A NOTE ON DETERMINING THE 3-DIMENSIONAL CONVEX HULL OF A SET OF POINTS ON A MESH OF PROCESSORS+

(Preliminary Version)

by

Frank Dehne, Jörg-R. Sack,
School of Computer Science, Carleton University
Ottawa, Canada K1S 5B6

and

Ivan Stojmenović
Institute of Mathematics, University of Novi Sad
21000 Novi Sad, Yugoslavia

Abstract

This paper discusses the construction of the 3-dimensional convex hull for a set of n points stored on a $\sqrt{n} \times \sqrt{n}$ mesh of processors. Lu has shown that this problem can be solved in $\sqrt{n}$ log n time if all points are located on a sphere. Here, we solve, in the same time-complexity, the 3-dimensional convex hull problem for arbitrary point sets. Furthermore, we observe a time/space trade off: if each processor is allocated O(log n) space then $\sqrt{n}$ time is sufficient to determine the 3-dimensional convex hull.

+ Research supported by Natural Science and Engineering Research Council of Canada. This work was done in part while the third author was visiting Carleton University in November 1987.

F. Dehne, J.-R. Sack, and I. Stojmenovic, "A note on determining the 3-dimensional convex hull of a set of points on a mesh of processors," in Proc. *Scandinavian Workshop on Algorithm Theory*, Halmstad (Sweden), 1988, Springer Verlag, Lecture Notes in Computer Science, Vol. 318, pp. 154-162.

I. INTRODUCTION

The study of computational geometry algorithms on parallel architectures has recently received considerable attention as demonstrated e.g. by [ACGDY85], [JL87], [MS86]. Yap has shown that "almost all" computational geometry problems are in NC, i.e. can be solved on a PRAM in poly-log time [Y87]. By a designing algorithms for a particular geometric problem frequently a more efficient solution can be obtained than by the general method due to Yap. A survey of the current state of such activities is given in [MS88] and [DS88].

One of the most extensively studied problems in computational geometry (both sequential as well as parallel) is the determination of the convex hull for sets of points in two or three dimensions. For sequential algorithms and for a discussion of many applications the interested reader is referred to [PS85]. Parallel algorithms for solving these problems are surveyed in [DS88].

This paper addresses the problem of determining the convex hull of a three dimensional point set using the mesh connected computer architecture (MCC). The problem is to determine the faces of the smallest convex polytope enclosing the given 3-dimensional point set.

A particular instance of the problem arises in the computation of the 2-dimensional Voronoi diagram when the well known dual transformation due to [B79] is applied. To compute the 2-dimensional Voronoi diagram the input point set is mapped onto a sphere, the 3-d convex hull of the mapped points is computed, and the convex hull then determines the 2-d Voronoi diagram.

In 1986, Lu has given an $O(\sqrt{n} \log n)$ time mesh-algorithm for determining the Voronoi diagram of a planar set of n points using this technique. The procedure to compute the 3-d convex hull has been designed for the particular instance of the point set lying on a sphere. For arbitrary sets of points it is easy to find examples for which the algorithm will fail. Furthermore, this technique does not seem to generalize to arbitrary 3-dimensional point sets.

In this note we will present an $O(\sqrt{n} \log n)$ time algorithm for a $\sqrt{n} \times \sqrt{n}$ mesh of processors to determine the 3-dimensional convex hull of an arbitrary point set in $\Re^3$. The general structure of the algorithm, also employed by Lu, follows the well known divide and conquer paradigm for 3-d convex hull determination [PH77].

Algorithm 3-Dimensional Convex Hull

Input: A set $S=\{p_1, ..., p_n\}$ of n points in $\Re^3$ where each point is stored in one of the $\sqrt{n} \times \sqrt{n}$ processors.

Output: The (triangulated) faces of the convex hull stored in the processors

(1) Split the point set S by a plane parallel to the y-z plane into two (roughly) equal-sized halves called S_1 and S_2; move S_1 and S_2 into one half of the MCC, each.

(2) Recursively compute the convex hulls H_1 and H_2 of S_1 and S_2, respectively, in parallel.

(3) Merge the convex hulls obtained from Step (2).

Step (1) can be performed in time $O(\sqrt{n})$ by applying an $O(\sqrt{n})$ time sorting algorithm [TK77]. The recursion (Step(2)) induces an overhead of $O(\sqrt{n})$ [UL84], we thus focus on the crucial merging step: Given two 3-dimensional convex hulls each stored in one half of the MCC, determine the convex hull faces of their union.

Excluding degeneracies, each face of the convex hull of S that is not a face of H_1 or H_2 is determined by an edge-vertex pair, where the edge and the vertex belong to different hulls. These edges and vertices are referred to as circuit edges and vertices, respectively [Lu87].

To determine whether a given edge is part of the final hull, Lu tests whether at most one of the planes incident to the edge is interior to the final hull. Unfortunately, for arbitrary point sets this test may fail, since an edge may belong to the final hull although both of its incident faces are interior. (An example of this situation is easily constructed and is left to the reader.)

The approach taken in this paper is to examine all edges of one hull (in parallel) and construct for each edge its supporting plane with respect to the other hull. If, for some edge e in H_1, no such plane exists then the line containing e intersects H_2 and, therefore, e is not on the final hull. Otherwise, it is easy to determine from the two faces incident with e in H_1 and the supporting planes, whether e is an edge of the final hull.

In the following section we will describe an $O(\sqrt{n} \log n)$ time algorithm for computing on a MCC of size n for an n-vertex polyhedron P and a set of n edges in $\Re^3$ for each edge e its supporting plane, i.e. the plane which contains e and is a tangent plane to P, if exists. This method is based on the hierarchical decomposition techniques developed in [DK87].

Utilizing this method we will then introduce, in Section 3, an $O(\sqrt{n} \log n)$ time implementation of step (3) which yields an $O(\sqrt{n} \log n)$ time complexity for the entire algorithm.

In Section 4 we will show that if each processor is allocated O(log n) space our method can be modified to run in time $O(\sqrt{n})$.

2. MULTIPLE TANGENT PLANE DETERMINATION

Consider an n-vertex convex polytope P and a set $E=\{e_1,...,e_m\}$ of $m=O(n)$ edges in $\Re^3$ (which are located exterior to P). The multiple tangent plane determination problem consists of finding for each edge e_i the tangent planes t_1, t_2 (if it exists) which contain e_i and are tangent planes to P.

This problem has been solved in [DK87] for the CREW-PRAM model in time $O(\log n \log^* n)$. Since the vertices and edges of a convex polytope form a planar graph, Dadoun and Kirkpatrick [DK87] are able to use a hierarchical representation of P which is equivalent to the well known hierarchical representation of a planar graph used for planar point location [K83].

The hierarchical representation of P is a sequence $P_1, ..., P_k$ of convex polytopes with vertex sets $V_1, ..., V_k$, respectively, such that

- $P_1 = P$
- $|V_k|$ is bounded by a constant
- $V_{i+1} \subseteq V_i$
- the vertices of V_i which are not in V_{i+1} and are independent (i.e., non-adjacent) in P_i
- $|V_{i+1}| \leq \alpha\ |V_i|$, $0<\alpha<1$, therefore $k \leq O(\log n)$ with $|P_i|$ denoting the size, i.e. number of vertices, edges or faces of P_i

In [DK87] the hierarchical representation is computed iteratively starting with $P_1=P$. Note that in the general case (no more than three points lie on the same plane) P consists of triangular faces only; therefore, constructing an initial triangulation, as in the general case considered in [DK87], is not necessary. In case of non-triangular faces, these can be triangulated in time $O(\sqrt{n})$ as follows: all edges of the polygon can be ranked (with respect to their circular order) using the list ranking technique in [AH86] and then a triangulation can be obtained by repeatedly connecting alternate vertices.

The main problem in computing P_{i+1} from P_i is to identify in P_i an independent set I of low-degree vertices such that $|I| > |V_i|/c$ for some fixed constant c. In [DK87] this problem is referred to as the fractional independent set problem and its time complexity is denoted by $FISP(|V_i|)$. There, it is shown that on the CREW-PRAM $\qquad FISP(|V_i|) \leq O(1) + c\ L\text{-}FISP(|V_i|)$

where $L\text{-}FISP(|V_i|)$ denotes the time complexity on a CREW-PRAM to solve the fractional independent set problem for a linear list with with $|V_i|$ vertices.

Since every O(1) time CREW-PRAM operation on V_i can be simulated on an MCC in time $O(\sqrt{|V_i|})$, it follows

$$FISP_{MCC}(|V_i|) \leq O(\sqrt{|V_i|}) + c\ L\text{-}FISP_{MCC}(|V_i|)$$

where $FISP_{MCC}(|V_i|)$ and $L\text{-}FISP_{MCC}(|V_i|)$ denote the time complexity to solve the fractional independent set problem and fractional independent set problem for linear lists, respectively, on an MCC. Note that we assume that the data for P_i are compressed into a subsquare of the MCC.

Furthermore, it is easy to see that $L\text{-}FISP_{MCC}(|V_i|)=O(\sqrt{|V_i|})$:

Using the algorithm in [AH86] to compute on an MCC the depth of all nodes of an n-vertex tree in time $O(\sqrt{n})$ all vertices of a linear list with $|V_i|$ vertices can be ranked. A fractional independent set can, thus, be determined in time $O(\sqrt{|V_i|})$.

Hence, $FISP_{MCC}(|V_i|) \le O(\sqrt{|V_i|})$ and, therefore, we get for the time $T_H(n/2)$ for computing on an MCC the hierarchical representation of P the recurrence

$$T_H(n) \le T_H(\alpha\, n) + O(\sqrt{n}), \quad 0<\alpha<1,$$

thus, $T_H(n) = O(\sqrt{n})$.

Note that since $|V_{i+1}| \le \alpha\, |V_i|$, $0<\alpha<1$, the space necessary to store all polytopes P_i is O(n).

In order to compute on an MCC for all edges e_i ($1 \le i \le m$) the tangent planes with respect to P, first, the tangent planes for all e_i with respect to P_k are determined by broadcasting P_k to all e_i. Together with these tangent planes each e_i also receives information about the constant number of edges and vertices of P_{k-1} it has to test in order to find its tangent planes with respect to P_{k-1}. Hence, the tangent planes for all e_i with respect to P_{k-1} can be computed by a constant number of random access read operations (see e.g. [MS84]) in time $O(\sqrt{n})$. This process is iterated O(log n) times until for all e_i the tangent planes with respect to P have been computed.

Summarizing, we have shown that given an n-vertex convex polytope P and a set $E=\{e_1,\ldots,e_m\}$ of $m=O(n)$ edges in $\Re^3$ together stored in an MCC of size n

a) the hierarchical representation of P can be computed in time $T_H(n)= O(\sqrt{n})$, and

b) the multiple tangent plane determination problem can be solved in time $T_{MT}(n)=O(\sqrt{n}\log n)$.

3. MERGING 3-DIMENSIONAL CONVEX HULLS ON AN MCC

We will now show how to merge two convex hulls H_1 and H_2 for sets S_1 and S_2 respectively, in time $O(\sqrt{n}\log n)$, on an MCC of size n. First, we determine those edges in H_1 or H_2 which are also edges of the resulting hull H:

(3.1) Solve the multiple tangent plane determination problem for all edges of H_1 with respect to polyhedron H_2, and for all edges of H_2 with respect to polyhedron H_1.

(3.2) For each edge e of H_1 [H_2] consider its tangent planes t_1, t_2 (if exists) as well as its two adjacent faces f_1 and f_2 in H_1 [H_2, respectively].

If t_1 and t_2 do not exist then, obviously, e is not an edge of H. Otherwise, e is an edge of H if and only if at least one of the two tangent planes t_1 and t_2 is also a tangent plane of H. The plane t_1 [t_2] is a tangent plane of H if t_2 [t_1], f_1, and f_2 are contained in the same halfspace created by t_1 [t_2, respectively]; see Figure 1.

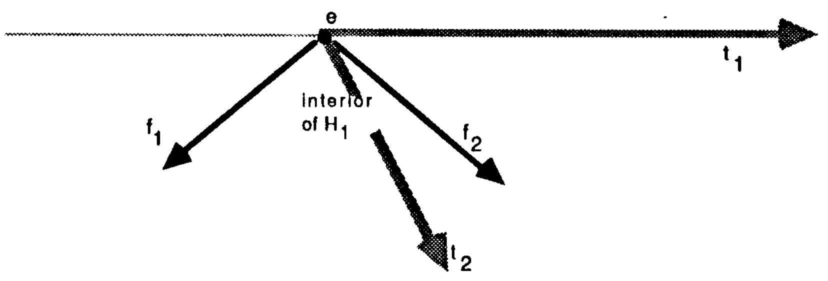

Figure 1: Deciding whether e is an edge of H ; t_1 is a tangent plane of H while t_2 is not.
(Seen after projection onto the plane perpendicular to e)

The following steps determine all edges and faces of H.

(3.3) All faces of H_1 and H_2 for which all three edges are edges of H, are faces of H.

(3.4) The additional edges and faces of H which are not contained in H_1 and H_2 can be easily obtained from Step (3.2) as follows: Each additional face (triangle) corresponds to a tangent plane of an edge e of H_1 [H_2], which is also a tangent plane of H, i.e. it is defined by e and the point of H_2 [H_1] contained in the tangent plane.

From Section 2 we know that Step (3.1) can be performed on an MCC in time $O(\sqrt{n} \log n)$. Furthermore it is easy to see that Steps (3.2), (3.3), and (3.4), together, can be performed in time $O((\sqrt{n}))$. Hence, given the convex hulls H_1 and H_2 of S_1 and S_2, respectively, the convex hull H of S can be computed on an MCC of size n in time $O(\sqrt{n} \log n)$. Therefore, this implementation of Step (3) of the algorithm presented in Section 1 results in the following main theorem:

Theorem: The 3-dimensional convex hull of an n point set in $\Re^3$ can be determined in $O(\sqrt{n} \log n)$ time on an MCC of size n.

4. MERGING 3-DIMENSIONAL CONVEX HULLS ON AN MCC OF SIZE N WITH O(LOG N) SPACE ALLOCATED AT EACH PROCESSOR

Usually, the definition of an MCC of size n implies that each of the processors has only a constant amount of memory available. As shown above, the 3-dimensional convex hull can then be computed in time $O(\sqrt{n} \log n)$. In this section we will show that the time can be reduced to $O(\sqrt{n})$ if each processor has O(log n) memory available.

The only step in the above algorithm which induces time $O(\sqrt{n} \log n)$ is Step (3.1), i.e. the solution of the multiple tangent plane determination problem for a set of m edges with respect to an n-vertex convex polytope P. Provided that this problem could be solved in time $O(\sqrt{n})$ the entire algorithm would have a running time of $O(\sqrt{n})$.

The hierarchical representation $P_1, \ldots, P_k$ of P can be computed in time $O(\sqrt{n})$ as shown in Section 2. The $O(\sqrt{n} \log n)$ time complexity for the multiple tangent plane determination problem results from iterating the search processes for all edges exactly $k=O(\log n)$ times, once for each P_i. To reduce this time, we exploit the fact that the sizes of P_i and P_{i+1} are related as $|P_{i+1}| \leq \alpha\, |P_i|$ for some α, $0<\alpha<1$.

Assume for a moment that we only want to store one of the polytopes, say P_i on the MCC. In this case only $n\,\alpha^{(i-1)}$ processors suffice. Therefore, we can store multiple copies of P_i on the mesh by partitioning the MCC into (roughly) $\alpha^{(1-i)}$ subsquares of size $n\,\alpha^{(i-1)}$; each such subsquare stores one copy of P_i. The storage structure for the entire hierarchical representation of P is constructed by overlaying the storage structures for all P_i, $1 \leq i \leq k$. An example for $\alpha=\frac{1}{4}$ and k=3 is given in Figure 2.

Notice that the construction time for creating the hierarchical subdivision remains $O(\sqrt{n})$. This is seen by observing that for all i, each subsquare containing P_i computes in parallel P_{i+1} independent on the other processor elements; these processors are idle in the previous algorithm described above. Subsequently P_{i+1} is duplicated the appropriate number of times. Every processor is part of k storage structures and needs O(1) space for each of $P_1, \ldots, P_k$. Since $k=O(\log n)$, the storage requirement per processor is $O(\log n)$.

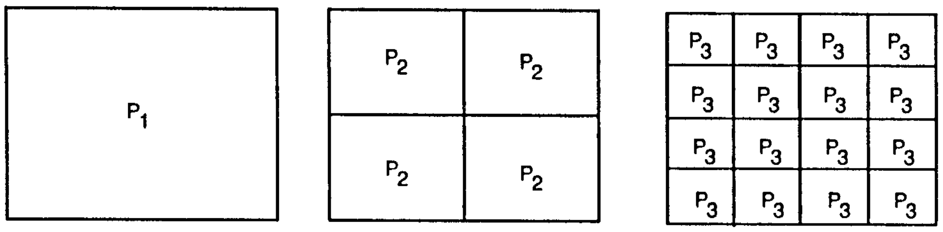

Figure 2: Storage structure for the hierarchical representation of P
(Overlay all partitionings)

Every processor is part of k storage structures and needs O(1) space for each of $P_1, \ldots, P_k$. Since $k=O(\log n)$, the storage requirement is $O(\log n)$ per processor.

In order to compute on an MCC for all edges the tangent planes with respect to P, the tangent planes for all e_j with respect to $P_k, P_{k-1}, \ldots, P_i, \ldots, P_1$ are determined. However, in contrast to the solution in

Section 2, the determination of all tangent planes in P_i (after those in $P_k,\ldots,P_{i-1}$ have been determined) can now be performed in time $O(\sqrt{n\,\alpha^{(i-1)}})$. This fact is easily verified by observing that all broadcast and random access read operations can now be executed in parallel on all $n\,\alpha^{(1-i)}$ subsquares of size $n\,\alpha^{(i-1)}$ in time $O(\sqrt{n\,\alpha^{(i-1)}})$. Hence, the time to compute for all edges the tangent planes with respect to P is:

$$\sum_{1\le i\le k} O(\sqrt{n\,\alpha^{(i-1)}}) = O(\sqrt{n}) .$$

Theorem: Using O(log n) extra storage allocated to each processor element, the 3-dimensional convex hull of an n point set in $\Re^3$ can be determined in $O(\sqrt{n})$ time on an MCC of size n.

CONCLUSIONS

We have shown that the 3-dimensional convex hull of n points stored on a MCC of size $\sqrt{n} \times \sqrt{n}$ can be determined in $O(\sqrt{n}\log n)$ time. It is also demonstrated that by allocating O(log n) storage to each processing element an $O(\sqrt{n})$ algorithm was designed. It is open whether an $O(\sqrt{n})$ MCC algorithm for this problem exists, with O(1) extra storage.

REFERENCES

[ACGDY85] A. Aggarwal, B. Chazelle, L. Guibas, C. O. Dunlaing, and C. Yap, "Parallel computational geometry", Proc. IEEE Symp. on Found. of Computer Science, Portland, Oregon, Oct. 1985

[AH86] M. Atallah, S. Hambrusch, "Solving tree problems on a mesh-connected processor array", Information and Control, Vol. 69, Nos. 1-3, 1986.

[B79] K. Q. Brown, "Voronoi diagrams from convex hulls", Information Processing Letters, Vol.9, 1979, pp. 223-228.

[DK87] N. Dadoun, D. G. Kirkpatrick, "Parallel construction of subdivision hierarchies", Proc. of the 3rd ACM Conference on Computational Geometry, 1987, pp. 205-214

[DS88] F. Dehne, J.-R. Sack, "A survey of parallel computational geometry algorithms", Tech. Rept. School of Computer Science, Carleton University, Ottawa, Canada, 1988, to be presented at Parcella '88, Berlin, GDR, October 1988

[K83] D. G. Kirkpatrick, "Optimal search in planar subdivisions", SIAM Journal of Computing 12,1, 1983, pp. 28-35.

[JL87] C. S. Jeong, D. T. Lee, "Parallel geometric algorithms on mesh-connected computers", FJCC, 1987.

[Lu87] M. Lu, "Constructing the Voronoi diagram on a mesh-connected computer", Proc. of the 1986 IEEE Conference on Parallel Processing, St. Charles, Ill., 1986, pp. 806-811.

[MS84] R. Miller, Q. F. Stout, "Computational geometry on a mesh-connected computer", Proc. Int. Conf. on Parallel Processing, 1984.