

**Original citation:**

Paterson, Michael S. and Yao, F. F. (1989) Binary partitions with applications to hidden-surface removal and solid modelling. University of Warwick. Department of Computer Science. (Department of Computer Science Research Report). (Unpublished)  
CS-RR-139

**Permanent WRAP url:**

<http://wrap.warwick.ac.uk/60834>

**Copyright and reuse:**

The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions. Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

**A note on versions:**

The version presented in WRAP is the published version or, version of record, and may be cited as it appears here. For more information, please contact the WRAP Team at: [publications@warwick.ac.uk](mailto:publications@warwick.ac.uk)



<http://wrap.warwick.ac.uk/>

# Research report 139

## BINARY PARTITIONS WITH APPLICATIONS TO HIDDEN-SURFACE REMOVAL AND SOLID MODELLING

Michael S Paterson\* & F Frances Yao†

(RR139)

### Abstract

We consider schemes for recursively dividing a set of geometric objects by hyperplanes until all objects are separated. Such a *binary partition* is naturally considered as a binary tree where each internal node corresponds to a division and the leaves correspond to the resulting fragments of objects. The goal is to choose the hyperplanes properly so that the *size* of the binary partition, i.e., the number of resulting fragments of the objects, is minimized. We construct binary partitions of size  $O(n \log n)$  for  $n$  edges in the plane, and of size  $O(n)$  if the edges are orthogonal. In three dimensions, we obtain binary partitions of size  $O(n^2)$  for  $n$  planar facets, and prove a lower bound of  $\Omega(n^{\frac{3}{2}})$ . Two applications of efficient binary partitions are given. The first is an  $O(n^2)$ -sized data structure for implementing a hidden-surface removal scheme of Fuchs, Kedem and Naylor [5]. The second application is in solid modelling: given a polyhedron described by its  $n$  faces, we show how to generate an  $O(n^2)$ -sized CSG (*constructive-solid-geometry*) formula whose literals correspond to half-spaces supporting the faces of the polyhedron (see Peterson [9] and Dobkin et al. [3]). The best previous results for both of these problems were  $O(n^3)$ .

---

\*The research was done while this author was visiting Xerox Palo Alto Research Center.

†Xerox Palo Alto Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, USA

## 1. Introduction

Recursive partitioning is a basic problem-solving technique which has proven to be extremely useful in algorithm design. For geometric problems where the input is a set of objects in the plane or in space, it is natural for this 'divide and conquer strategy' to be employed in such a way that the divide step is accomplished by making a *linear cut* of the input, that is, by splitting the objects along a line (in the 2-D case) or along a plane (in the 3-D case). This creates two subproblems which can then be divided recursively, again by linear cuts, until finally subproblems of some trivial size are obtained. Since each divide step may split some of the objects into several parts, the process described above can lead to a proliferation of objects and result in an inefficient algorithm. Thus, one is motivated to choose the dividing cuts carefully so that fragmentation of the input objects is kept to a minimum. We refer to the recursive partition mentioned above as a *binary partition*, and we are interested in the question of constructing binary partition trees whose size is not too large as a function of the original input size.

For example, in the three-dimensional formulation of our problem, we take the input to consist of a set of  $n$  non-intersecting triangular faces in  $R^3$ , since triangular tiling is common for representing surfaces in space. Let  $p(n)$  be the maximum value over all inputs of size  $n$  of the size of a minimal binary partition, as measured by the number of internal nodes in the corresponding binary tree. (Precise definitions of binary partitions and  $p(n)$  are given in Section 2.) A straightforward upper bound for  $p(n)$  is  $O(n^3)$ . One of the main results of this paper is that  $p(n) = O(n^2)$ . A corresponding lower bound of  $p(n) = \Omega(n^{1.5})$  follows from an example due to Thurston [10].

As applications of efficient binary partitions, we describe two well-known problems in computer graphics and show that solutions better than those previously known can be obtained readily from our results. The first problem arises in removing hidden surfaces in real-time. In some graphics applications such as flight simulation and computer animation, one needs to generate rapidly images of a 3-D scene as viewed from changing positions. A good strategy is to preprocess the scene suitably so as to simplify the hidden-surface computation at run-time. Fuchs, Kedem and Naylor [5] observed that, if the objects comprising the scene are represented as a binary partition tree, then traversal of the tree in a symmetric order relative to the viewing position will produce a correct priority order of (the fragments of) the objects for achieving the desired obscuring effect. In this scheme, storage space as well as tree traversal time is proportional to the size of the tree. The only previous bound known for the tree size is  $O(n^3)$  [5]. As a direct consequence of our main theorem, this bound can be improved to  $O(n^2)$ .

The second application of binary partitions is found in converting boundary representations of three-dimensional objects into constructive-solid-geometry (CSG) representations. The boundary representation of an object specifies the surface elements forming its boundary; whereas the CSG representation expresses the interior of the object by a boolean formula where the operations are intersection and union and the literals are primitive solids such as boxes, cylinders, etc. It is important in solid modelling to be able to convert efficiently between the two styles of representation. In a special type of CSG representation considered by Peterson [9], the literals correspond to the half-spaces supporting faces of the

given object. A natural question is: given a polyhedron described by its  $n$  faces, can one generate a short Peterson-style formula? A straightforward upper bound on the size of the formula is  $O(n^3)$ . We will show that our result on binary partitions implies an  $O(n^2)$  bound on formula size.

In Section 2, we give the definition and basic properties of binary partitions. In Sections 3 and 4 partitions of size  $O(n \log n)$  for the planar case are presented. Section 5 contains our main result, an algorithm to find partitions of size  $O(n^2)$  in three dimensions, which is complemented by the  $\Omega(n^{\frac{3}{2}})$  lower bound of Section 6. In Section 7 we show that for orthogonal line segments in the plane a linear solution is achievable. The applications are discussed in Section 8, and we conclude with some comments and open problems in Section 9.

## 2. Preliminaries

In this section we give the mathematical formulation of the binary partition problem, and discuss some basic properties of binary partitions.

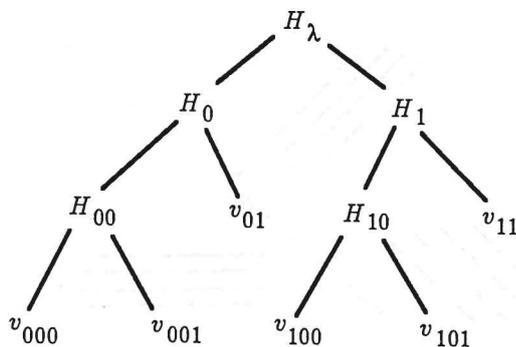


Figure 1.

A  $d$ -dimensional *binary partition*  $P$  is a recursive partition of  $d$ -dimensional Euclidean space,  $R^d$ , defined by a set of hyperplanes. Let  $\mathcal{H}$  be a collection of (oriented) hyperplanes that are organized as a binary tree and labelled accordingly as  $H_\lambda, H_0, H_1, H_{00}, H_{01}, \dots$  (see Figure 1). Then  $\mathcal{H}$  defines a binary partition  $P$  under which  $R^d$  is first partitioned by the root hyperplane  $H_\lambda$  into two open half-spaces,  $H_\lambda^-, H_\lambda^+$ , and  $H_\lambda$  itself. Recursively,  $H_\lambda^-$  and  $H_\lambda^+$  are partitioned by the subtrees rooted at  $H_0$  and  $H_1$  respectively. We will refer to the hyperplanes  $H_i \in \mathcal{H}$ ,  $i \in \{0, 1\}^*$ , as the *cut hyperplanes* (in particular, *cut lines* when  $d = 2$  and *cut planes* when  $d = 3$ ) of the partition. For any node  $v$  of the tree we define  $R(v)$  to be the convex region which is the intersection of all the open half-spaces defined at the ancestor nodes of  $v$ . The components of the partition  $P$  then consist of  $R(v)$  for each leaf node  $v$ , and, for every internal node  $v$ , the intersection of  $R(v)$  with  $H_v$ , the hyperplane at  $v$ .

To simplify our presentation we shall ignore the  $(d - 1)$ -dimensional regions of  $P$  corresponding to the internal nodes of the tree. Though they may have a non-trivial content, in many applications only  $(d - 1)$ -dimensional regions are of consequence. In any case the

reduction in dimension allows a simple recursive structure and we shall concentrate on the broader complexity issues presented in the top dimension.

Let  $\Sigma$  be a collection of *facets*, simplices of dimension  $(d - 1)$  or less, in  $R^d$ . (One-dimensional facets are line segments and two-dimensional facets are triangles.) For a given  $\Sigma$ , we shall be interested in binary partitions  $P$  of  $R^d$  with the property that, at each leaf  $v$ , the set  $\Sigma \cap R(v)$  is empty; we refer to such a  $P$  as a *complete binary partition* of  $\Sigma$ . The *size* of a binary partition,  $|P|$ , is the number of its internal nodes, and the *partition complexity* of  $\Sigma$ , denoted by  $p(\Sigma)$ , is  $\min\{|P| \mid P \text{ is a complete binary partition of } \Sigma\}$ . Define  $p(n) = \max\{p(\Sigma) \mid |\Sigma| = n\}$ . Complete binary partitions will often be called *binary partitions* or simply *partitions*.

For any facet  $A$ , we define *hyperplane*( $A$ ) to be the hyperplane through  $A$  with some definite (but arbitrary) orientation. In two and three dimensions we shall use the terms *line*( $A$ ) and *plane*( $A$ ) respectively.

A natural class of binary partitions can be obtained by imposing the restriction that each cut hyperplane be *hyperplane*( $A$ ) for some facet  $A$  in  $\Sigma$ . Such a partition will be called an *autopartition*.

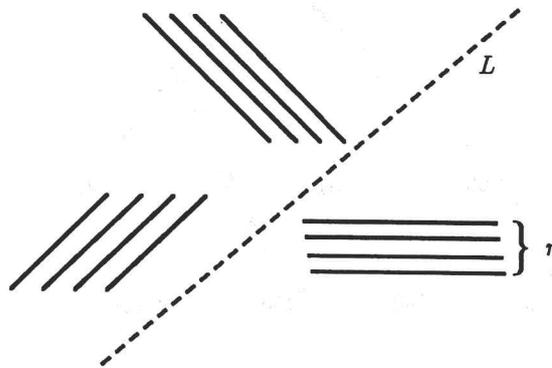


Figure 2.

Note that a minimum partition for  $\Sigma$  is not always achievable by an autopartition.

**Example 1** In Figure 2,  $\Sigma$  consists of three sets of  $r$  parallel segments, where the sets would cut each other cyclically. It can be verified that any autopartition of  $\Sigma$  has size at least  $4r$ , but a partition that begins with a cut line such as  $L$  shows that  $p(\Sigma) = 3r + 1$ .

In three dimensions we can demonstrate a greater disparity between autopartitions and general binary partitions.

**Example 2** In Figure 3, two sets of  $r$  parallel planes attack each other. In this case every autopartition has size exactly  $r^2$  but there is a binary partition which begins with a plane separating the two sets and has size only  $2r + 1$ .

If  $\Sigma$  has no co(hyper)planar facets then the region corresponding to each internal node contains at most one  $(d - 1)$ -dimensional fragment of a facet, and so the size of the partition is at least as large as the total number of such fragments generated. In an autopartition of such a  $\Sigma$ , these numbers are equal and we will refer to them interchangeably.

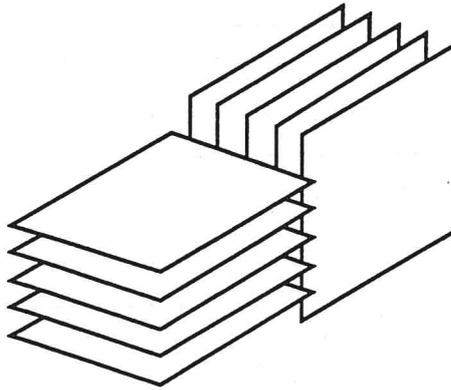


Figure 3.

There is a simple yet useful device which prevents excessive fragmentation of  $\Sigma$  during a partition. Consider the two-dimensional case. Assume that at some stage of a partition we have a region  $S$  and a segment  $A$  which is a chord of  $S$ . Then  $A$  divides  $S$  into two regions,  $S_0$  and  $S_1$ , and any other segment of  $\Sigma \cap S$  must lie completely within one of these regions. In such a situation, it is always advantageous to partition  $S$  immediately along the line  $A$ , since this cuts through no segments, and prevents the segment  $A \cap S$  from ever being cut.

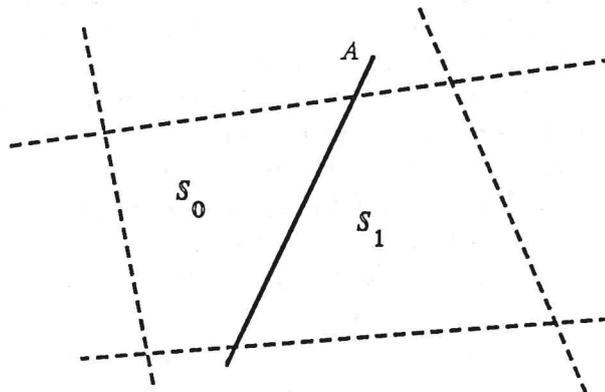


Figure 4.

More generally, in higher dimensions, the above observation holds for any region which is completely separated by some facet  $A$ . We shall term the cut defined by hyperplane( $A$ ) in such a situation a *free cut*.

### 3. $O(n \log n)$ Partitions in Two Dimensions

In this section and the next we consider the two-dimensional partitioning problem. The motivation is two-fold. Some special forms of 3-D objects such as prisms can be treated as 2-D objects directly, and the algorithms introduced in Section 4 provide useful insight for the higher-dimensional case later.

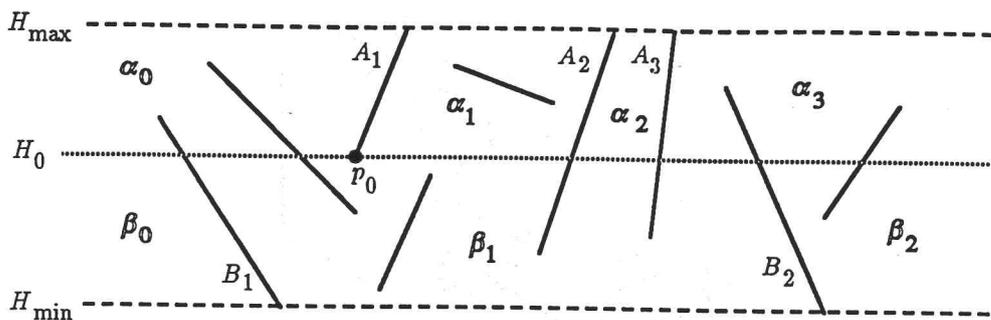


Figure 5.

A recursive procedure which performs binary splitting on the set of endpoints of the segments and takes advantage of free cuts yields our first result.

**Theorem 1.** For any  $n$  disjoint line segments in the plane, there is a complete binary partition of size  $O(n \log n)$ , i.e.,  $p(n) = O(n \log n)$ .

**Proof.** Let  $\Sigma$  be a set of input line segments, and  $V$  be the set of endpoints of  $\Sigma$ . Our algorithm initially finds two points  $p_{\min}$  and  $p_{\max}$  of  $V$  with minimum and maximum  $y$ -coordinates. Thus all segments of  $\Sigma$  lie in the strip bounded by the two horizontal lines  $H_{\max}$  and  $H_{\min}$  going through  $p_{\max}$  and  $p_{\min}$  respectively. In each stage of the algorithm, we select a point  $p_0$  of  $V$  which has the median  $y$ -coordinate among all points of  $V$ . The horizontal line  $H_0$  going through  $p_0$  splits  $\Sigma$  into two sets of segments  $\Sigma_a$  and  $\Sigma_b$ , which lie above and below  $H_0$  respectively. Let  $A_1, A_2, \dots, A_s$  be the segments in  $\Sigma_a$  which intersect both  $H_{\max}$  and  $H_0$ . Free cuts using these segments divide the strip between  $H_{\max}$  and  $H_0$  into  $s + 1$  regions ordered from left to right. We use  $\alpha_i$ , for  $0 \leq i \leq s$ , to denote the region that lies between  $A_i$  and  $A_{i+1}$ . Similarly, if  $B_1, B_2, \dots, B_t$  are the segments that intersect both  $H_{\min}$  and  $H_0$ , then free cuts with these segments separate  $\Sigma_b$  into  $t + 1$  disjoint regions ordered from left to right, denoted by  $\beta_j$ , for  $0 \leq j \leq t$ . (See Figure 5.) We then repeat the partitioning for each of the  $\alpha_i$  and  $\beta_j$  separately.

The partition scheme given above can be represented by a multi-way tree. (See Figure 6.) This tree eventually needs to be transformed into a binary partition but the analysis below is based on the multi-way tree structure.

We will show that in the partition defined above, each original segment of  $\Sigma$  is cut into at most  $O(\log n)$  pieces. In any region  $R(v)$ , let  $m(v)$  be the number of endpoints of  $\Sigma$  that are in the interior of  $R(v)$ . Then, since  $H_0$  is a median divider, each of the regions  $\alpha_i$ ,  $0 \leq i \leq s$  and  $\beta_j$ ,  $0 \leq j \leq t$ , has  $m \leq m(v)/2$ . Thus the multi-way partition tree has depth at most  $\log_2 n$ . If segment  $\ell$  of  $\Sigma$  is cut into two pieces for the first time at some node  $v$ , then each of the two pieces can be further cut only along the unique path from  $v$  leading to the node  $v'$  such that  $R(v')$  contains the endpoint of that piece. Along any other path a subsegment of  $\ell$  is eliminated by a free cut. Hence  $\ell$  is cut into at most  $2 \log_2 n$  fragments.  $\square$

A specific example  $\Sigma$  which achieves the worst case bound  $O(n \log n)$  under the above procedure can be easily constructed.

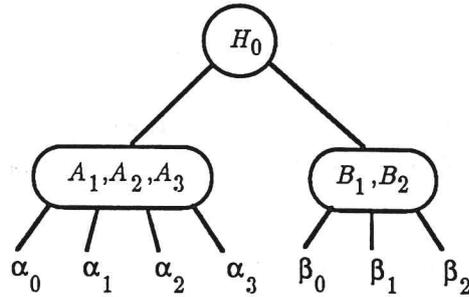


Figure 6.

Although our major concern is to minimize the size of the partition, we would like to have an efficient algorithm to find small partitions and in some applications the depth of the corresponding tree may also be important. Both concerns are addressed in the following strengthening of Theorem 1.

**Theorem 2.** There is an  $O(n \log n)$ -time algorithm which, for  $n$  disjoint line segments in the plane, produces a complete binary partition of size  $O(n \log n)$  and depth  $O(\log n)$ .

**Proof.** We outline an algorithm which has the stated bounds. In the scheme used in the proof of Theorem 1, we can achieve logarithmic depth by constructing a balanced binary partition after applying free cuts to some horizontal strip. Suppose the number of endpoints in  $\alpha_i$  is  $m_i$ , then using the balancing algorithm due to Gilbert and Moore [6] (see Knuth [8], Section 6.2.2) on the sequence of weights  $m_1, m_2, \dots$ , the depth of  $\alpha_i$  in the binary expansion of the multi-way branch is at most  $2 + \log_2(\sum m_j) - \log_2(m_i)$ . In the worst case the depth of the resulting partition is at most  $3 \log_2 n + O(1)$ . The running time of this step is  $O(\sum_i (\log(\sum m_j) - \log(m_i)))$ , and contributes a total of only  $O(n \log n)$  to the running time.

The whole construction can be completed in  $O(n \log n)$  time by an algorithm of the following kind. Since the median-splitting is done with respect to the original endpoints, if these points are sorted initially by their  $y$ -coordinates then all the splits can be made by index calculations on the sorted list. The separation of segments and subsegments by the free cuts is more of a problem; for example, segment  $u$  may lie entirely to the left of segment  $v$  but a slanting cut can put  $u$  into a region to the right of that containing  $v$ .

The latter difficulty can be overcome by using a total 'right-to-left' ordering,  $\succ$ , of all the segments with the property that if  $u \succ v$  then no subsegment of  $u$  can ever be in a region in the same horizontal slice as, and to the left of, a region containing a subsegment of  $v$ . The partial order 'xdom', introduced by Guibas and Yao [7], is defined by  $u \text{ xdom } v$  if some point of  $u$  has the same  $y$ -coordinate as, but a larger  $x$ -coordinate than, some point of  $v$ . It can be verified that any extension of xdom to a total order will serve our purpose. Such a total order can be found in  $O(n \log n)$  time by using a plane sweep algorithm [7].

While our partitioning procedure is working on some slice, it will have available the restriction to this slice of the ' $\succ$ ' relation. When a horizontal cut is made, the two resulting restrictions can be produced in linear time; and when a sequence of free cuts is made the

restrictions to each subregion are found by partitioning the total order with respect to the cutting segments.

Our claim that the total running time is  $O(n \log n)$  is proved by showing that, after an initial  $O(n \log n)$  preprocessing stage, each level of the depth- $O(\log n)$  partition tree is generated in only  $O(n)$  time.  $\square$

#### 4. Probabilistic Methods for Planar Partitions

In this section we present two further algorithms for planar partitions; while the first is somewhat simpler, the second appears to be more efficient. For each an  $O(n \log n)$  size bound will be proved using probabilistic arguments.

We define a special form of autopartition in which the facets of  $\Sigma$  are used as cutting hyperplanes according to some specified linear order.

**Definition.** Let the facets in  $\Sigma$  be denoted by  $\{u_1, u_2, \dots, u_n\}$ , and let  $\pi$  be a permutation of  $\{1, 2, \dots, n\}$ . A unique binary partition can be obtained by the following recursive construction.

While a region is nonempty, make a cut with hyperplane( $u_i$ ) where  $i$  is first in the ordering  $\pi$  such that  $u_i$  intersects the region. Then proceed recursively in each subregion.

The resulting partition is the *autopartition with respect to  $\pi$* , written as  $P_\pi$ .

The size of the autopartition  $P_\pi$  on  $n$  line segments in the plane for a random choice of  $\pi$  is shown below to be  $O(n \log n)$ . We then present a refinement of this construction which will be generalized to an efficient method for higher dimensions in the next section. We also give a corresponding deterministic algorithm for finding an  $O(n \log n)$  autopartition.

**Theorem 3.** The expected size of the autopartition  $P_\pi$  for  $n$  segments in  $R^2$  when  $\pi$  is chosen uniformly over all  $n!$  possibilities is  $O(n \log n)$ .

**Proof.** For line segments  $u, v$ , we define  $\text{index}(u, v) = i$  if line( $u$ ) intersects  $i - 1$  other segments before hitting  $v$ , and  $\text{index}(u, v) = \infty$  if  $\text{line}(u) \cap v = \emptyset$ . Since a segment  $u$  can be extended in two directions, we may have  $\text{index}(u, v) = i$  for two different  $v$ 's. (See Figure 7 where  $\text{index}(u, v) = 3$ .) We will say for short that ' $u$  cuts  $v$ ' when line( $u$ ) divides segment  $v$  in the partition.

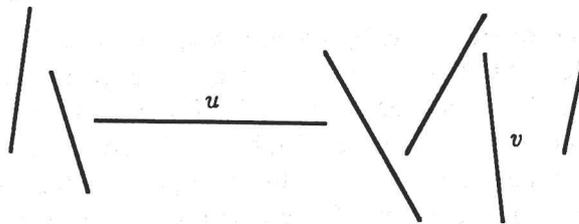


Figure 7.

First we show that the probability that  $u$  cuts  $v$  in  $P_\pi$  for a random  $\pi$  is at most  $\frac{1}{\text{index}(u, v) + 1}$ .

Assume  $\text{index}(u, v) = i$ , and let  $u_1, u_2, \dots, u_{i-1}$  be the segments that  $\text{line}(u)$  intersects before hitting  $v$ . The event ' $u$  cuts  $v$ ' can happen only if  $u$  is chosen as a cut line before any of  $\{u_1, u_2, \dots, u_{i-1}, v\}$  is chosen. The probability of the latter event is  $1/(i+1)$ .

The size of an autopartition is equal to the number of fragments generated, i.e.,  $n$  plus the number of intersections. Therefore the expected size,  $E(P_\pi)$ , of  $P_\pi$  satisfies:

$$\begin{aligned} E(P_\pi) &= n + \sum_{u,v} \text{Prob}(u \text{ cuts } v \text{ in } P_\pi) \\ &\leq n + \sum_{u,v} \frac{1}{(\text{index}(u, v) + 1)} \\ &\leq n + 2 \sum_u \sum_{i=1}^{n-1} \frac{1}{i+1} \\ &\leq n + 2n \ln n. \end{aligned}$$

□

Note that the autopartition  $P_\pi$  fails to take advantage of possible free cuts. This is remedied in the obvious way by the following modification. In defining  $P_\pi^*$ , free cuts are made wherever possible. For definiteness, when there are several possible free cuts we choose the one which is earliest according to  $\pi$ . For some applications it may be desirable to minimize the depth of the partition tree, in which case we would choose the free cuts to try to balance the tree as in the previous section.

#### Recursive procedure for $P_\pi^*$

While there is some nontrivial free cut, make that free cut which is earliest in the ordering  $\pi$ ; otherwise make the (non-free) cut which is earliest in the ordering. Then proceed recursively in each component with the autopartition derived from the restriction of  $\pi$  to those facets of  $\Sigma$  which intersect that component.

Although the following theorem can also be derived as a corollary of Theorem 3 and the observation that the use of free cuts does not increase the size of the resulting partition, we present a direct proof as a precursor to the three-dimensional case in the next section.

**Theorem 4.** The expected size of the autopartition  $P_\pi^*$  for  $n$  segments in  $R^2$  when  $\pi$  is chosen uniformly over all  $n!$  possibilities is  $O(n \log n)$ .

**Proof.** For a given segment  $v$ , consider the set of all segments whose extensions can intersect  $v$ , and label these segments as  $u_1, u_2, \dots, u_k$ , based on the order in which the intersections occur on  $v$  from left to right. (See Figure 8 for an example.)

The effect of free cuts can be illustrated in the configuration shown in Figure 8. Suppose that the ordering induced by  $\pi$  is  $u_1, u_3, u_4, u_2, v$ . Then  $v$  is cut by  $u_1, u_3$ , and  $u_4$ , but not by  $u_2$ . As soon as the cuts by  $u_1$  and  $u_3$  are made the subsegment of  $v$  between these cuts is removed by a free cut using  $\text{line}(v)$ . In other words, an intersection of  $v$  with some  $\text{line}(u)$  results in an actual cut in  $P_\pi^*$  only if  $u$ 's intersection point on  $v$  is not sandwiched between two earlier intersections.

Thus, in an autopartition  $P_\pi^*$ ,  $\text{line}(u_i)$  can cut  $v$  only if either  $u_i$  precedes all of  $v, u_1, u_2, \dots, u_{i-1}$  in the ordering  $\pi$ , or  $u_i$  precedes all of  $v, u_{i+1}, u_2, \dots, u_k$ . (Both conditions

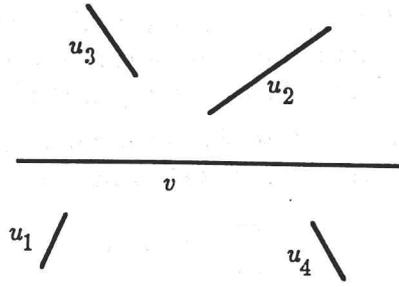


Figure 8.

hold when  $u_i$  is the first of all of  $v, u_1, u_2, \dots, u_k$ , which has a probability of  $\frac{1}{k+1}$ .) Therefore,

$$\text{Prob}(u_i \text{ cuts } v) \leq \frac{1}{i+1} + \frac{1}{k-i+2} - \frac{1}{k+1},$$

and so

$$\begin{aligned} E(P_\pi^*) &\leq n + \sum_v \sum_{i=1}^k \left( \frac{1}{i+1} + \frac{1}{k-i+2} - \frac{1}{k+1} \right) \\ &\leq n + \sum_v \left( 2\left(\frac{1}{2} + \dots + \frac{1}{k+1}\right) - \frac{k}{k+1} \right) \\ &\leq n + 2n \ln n. \end{aligned}$$

□

A specific permutation  $\pi$  which achieves the  $O(n \log n)$  size bound can be easily constructed.

**Theorem 5.** For any  $n$  disjoint line segments in the plane, a complete binary autopartition of size  $O(n \log n)$  can be found in  $O(n^2)$  time.

**Proof.** A permutation  $\pi$  is constructed in reverse order. We first choose  $\pi(n)$  arbitrarily. Now suppose that  $\pi(k+1), \dots, \pi(n)$  have been chosen. For each of  $u_{\pi(k+1)}, \dots, u_{\pi(n)}$ , we find the ordered set of intersection points with the lines through the remaining  $k$  segments. There are at most  $2(n-k)$  extreme intersection points on the segments  $u_{\pi(k+1)}, \dots, u_{\pi(n)}$ , so there is some one of the  $k$  remaining segments,  $u_j$  say, whose line accounts for no more than  $2(n-k)/k$  of these. We choose  $\pi(k) = j$ , and continue in this way until  $\pi$  is complete. Summing the number of cuts, we have:

$$\begin{aligned} \text{size}(P_\pi^*) &\leq n + \sum_{k=1}^n \frac{2(n-k)}{k} \\ &\leq 2n \ln n - n. \end{aligned}$$

A fairly simple  $O(n^2 \log n)$ -time algorithm would find all  $O(n^2)$  line intersections and then sort the intersections which occur on each segment. After this stage the selection and updating required for the construction described above can be easily accomplished in  $O(n)$

time per step. To reduce the total time to  $O(n^2)$  we perform the first stage in the following manner. The *line graph* of the  $n$  lines can be set up in  $O(n^2)$  time [1], and then for each segment, to find the ordered sequence of intersections with the other lines takes only  $O(n)$  time [3].  $\square$

## 5. Partitions of Size $O(n^2)$ in Three Dimensions

We extend the ideas of the previous section to three dimensions. Let  $\Sigma = \{u_1, u_2, \dots, u_n\}$  be  $n$  facets in  $R^3$ . We will consider the expected size of the autopartition  $P_\pi^*$  of  $\Sigma$  when  $\pi$  is a random permutation. Let  $Y_k$  be the number of additional facets created by the  $k^{\text{th}}$  cut of  $P_\pi^*$ , i.e., by the cut plane  $u_{\pi(k)}$  after the cuts  $u_{\pi(1)}, \dots, u_{\pi(k-1)}$  have been made.

**Lemma 1.**  $E(Y_k)$ , the expected size of  $Y_k$ , is  $O(n)$ .

**Proof.** Let  $Y_{k,u}$  be the contribution to  $Y_k$  from facet  $u \in \Sigma$ , i.e.,  $Y_{k,u}$  is the number of extra subfacets created on facet  $u$  by the cut plane  $u_{\pi(k)}$ . We will show that  $E(Y_{k,u}) = O(1)$ . Consider the arrangement  $L_{\pi,k}$  of the set of lines,  $\{\ell_{\pi(1)}, \dots, \ell_{\pi(k)}\}$ , where the line  $\ell_{\pi(i)}$  is the intersection of cut plane  $u_{\pi(i)}$  with facet  $u$  for  $1 \leq i \leq k$ . To calculate  $Y_{k,u}$ , consider the subfacets of  $u$  which are cut by plane  $u_{\pi(k)}$  in  $P_\pi^*$ . In  $P_\pi$ , i.e., without free cuts, these subfacets would correspond exactly to those regions of  $L_{\pi,k-1}$  which are intersected by  $\ell_{\pi(k)}$ . However in  $P_\pi^*$ , any of the regions of  $L_{\pi,k-1}$  which are *internal*, i.e., are bounded entirely by cuts, would have been eliminated earlier by free cuts, so that  $Y_{k,u}$  is just the number of *external* regions intersected by  $\ell_{\pi(k)}$ . For any arrangement  $H$  of  $k$  lines,  $h_1, h_2, \dots, h_k$ , on facet  $u$  and any  $i$ ,  $1 \leq i \leq k$ , define  $x(H, i)$  to be the number of external regions in the line arrangement  $H - \{h_i\}$  that are intersected by  $h_i$ , and denote the average  $\frac{1}{k} \sum_{i=1}^k x(H, i)$  by  $\bar{x}(H)$ . Note that the sum  $\sum_{i=1}^k x(H, i)$  is equal to the total number of edges of those regions in the arrangement  $H$  which are intersected by the boundary of the facet  $u$ . (In Figure 9 the edges bounding these regions are marked by dashed lines.) It is shown in [4] that the number of bounding edges corresponding to any segment, such as side AB, is  $O(k)$ .

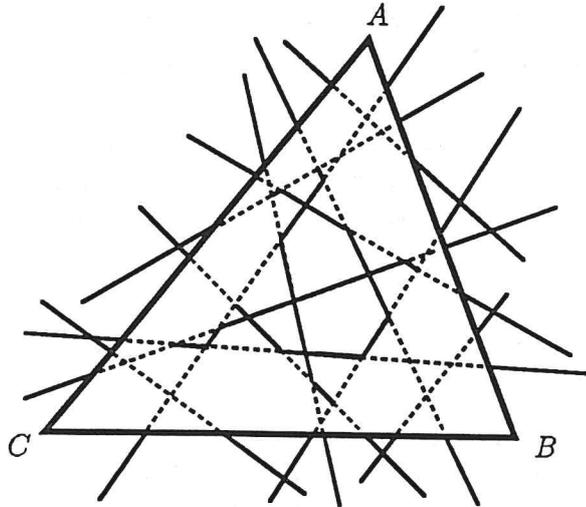


Figure 9.

Thus the sum  $\sum_{i=1}^k x(H, i)$  is bounded above by  $O(k)$ , hence  $\bar{x}(H) \leq O(1)$ . Now,

$$\begin{aligned} E(Y_{k,u}) &= \frac{1}{n!} \sum_{\pi} x(L_{\pi,k}, \pi(k)) \\ &= \frac{1}{n!} \sum_{\pi} \bar{x}(L_{\pi,k}) \\ &= O(1). \end{aligned}$$

Thus  $E(Y_k) = O(n)$ . □

**Theorem 6.** The expected size of the autopartition  $P_{\pi}^*$  for  $n$  facets in  $R^3$  when  $\pi$  is chosen uniformly is  $O(n^2)$ . There are sets of  $n$  facets in  $R^3$  for which the size of every autopartition is  $\Omega(n^2)$ .

**Proof.** From Lemma 1, it follows that the total number of facets created by  $P_{\pi}^*$  is  $\sum_{k=1}^n E(Y_k) = O(n^2)$ . Example 2 yields the lower bound. □

**Theorem 7.** An autopartition  $P_{\pi}^*$  of size  $O(n^2)$  for  $n$  facets in  $R^3$  can be constructed in  $O(n^3)$  time.

**Proof.** The existence of such an autopartition is immediate from Theorem 6. By Lemma 1, the following iterative procedure generates an appropriate ordering,  $\pi$ , in reverse order.

**Step 1.** First construct the line graph of the arrangement on each facet.

**Step 2.** Trace the boundary regions on each facet and find which line generates the smallest total number of bounding edges. Choose this for the final cut and remove this element from the arrangement.

**Step 3.** Repeat Step 2 to find the cutting sequence in reverse order.

One can show that Step 1 takes time  $O(n^3)$  (see [2], [4]), while each iteration of Step 2 takes  $O(n)$  time (see [4]). □

The previous theorem is easily generalised to higher dimensions.

**Theorem 8.** An autopartition  $P_{\pi}^*$  of size  $O(n^{k-1})$  for  $n$  facets in  $R^k$  can be constructed in polynomial time.

**Proof.** The proof is analogous to that of Theorem 7. It is proved in [4] that, for any  $k \geq 2$  and for any arrangement of  $n$  hyperplanes in  $R^k$ , the total number of boundary hyperplanes of all regions which are intersected by some other given hyperplane is  $O(n^{k-1})$ . □

## 6. A Lower Bound

We present a lower bound on the partition complexity in three dimensions which is due to an example by Thurston [10].

**Example 3** For any  $m \geq 1$ , let the collection  $C_m$  consist of three mutually orthogonal families of  $m^2$  line segments each defined as follows. Let  $\delta$ ,  $0 < \delta < \frac{1}{6}$ , be fixed and, for all  $p, q, r$ , with  $1 \leq p, q, r \leq m$ , let  $e_{*qr}$  be the line segment defined by:

$$0 \leq x_1 \leq m + 1, x_2 = q - \delta, x_3 = r + \delta.$$

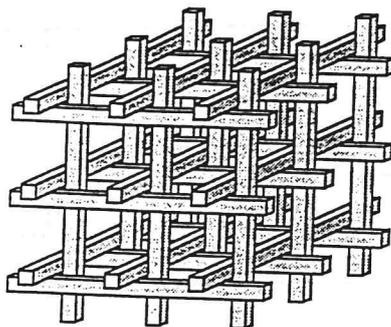


Figure 10.

Similarly we define:

$$e_{p^*r} = \{x_1 = p + \delta, 0 \leq x_2 \leq m + 1, x_3 = r - \delta\},$$

$$e_{pq^*} = \{x_1 = p - \delta, x_2 = q + \delta, 0 \leq x_3 \leq m + 1\}.$$

Then,  $C_m = \{e_{*qr}, e_{p^*r}, e_{pq^*} | 1 \leq p, q, r \leq m\}$ , and  $|C_m| = 3m^2$ .

A sketch of this configuration with  $m = 3$  is shown in Figure 10. For clarity the line segments are represented as solid rods.

**Theorem 9.** In  $R^3$ ,  $p(n) = \Omega(n^{\frac{3}{2}})$ .

**Proof.** Observe that in Example 3, for any  $p, q, r$ , where  $1 \leq p, q, r \leq m$ , the line segments  $e_{*qr}, e_{p^*r}, e_{pq^*}$  come close to the grid point  $(p, q, r)$  and from some viewpoint in the positive orthant cyclically overlap each other. This is sufficient to ensure that any binary partition for  $C_m$  must cut one of these three segments in the neighborhood of  $(p, q, r)$ . The total number of subsegments generated is therefore at least  $3m^2 + m^3$ , and our lower bound is proved.  $\square$

We note that the above configuration uses only orthogonal facets.

## 7. Optimal Orthogonal Partitions

In this section we consider the case where  $\Sigma$  consists of  $n$  horizontal or vertical segments in the plane, and show that an  $O(n)$  partition for  $\Sigma$  can be found.

Let  $R = \{(x, y) | x_0 \leq x \leq x_1, y_0 \leq y \leq y_1\}$  be a bounding rectangle for  $\Sigma$ , that is, all segments of  $\Sigma$  lie within  $R$ . A segment  $s$  is said to be *anchored* on a side of  $R$  if one of the endpoints of  $s$  lies on that side and the other endpoint in the interior of  $R$ . Let  $\mathcal{H}_0$  and  $\mathcal{H}_1$  denote the sets of horizontal segments anchored on the sides  $(x = x_0)$  and  $(x = x_1)$  respectively. Define the sets,  $\mathcal{V}_0$  and  $\mathcal{V}_1$ , of vertical segments similarly.

We define a *T-decomposition* of  $R$  into  $R_1 \cup R_2 \cup R_3$  as follows. Let  $s_0$  be a longest segment in  $\mathcal{H}_0$ , and suppose  $\text{line}(s)$  intersects some segment of  $\mathcal{V}_0 \cup \mathcal{V}_1$ . Let  $t$  be the first (i.e., leftmost) such segment. We decompose  $R$  into three rectangles,  $R_1, R_2$ , and  $R_3$ , by first cutting  $R$  along  $\text{line}(t)$ , and then splitting the area to the left of  $t$  along  $\text{line}(s)$ . (See Figure 11.) The following fact is easy to verify.

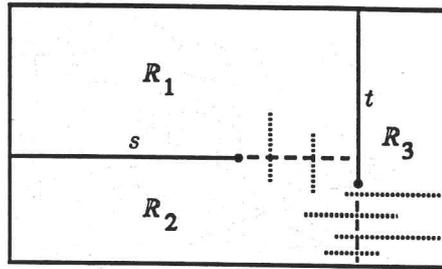


Figure 11.

**Fact.** Among the anchored segments of  $\Sigma$ , only those in  $\mathcal{H}_1$  may be cut (and cut only once) in a T-decomposition.

We shall define a partition for  $\Sigma$  by recursively applying T-decompositions. Furthermore, we will rotate  $R_1$ ,  $R_2$  and  $R_3$  suitably in the recursion so as to ensure that each segment of  $\Sigma$  is cut at most a constant number of times. To this end, we attach a label of 'green' or 'red' to each anchored segment during the course of the partition to represent the status of that segment.

Any initially anchored segment is *green*. After the first cut of an unanchored segment, each of the two resulting anchored segments is made *green*. When a *green* segment is cut again, the part between the first and the second cut is no longer anchored while the remaining part is made *red*. (Figure 12.) We shall ensure that no *red* segment is ever cut.

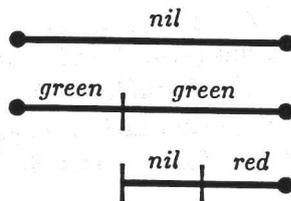


Figure 12.

A side of a rectangle  $R$  is considered *green* if all segments anchored on that side are *green*; otherwise the side is *red*.

**Lemma 2.**

- (i) After a cut which does not divide any anchored edge, each resulting rectangle has a *green* edge.
- (ii) If the side ( $x = x_1$ ) of  $R$  is *green*, then after a T-decomposition, each of  $R_1$ ,  $R_2$  and  $R_3$  has at least one side which is *green*.

**Proof.**

- (i) The cut edges are *green*.
- (ii) The sides of  $R_1$  and  $R_2$  defined by line( $s$ ) are each *green*. The sides of  $R_3$  defined by line( $t$ ) and by the side ( $x = x_1$ ) are both *green*.

□

Note that in Figure 12 the side of  $R_2$  defined by  $\text{line}(t)$  may be *red*.

We describe our partition algorithm recursively.

### Orthogonal Partition Algorithm (OPA)

Precondition:  $R$  is a bounding rectangle for the set of segments and  $R$  has a *green* side.

- (i) If  $R$  is empty then we are finished.
- (ii) If there is some segment  $s$  such that  $\text{line}(s)$  cuts no anchored segment, then cut  $R$  along  $\text{line}(s)$ , and recurse on the two resulting rectangles.
- (iii) Otherwise, re-orient  $R$  if necessary so that its right side is *green* and apply a T-decomposition. Recurse on the three resulting rectangles.

**Theorem 10.** Algorithm OPA finds an  $O(n)$  partition.

**Proof.** By Lemma 2, the invariant that  $R$  has a *green* side is preserved. Since any original segment can form at most two *green* segments, each of which can be cut only once more, the size of such a partition is at most  $4n$ . □

## 8. Applications

We describe how complete binary partitions can be applied to give  $O(n^2)$  solutions to the two problems mentioned in the Introduction.

### Hidden-Surface Removal

To speed up hidden-surface removal when a 3-D scene is viewed from different positions, Fuchs, Kedem and Naylor [5] proposed preprocessing the scene into a *binary-space-partition* (BSP) tree. Indeed, a BSP as defined in [5] corresponds to what we call an ‘autopartition’ in the present paper, and the fact that finding efficient BSP’s for general 3-D scenes remained an open problem served as our initial motivation for studying binary partitions.

We first describe the relation between visibility computation and complete binary partitions (as observed for BSP in [5]), and then state the new result.

**Definition.** Let  $\Sigma = \{u_1, u_2, \dots, u_n\}$  be  $n$  facets in  $R^3$ , and  $w \in R^3$  a viewpoint. A permutation  $\pi$  of  $\{1, 2, \dots, n\}$  is said to be a *visibility ordering* of  $\Sigma$  with respect to  $w$ , if for any  $i, j$  with  $\pi(i) \leq \pi(j)$  and any point  $q \in u_{\pi(j)}$ , we have  $\ell(w, q) \cap u_{\pi(i)} = \emptyset$  where  $\ell(w, q)$  is the line segment connecting  $w$  and  $q$ , i.e., facet  $u_{\pi(i)}$  cannot obstruct the view of  $u_{\pi(j)}$  from  $w$ .

Hidden-surface removal can be accomplished by using a visibility ordering to drive a ‘painter’s algorithm’ which paints each facet in low-to-high order onto the screen’s image buffer. Note that visibility ordering is a function of the viewing position; also such an ordering may not always exist, as one can easily find an example where three facets block each other cyclically (cf. Figure 10). However, if a complete binary partition of the input is made then the resulting set of facets always permits a visibility ordering for any viewing position  $w$ . Furthermore, the desired ordering can be computed quickly for any given  $w$  via a tree traversal.

**Definition.** Let  $P$  be a binary partition tree of  $\Sigma$ . For any point  $w \in R^3$ , an *in-order traversal of  $P$  with respect to  $w$*  is an (otherwise conventional) in-order traversal where at each internal node  $v$  with cut plane  $H_v$ , the half-space of  $H_v$  containing  $w$  is visited *after* the half-space not containing  $w$ . (In the case that  $w$  lies on  $H_v$ , either half-space may be visited first.) Let  $\Sigma'$  denote the set of facet fragments produced by  $P$ , that is,  $\Sigma'$  is the union of  $R(v) \cap H_v \cap \Sigma$  over all internal nodes  $v$  of  $P$ .

**Lemma 3.** Let  $P$  be a complete binary partition of  $\Sigma$  with output  $\Sigma'$ . A visibility ordering of  $\Sigma'$  with respect to any viewing position  $w$  can be generated in time  $O(|P|)$  via an in-order traversal of  $P$  with respect to  $w$ .

**Proof.** If  $w$  lies in a half-space  $H^+$ , then no facet fragments which lie completely in  $H^-$  can obstruct any facet fragment lying completely in  $H^+$ . This justifies assigning larger visibility numbers to facets in  $H^+$  than to those in  $H^-$ . The time required for the tree traversal is clearly  $O(|P|)$ .  $\square$

Thus, in applying the scheme of [5] to solve hidden-surface removal for real-time graphics systems, both the storage space and the tree traversal time are proportional to the size of the partition tree. Previously, only an  $O(n^3)$  upper bound on tree size was known [5]. The following new result is a direct consequence of Theorem 7 and Example 2.

**Theorem 11.** For inputs of size  $n$ , a BSP tree of size  $O(n^2)$  can be constructed and there is a lower bound of  $\Omega(n^2)$ .  $\square$

Constructive Solid Geometry.

Another application of binary partitions is in generating a constructive-solid-geometry (CSG) representation of an object from its boundary representation. For polyhedral objects, Peterson [9] considered CSG formulae where the literals are half-spaces supporting the faces of the polyhedron and the operations are intersection and union; we call such a formula a *Peterson-style formula*. A natural question is: given a polyhedron described by its  $n$  faces, can one generate a short Peterson-style formula? In two dimensions, it is known that a formula of size  $O(n)$  can be found for a simple polygon of  $n$  sides (Dobkin et al. [3]; also see [9]). In three dimensions, it remained an open problem (see [3]) whether the straightforward  $O(n^3)$  bound on formula size could be improved.

We observe that a complete binary autopartition  $P$  for the facets of a polyhedron  $D$  naturally leads to a Peterson-style formula  $f(D)$  of size  $O(|P|)$  for the polyhedron. If  $D$  is a half-space, then  $f(D)$  is a single literal. Recursively, if  $D$  is divided into two parts by a cut plane  $H$  (corresponding to some facet of  $D$ ) at an internal node  $v$  of  $P$ , then let  $f_v(D) = (f_1 \cap H^+) \cup (f_2 \cap H^-)$ , where  $f_1$  and  $f_2$  are the formulae corresponding to the two subtrees of  $v$ . Thus our main theorem implies the following result.

**Theorem 12.** Every polyhedron in  $R^3$  with  $n$  facets has a Peterson-style CSG formula of size  $O(n^2)$ .  $\square$

## 9. Further Work and Open Problems

We have concentrated mostly on the size of the partitions constructed, since this governs the time for hidden-surface computation with a moving viewpoint. More work is needed to develop efficient algorithms for constructing small partitions in special cases, and, in some

applications, probabilistic algorithms may be useful.

Several important questions remain open. In particular, we have shown that in three dimensions  $\Omega(n^{\frac{3}{2}}) \leq p(n) \leq O(n^2)$ , and this gap is provoking. In particular, for orthogonal line segments in three dimensions the lower bound of  $\Omega(n^{\frac{3}{2}})$  still holds, since Example 3 is of this form, but we would hope to improve the upper bound in this restricted case. Note that the lower bound has not been proved for the CSG application, so that here the gap is wider.

In two dimensions the bounds are given by:  $\Omega(n) \leq p(n) \leq O(n \log n)$ . Progress could be made in relaxing the orthogonality constraint for our construction or in finding linear-size partitions for other special situations.

**Conjecture.** In  $R^2$ ,  $p(n) = O(n)$ .

### Acknowledgement

We thank Jack Snoeyink for bringing to our attention the application to constructive solid geometry, and also Marshall Bern and Dan Greene for helpful discussions.

### References

- [1] C. Chazelle, Intersecting is easier than sorting, *Proc. 16th Ann. ACM Sympos. on Theory of Computing*, 1983, 125-134.
- [2] B. Chazelle, L. Guibas and D. Lee, The power of geometric duality, *BIT*, 25, 1985, 76-90.
- [3] D. Dobkin, L. Guibas, J. Hershberger and J. Snoeyink, An efficient algorithm for finding the CSG representation of a simple polygon, *Computer Graphics*, 22, 4, 1988, 31-40.
- [4] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987.
- [5] H. Fuchs, Z. Kedem and B. Naylor, On visible surface generation by a priori tree structures, *Computer Graphics (SIGGRAPH '80 Conference Proceedings)*, 124-133.
- [6] E. Gilbert and E. Moore, Variable-length binary encoding, *Bell System Tech. J.*, 38, 1959, 933-968.
- [7] L. Guibas and F. Yao, On translating a set of rectangles, *Advances in Computing Research*, Vol.1, JAI Press, 1983, 61-77.
- [8] D. Knuth, *The Art of Computer Programming, Vol.3: Sorting and Searching*, Addison-Wesley, 1973.
- [9] D. Peterson, Halfspace representations of extrusions, solids of revolution, and pyramids, SANDIA Report SAND84-0572, Sandia National Laboratories, 1984.
- [10] W. Thurston, private communication.

