed to the construction of the sequence of separating pairs $(p_1, s_1), (p_2, s_2), \ldots, (p_k, s_k)$, where, denoting by $H$ a plane orthogonal to $S$, $p_i \in (P_i)^H$ and $s_i \in S^H$ are a pair of points realizing the distance $\sigma(P_i, S)$ between $P_i$ and $S$. Given $(p_{i+1}, s_{i+1})$, the HR of $P$ enables us to obtain $(p_i, s_i)$ in time $O(1)$. Adapting the approach presented in [10], this is done as follows. On plane $H$, let $l_{i+1}$ be the line normal to $P_{i+1}S_{i+1}$ and passing by $p_{i+1}$, and let $l^+_{i+1}$ and $l^-_{i+1}$ be the two halfplanes defined by $l_{i+1}$ such that $(P_{i+1})^H \subset l^+_{i+1}$. Thus, $(P_i)^H = ((P_i)^H \cap l^+_{i+1}) \cup (P_i)^H \cap l^-_{i+1})$ and $\sigma(P, S) = \sigma((P_i)^H, S) = \min(\sigma((P_i)^H \cap l^+_{i+1}, S), \sigma((P_i)^H \cap l^-_{i+1}, S))$. Since $\sigma((P_i)^H \cap l^+_{i+1}, S)$ is realized by $(p_{i+1}, s_{i+1})$, what remains to be done is the construction of $(P_i)^H \cap l^-_{i+1}$. This is easily done from the HR of $P$. If $(P_i)^H \cap l^-_{i+1} \neq 0$, then at least one edge $e'$ of $(P_{i+1})^H$ is internal to $(P_i)^H$ (one such edge is either incident to $p_{i+1}$ - if $p_{i+1}$ is a vertex of $(P_{i+1})^H$ - or it contains $p_{i+1}$ in its interior). This edge $e'$ is the projection onto $H$ of an edge $e$ of $P_{i+1}$, incident with a facet $f$ of $P_{i+1}$ that is not a facet of $P_i$. By Property 5, facet $f$ has a pointer to a unique vertex $v$ of $P_i$; to identify this vertex it is sufficient to also associate with $e$ a pointer, stamped with the integer $i$, to vertex $v$. In this manner each edge has at most two pointers with the same stamp; it is immediate to select the correct one, by testing on which side of the line containing $e'$ the projection of the vertex pointed to lies. The projection $v'$ of $v$ on $H$ can therefore be found in time $O(1)$. If $v' \notin l^+_{i+1}$, then $(P_i)^H \cap l^-_{i+1} = \emptyset$ and we are done, i.e., $\sigma(P, S) = \sigma(P_i, S)$ and $(p_{i+1}, s_{i+1}) = (p_i, s_i)$. Otherwise, $v' \in l^-_{i+1}$ and we must find the supporting lines from $v'$ to $(P_{i+1})^H$ in order to compute $(P_i)^H \cap l^-_{i+1}$ (see Figure 4). Note that the supporting lines from $v'$ to $(P_{i+1})^H$ will be projections onto $H$ of the lines containing two of the edges incident to $v$ in $P_i$, and recall that $v$ is incident to at most $\mu - 1 = 6$ vertices of $P_{i+1}$. Therefore, the supporting lines from $v'$ to $(P_i)^H$ (and thus $(P_i)^H \cap l^-_{i+1}$) can be computed in $O(1)$ time. Once $(P_i)^H \cap l^-_{i+1}$ has been found, $\sigma((P_i)^H \cap l^-_{i+1}, S)$ and consequently, $\sigma((P_i)^H, S)$ and a corresponding separating pair $(p_i, s_i)$, can be be computed in $O(1)$ time.

It is easy to prove the claim that the additional edge-to-vertex pointer does not essentially alter the size of the data structure. Recall that the surface of each $P_i$, $i = 1, 2, \ldots, k$, is assumed to be triangulated. Each facet points to at most one vertex; therefore each facet may give rise to at most three edge-to-vertex pointers. Since the total number of facets is $O(|V(P)|)$, the claim is established.

We can now give an overall description of a 3D parallel algorithm based on the above ideas. We recall that a technique of Cole and Zajicek [4] can be used to build an HR of a convex polytope in $O(\log n)$ time using $n/\log n$ processors on an EREW PRAM. A hierarchical representation constructed in this manner will have $k = O(\log n)$. 

8
Figure 4: The supporting lines from \( v' \) to \((P_{i+1})^*(H)\) determine the structure of \((P_i)^*(H) \cap \Gamma_{i+1}^*\).

Theorem 1: The convex hull of a set of \( n \) points in three-dimensional space can be computed in \( O(\log^2 n) \) time using \( O(n) \) processors on a CREW PRAM.

Proof: It suffices to consider the "merge" step of the divide-and-conquer convex hull algorithm by Preparata and Hong [17]; we are dealing here with two separated polytopes \( P \) and \( Q \), whose surfaces may be assumed to be triangulated. We can build HRs of \( P \) and \( Q \) in \( O(\log n) \) time using \( n/\log n \) processors on an EREW PRAM [4]. Using the serial technique of Dobkin and Kirkpatrick [10] for each of the \( O(|P| + |Q|) \) edges of \( CH(P) \) and \( CH(Q) \), we can perform the line and plane queries required to determine if that edge is part of the seam of \( CH(P \cup Q) \) in \( O(k) = O(\log |P| + \log |Q|) = O(\log n) \) time using one processor on a CREW PRAM. (We need a CREW PRAM because the same HR of \( CH(Q) \) must be accessed for each edge of \( CH(P) \), and analogously for the HR of \( CH(P) \)). After we have determined which edges belong to the seam their cyclic connection order can be determined by a list-ranking process [13] in \( O(\log n) \) time using \( O(n) \) processors.

Once the cyclic connection order of the seam is determined, we can join \( CH(P) \) and \( CH(Q) \) to form \( CH(P \cup Q) \) with a simple merging process in \( O(\log n) \) time using \( O(n) \) processors as follows. Assume that the edges of both the upper and lower seams have been numbered in clockwise order with an edge receiving two indices if it is visited twice, and let there be \( n_p \) and \( n_q \) indices in the upper and lower seams, respectively. Using \( n_q \) processors we find the planes (and thus the faces) through edges \( e_1 \) and \( [e_{n_p}/2] \) of the upper seam that are tangent to the lower seam at vertices \( v_i \) and \( v_j \), respectively, in \( O(1) \) time. We now note that the planes through \( [e_{n_p}/4] \) and \( [e_{3n_p}/4] \) of the upper seam will be tangent to vertices \( v_{i'} \) and \( v_{j'} \) of the lower seam, such that \( i' \) lies on the portion of the lower seam between \( i \) and \( j \) and \( j' \) lies on the portion of the lower seam between \( j \) and \( i \). Thus, these planes can be found in constant time by assigning \((j-i) \mod n_q \) processors to \([e_{n_p}/4] \) and \((i-j) \mod n_q \) processors to \([e_{3n_p}/4] \), for a total of \( n_q \) processors. Continuing in this manner, the faces of \( CH(P \cup Q) \) that contain an edge of the upper seam can be found in \( O(\log n) \) time (the faces that contain
an edge of the lower seam can be found analogously). Note that because the surfaces of $\text{CH}(P)$ and $\text{CH}(Q)$ were triangulated, the surface of $\text{CH}(P \cup Q)$ will be triangulated as well. Thus, each stage of the divide-and-conquer process can be accomplished in $O(\log n)$ time using $O(n)$ processors on a CREW PRAM. The fact that there are $O(\log n)$ stages, each of complexity $O(\log n)$, establishes that the total complexity is $O(\log^2 n)$ time using $O(n)$ processors on a CREW PRAM.

References


