Discrete Comput Geom 5:485-503 (1990)

Discrete & Computational
Geometry
© 1990 Springer-Verlag New York Inc.

Efficient Binary Space Partitions for Hidden-Surface Removal and Solid Modeling*

Michael S. Paterson[1] and F. Frances Yao[2]

[1] Department of Computer Science, University of Warwick, Coventry, CV4 7AL, England

[2] Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, USA

Abstract. We consider schemes for recursively dividing a set of geometric objects by hyperplanes until all objects are separated. Such a *binary space partition*, or BSP, is naturally considered as a binary tree where each internal node corresponds to a division. The goal is to choose the hyperplanes properly so that the size of the BSP, i.e., the number of resulting fragments of the objects, is minimized. For the two-dimensional case, we construct BSPs of size $O(n \log n)$ for n edges, while in three dimensions, we obtain BSPs of size $O(n^2)$ for n planar facets and prove a matching lower bound of $\Omega(n^2)$. Two applications of efficient BSPs are given. The first is an $O(n^2)$-sized data structure for implementing a hidden-surface removal scheme of Fuchs *et al.* [6]. The second application is in solid modeling: 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. The best previous results for both of these problems were $O(n^3)$.

1. Introduction

Recursive partitioning is a basic problem-solving technique which has proven to be most 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

* This research was done while M. S. Paterson was visiting the Xerox Palo Alto Research Center. This author is supported by a Senior Fellowship of the SERC and by the ESPRIT II BRA Program of the EC under Contract 3075 (ALCOM).

linear cut of the input, that is, by splitting the objects along a line (in the two-dimensional case) or along a plane (in the three-dimensional 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, we are motivated to choose the dividing cuts carefully so that fragmentation of the input objects is kept to a minimum. The recursive partition mentioned above was first considered by Fuchs *et al.* [6], and is called a *binary space partition* (or BSP).

We are interested in the question of constructing BSP trees whose size is not too large as a function of the original input size. In the three-dimensional formulation of the problem, we take the input to consist of a set of n nonintersecting convex polygons in R^3, since polygonal tiling is common for representing surfaces in space. Let $p(n)$ be the maximum value over all inputs of cardinality n of the size of a minimal BSP tree. (Precise definitions of binary space 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 $p(n) = \Omega(n^2)$ follows from an example due to Eppstein [5].

In this paper we prove upper and lower bounds for BSPs in the general case. Better bounds can be obtained in several important special cases. In [14] we consider BSPs for orthogonal sets of elements and derive exact bounds of $\Theta(n)$ and $\Theta(n^{3/2})$ for the two- and three-dimensional cases, respectively.

As applications of efficient BSPs, 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, it is necessary to generate rapidly images of a three-dimensional scene as viewed from changing positions. A good strategy is to preprocess the scene suitably so as to simplify the hidden-surface computation at runtime. In [6] it was observed that if the objects comprising the scene are represented as a BSP 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 $O(n^3)$ [6]. As a direct consequence of our main theorem, this bound can be improved to $O(n^2)$.

The second application of BSPs 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. In contrast, 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 modeling to be able to convert efficiently between the two styles of representation. In a special type of CSG representation considered by Peterson [15], 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 a short

Peterson-style formula be generated? A straightforward upper bound on the size of the formula is $O(n^3)$. We will show that our result on BSPs implies an $O(n^2)$ bound on formula size.

Other applications of BSPs include point location, and the convex decomposition of polygons and polyhedra. Previous work on BSP trees and their uses in graphics applications can be found in [11] and [17].

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 $\Omega(n^2)$ lower bound of Section 6. The applications are discussed in Section 7, and we conclude with some comments and open problems in Section 8.

A preliminary version of this paper appeared in [13].

2. Preliminaries

In this section we give the mathematical formulation of the partitioning problem, and discuss some basic properties of BSPs.

A d-dimensional *binary partition* P is a recursive partition of d-dimensional Euclidean space, R^d, defined by a set of hyperplanes. Let $\mathscr{H}$ be a collection of (oriented) hyperplanes that are organized as a binary tree and labeled accordingly as $H_\lambda, H_0, H_1, H_{00}, H_{01}, \ldots$ (see Fig. 1). Then $\mathscr{H}$ defines a binary partition P under which R^d is first partitioned by the root hyperplane H_λ into two open half-spaces, H_λ^-, H_λ^+, and H_λ itself. Recursively, H_λ^- and H_λ^+ are partitioned by the subtrees rooted at H_0 and H_1, respectively. We refer to the hyperplanes $H_i \in \mathscr{H}$, $i \in \{0, 1\}^*$, 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 (proper) 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.

Let Γ be a collection of *facets*, i.e., convex polytopes of dimension $(d-1)$ or less, in R^d. One-dimensional facets are line segments and two-dimensional facets are

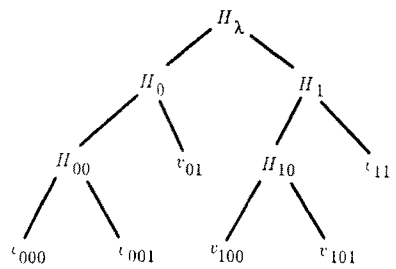

Fig. 1

convex polygons. A binary partition P naturally induces a decomposition of Γ. For any node v of P, let $\Gamma(v)$ denote the collection of subfacets, $\Gamma \cap R(v)$. For a given Γ, we are interested in binary partitions P of R^d with the property that, at each leaf v, the set $\Gamma(v)$ is empty; we refer to such a P as a *BSP* of Γ. We define the *weight* of an internal node v to be the number of subfacets of $\Gamma(v)$ that lie within H_v. The *size*, $|P|$, of a BSP of Γ is the total weight of its internal nodes, which is also the total number of subfacets generated by P. The *partition complexity* of Γ, denoted by $p(\Gamma)$, is $\min\{|P| \mid P \text{ is a BSP of } \Gamma\}$. Define $p(n) = \max\{p(\Gamma) \mid |\Gamma| = n\}$.

Note that if Γ has no co(hyper)planar facets, then the weight of an internal node in a BSP is always one or zero. In any case, for simplicity of presentation, we are not concerned with further partitioning of the subfacets at internal nodes of the tree. The reduction in dimension allows a simple recursive structure and we concentrate on the broader complexity issues presented in the top dimension.

We could alternatively define the size of a BSP to be the number of leaves of the tree, i.e., the number of convex regions of the partition space. This number is at least $|P| + 1$, assuming that Γ has no coplanar facets, and is at most $2|P|$ if trivial cuts are avoided so that each leaf region contains at least one subfacet in its boundary. The measure $|P|$ that we have chosen seems the most convenient to work with.

Figure 2 shows a BSP where Γ consists of six edges forming a close polygon. The BSP has size 8, and decomposes the interior of the polygon into three convex regions, $\alpha_1, \alpha_2, \alpha_3$. For each cut line the positive half-plane is the upper half-plane in the diagram.

The *description size* of a facet or set of facets is the total number of boundary elements of all dimensions for these facets. To simplify our treatment of BSPs we assume a fixed bound on the description size of each initial facet. In particular, in R^3 the input facets are polygons with a bounded number of edges. Thus, for all Γ, the description size of Γ is $O(|\Gamma|)$. We naturally define the *description size* of a BSP

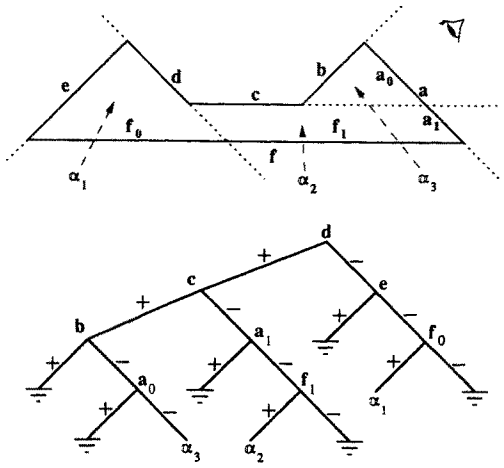

Fig. 2

to be the description size of the corresponding set of subfacets. In R^3 it is easy to bound the description size of a BSP.

Lemma 1. *Let P be a BSP of Γ in R^3.*

$$(\textit{description size of } P) = O((\textit{description size of } \Gamma) + |P|) = O(|P|).$$

Proof. Each division of a subfacet into two introduces four new edges: two by dividing two existing edges and two from the new boundary edges created. Hence

$$\text{number of edges of } P = \text{number of edges of } \Gamma + 4(|P| - |\Gamma|).$$

The lemma follows from this and the bound on the description size of Γ given above. □

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 use the terms *line*(A) and *plane*(A), respectively.

3. Autopartitions

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

Note that a minimum partition for Γ is not always achievable by an autopartition.

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

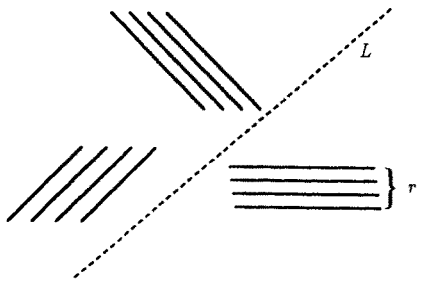

Fig. 3

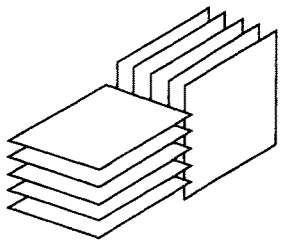

Fig. 4

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

Example 2. Consider a configuration with two sets of r parallel squares as illustrated in Fig. 4. In this case every autopartition has size exactly $r^2 + 2r$, but there is a BSP which begins with a plane separating the two sets and has size only $2r$.

The above example generalizes readily to d dimensions.

Theorem 1. *For any $d > 2$, there is a configuration of size n in R^d which has a BSP of size n but for which every autopartition is of size $\Omega(n^{d-1})$.*

Proof. We define $(d-1)$ families of facets where, for $i = 1, \ldots, d-1$, family $F_i = \{(x_i = j, 2i - 1 \leq x_d \leq 2i) | j = 1, \ldots, r_i\}$ and $n = \sum_{i=1}^{d-1} r_i$. Since these families are easily separable from each other by $(d-1)$ hyperplanes orthogonal to the x_d-axis, there is a BSP of size n. We can prove by induction on the number of facets that the size of any autopartition is $\prod_{i=1}^{d-1}(r_i + 1) - 1$. The first hyperplane of any autopartition, $x_k = s$ say, decomposes the configuration into two smaller subconfigurations of the same type and

$$1 + \left(s \prod_{i \neq k} (r_i + 1) - 1\right) + \left((r_k - s + 1) \prod_{i \neq k} (r_i + 1) - 1\right) = \prod_i (r_i + 1) - 1. \quad \square$$

There is a simple yet useful device which could prevent excessive fragmentation of Γ during a partition. If a hyperplane H can partition Γ nontrivially without dividing any facet of Γ, then, obviously, H can be used as a cut hyperplane in an optimal partition. The cut by H is referred to as a *free cut.*

One particular type of free cut presents itself naturally in the course of 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. (See Fig. 5.) Then A divides S into two regions, S_0 and S_1, and any other segment of $\Gamma \cap S$ must lie completely within one of these regions. In such a situation, an immediate partition of S along A is advantageous, since the cut is free and it prevents the

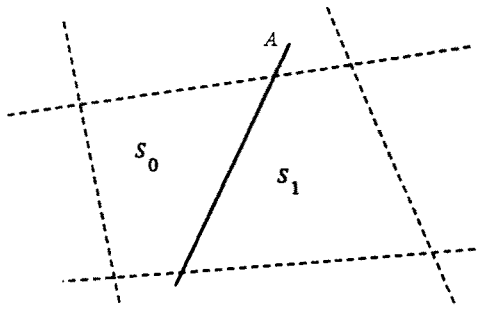

Fig. 5

segment $A \cap S$ from ever being cut. More generally, in higher dimensions, the above observation holds for any region which is completely separated by some facet A. We term the free cut defined by hyperplane(A) in such a situation a *bounded cut*.

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

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

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

Theorem 2. *For any n disjoint line segments in the plane, there is a BSP of size $O(n \log n)$, i.e., $p(n) = O(n \log n)$.*

Proof. Let Γ be a set of input line segments, and let V be the set of endpoints of Γ. Our algorithm initially finds two points $p_{\min}$ and $p_{\max}$ of V with minimum and maximum y-coordinates. Thus all segments of Γ 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 Γ into two sets of segments Γ_a and Γ_b, which lie above and below H_0, respectively. Let $A_1, A_2, \ldots, A_s$ be the segments in Γ_a which intersect both $H_{\max}$ and H_0. Bounded cuts using these segments divide the strip between $H_{\max}$ and H_0 into $s + 1$ regions ordered from left to right. We use α_i, for $0 \le i \le s$, to denote the region that lies between A_i and A_{i+1}. Similarly, if $B_1, B_2, \ldots, B_t$ are the segments that intersect both $H_{\min}$ and H_0, then bounded cuts with these segments separate Γ_b into $t + 1$ disjoint regions ordered from left to right, denoted by β_j, for $0 \le j \le t$. (See Fig. 6.) We then repeat the partitioning for each of the α_i and β_j separately.

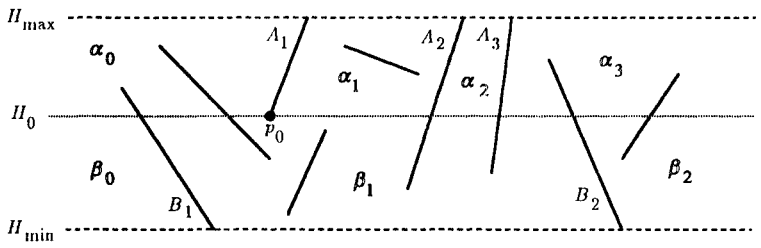

Fig. 6

The partition scheme given above can be represented by a multiway tree. (See Fig. 7.) This tree eventually needs to be transformed into a BSP but the analysis is based on the multiway tree structure.

We show that in the partition defined above, each original segment of Γ is cut into at most $O(\log n)$ pieces. In any region $R(v)$, let $m(v)$ be the number of endpoints of Γ that are in the interior of $R(v)$. Then, since H_0 is a median divider, each of the regions α_i, $0 \le i \le s$, and β_j, $0 \le j \le t$, has $m \le m(v)/2$. Thus the multiway partition tree has depth at most $\log_2 n$. If segment ℓ of Γ 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 ℓ is eliminated by a bounded cut. Hence ℓ is cut into at most $2 \log_2 n$ fragments. □

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

Although our major concern is to minimize the size of the partition, the construction time and, in some applications, the depth of the corresponding tree may also be important. Both concerns are addressed in the following strengthening of Theorem 2.[1]

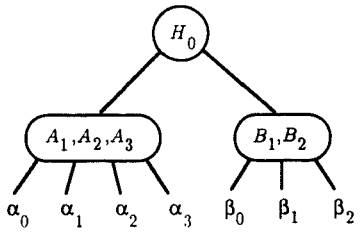

Fig. 7

[1] A referee has pointed out that our construction is similar to that used by Preparata [16] for planar point location.

Theorem 3. *There is an $O(n \log n)$-time algorithm which, for n disjoint line segments in the plane, produces a complete BSP 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 2, we can achieve logarithmic depth by constructing a balanced BSP after applying bounded cuts to some horizontal strip. Suppose the number of endpoints in α_i is m_i, then using the balancing algorithm due to Gilbert and Moore [7] (see Section 6.2.2 of [10]) on the sequence of weights $m_1, m_2, \ldots$, the depth of α_i in the binary expansion of the multiway branch is at most $2 + \log_2(\Sigma m_j) - \log_2(m_i)$. 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 $O(\Sigma_i(\log(\Sigma m_j) - \log(m_i)))$, and contributes a total of only $O(n \log n)$ to the running time.

The entire 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 bounded 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 [8], is defined by "u *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 [8].

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 bounded 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. □

5. Probabilistic Methods for Planar Partitions

In this section we begin by presenting two randomized algorithms for planar partitions; while the first is somewhat simpler, the second generalizes more readily to higher dimensions. For each algorithm an $O(n \log n)$-size bound will be proved using probabilistic arguments. We also give a corresponding deterministic algorithm for finding an $O(n \log n)$ autopartition.

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

Let the facets in Γ be denoted by $\{u_1, u_2, \ldots, u_n\}$, and let π be a permutation of $\{1, 2, \ldots, n\}$. A unique BSP based on π is obtained by the following construction.

Procedure for P_π
For $k = 1$ **to** n **do**
Stage k: For each region intersected by u_k, make a cut with hyperplane(u_k).

The resulting partition is the *autopartition with respect to* π, written as P_π.

The size of the autopartition P_π on n line segments in the plane for a random choice of π is shown below to be $O(n \log n)$.

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

Proof. For line segments u, v, we define *index*$(u, v) = i$ if line(u) intersects $i - 1$ other segments before hitting v, and *index*$(u, v) = \infty$ if line(u) $\cap\, v = \varnothing$. Since a segment u can be extended in two directions, we may have index$(u, v) = i$ for two different v's. (See Fig. 8 where index$(u, v) = 3$.) We say for short that "*u cuts v*" when line(u) divides segment v in the partition.

First we show that the probability that u cuts v in P_π for a random π is at most $1/(\text{index}(u, v) + 1)$. Assume index$(u, v) = i$, and let $u_1, u_2, \ldots, u_{i-1}$ be the segments that 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 $\{u_1, u_2, \ldots, u_{i-1}, v\}$ 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_π satisfies

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

□

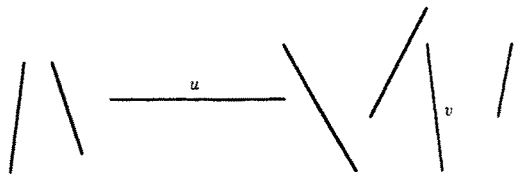

Fig. 8

Note that the autopartition P_π did not make use of possible free cuts. In the following alternative partition, P_π^*, bounded cuts are made wherever possible. For definiteness, when there are several possible bounded cuts we choose the one which is earliest according to π.

Procedure for P_π^*
For $k = 1$ **to** n **do**
Stage k: For each nonempty region, make a cut with hyperplane(u_k).
While there is some bounded cut, make that bounded cut which is earliest in the ordering π.

In comparison with P_π, although P_π^* benefits from bounded cuts, it also allows a region to be cut by hyperplane(u) even when u does not intersect the region.

We analyze P_π^* in the two-dimensional case as a precursor to the three-dimensional case in the next section.

Theorem 5. *The expected size of the autopartition P_π^* for n segments in R^2 when π 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 $u_1, u_2, \ldots, u_k$, based on the order in which the intersections occur on v from left to right.

The effect of bounded cuts can be illustrated in the configuration shown in Fig. 9. Suppose that the ordering induced by π 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 bounded cut using line(v). In other words, an intersection of v with some line(u) results in an actual cut in P_π^* only if u's intersection point on v is not sandwiched between two earlier intersections.

Thus, in an autopartition P_π^*, u_i can cut v only if either u_i precedes all of $v, u_1, u_2, \ldots, u_{i-1}$ in the ordering π, or u_i precedes all of $v, u_{i+1}, u_2, \ldots, u_k$. (Both conditions hold when u_i is the first of all of $v, u_1, u_2, \ldots, u_k$, which has a probability of $1/(k + 1)$.) Therefore,

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

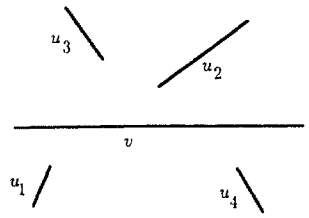

Fig. 9

and so

$$E(P_\pi^*) \le n + \sum_v \sum_{i=1}^{k} \left(\frac{1}{i+1} + \frac{1}{k-i+2} - \frac{1}{k+1} \right)$$

$$\le n + \sum_v \left(2\left(\frac{1}{2} + \cdots + \frac{1}{k+1} \right) - \frac{k}{k+1} \right)$$

$$\le n + 2n \ln n. \qquad \square$$

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

Theorem 6. *For any n disjoint line segments in the plane, an autopartition of size $O(n \log n)$ can be found in $O(n^2)$ time.*

Proof. A permutation π is constructed in reverse order. We first choose $\pi(n)$ arbitrarily. Now suppose that $\pi(k+1), \ldots, \pi(n)$ have been chosen. For each of $u_{\pi(k+1)}, \ldots, u_{\pi(n)}$, 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 $u_{\pi(k+1)}, \ldots, u_{\pi(n)}$, so for one of the k remaining segments, u_j say, its line accounts for no more than $2(n-k)/k$ of these. We choose $\pi(k) = j$, and continue in this way until π is complete. Summing the number of cuts, we have

$$size(P_\pi^*) \le n + \sum_{k=1}^{n} \frac{2(n-k)}{k}$$

$$\le 2n \ln n - n.$$

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 the $O(n^2)$ time [1], and then, for each segment, to find the ordered sequence of intersections with the other lines takes only $O(n)$ time. $\square$

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

The ideas of the previous section can be extended to three dimensions to yield both randomized and deterministic algorithms for constructing efficient BSPs.

Let $\Gamma = \{u_1, \ldots, u_n\}$ be n facets in R^3. We consider the expected size of the autopartition P_π^* of Γ when π is a random permutation. Let Y_k be the number of additional facets created at the kth stage of P_π^*, i.e., by plane$(u_{\pi(k)})$ after the cuts by $u_{\pi(1)}, \ldots, u_{\pi(k-1)}$ have been made.

Lemma 2. *$E(Y_k)$, the expected size of Y_k, is $O(n)$.*

Proof. Let $Y_{k,u}$ be the contribution to Y_k from facet $u \in \Gamma$, i.e., $Y_{k,u}$ is the number of extra subfacets created on facet u by 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, $\{l_{\pi(1)}, \ldots, l_{\pi(k)}\}$, where the line $l_{\pi(i)}$ is the intersection of plane$(u_{\pi(i)})$ with facet u for $1 \le i \le k$. To calculate $Y_{k,u}$, consider the subfacets of u which are cut by plane$(u_{\pi(k)})$ in P_π^*. In P_π, i.e., without bounded cuts, these subfacets would correspond exactly to those regions of $L_{\pi,k-1}$ which are intersected by $l_{\pi(k)}$. However, in P_π^*, 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 bounded cuts, so that $Y_{k,u}$ is just the number of *external* regions intersected by $l_{\pi(k)}$. For any arrangement H of k lines, $h_1, h_2, \ldots, h_k$, on facet u and any i, $1 \le i \le k$, 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 $(1/k)\sum_{i=1}^{k} x(H, i)$ 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 Fig. 10 the edges bounding these regions are marked by dashed lines.) It is known [2], [4] that the number of bounding edges corresponding to any segment, such as side AB, is $O(k)$. (This is the point where the constant bound on the number of edges for each original facet is needed.)

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

$$E(Y_{k,u}) = \frac{1}{n!}\sum_{\pi} x(L_{\pi,k}, \pi(k))$$

$$= \frac{1}{n!}\sum_{\pi} \bar{x}(L_{\pi,k})$$

$$= O(1).$$

Thus $E(Y_k) = O(n)$. □

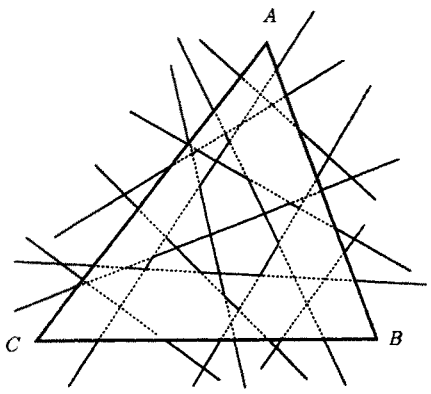

Fig. 10

Theorem 7. *The expected size of the autopartition P_π^* for n facets in R^3 when π 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 2, it follows that the total number of facets created by P_π^* is given by $\sum_{k=1}^{n} E(Y_k) = O(n^2)$. Example 2 yields the lower bound. □

Theorem 8. *An autopartition 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 P_π^* follows from Theorem 7. We must consider both the time required to find a suitable permutation π and the time to construct P_π^*.

By Lemma 2, the following iterative procedure generates an appropriate ordering, π, in reverse order.

> Choose $\pi(n)$ arbitrarily.
> **For** $k = n - 1, \ldots, 2, 1$ **do**
> **Select**: Assume $\pi(k+1), \ldots, \pi(n)$ have been chosen. For each of $u_{\pi(k+1)}, \cdots, u_{\pi(n)}$, find the line arrangement on that facet generated by its intersections with the planes of the remaining k facets. Examine the boundary regions in each arrangement and find which of the unselected k facets generates the smallest total number of bounding edges summed over all the $(n - k)$ arrangements. Choose this for the kth cut.

In each **Select** we have to construct one new line arrangement for $O(n)$ lines and update $O(n)$ other arrangements by the removal of the line corresponding to the most recently chosen facet. Hence each **Select** takes time $O(n^2)$ (see [2] and [4]).

Once the permutation π has been determined, we can construct the BSP tree P_π^* in $O(n^3)$ time as follows. The facets are initially sorted according to π, and this list of facets is associated with the roof node of the BSP tree we will construct. Assume now that we have constructed some initial subtree of the BSP, and at any frontier node v we have a representation of $\Gamma(v)$, the collection of subfacets $\Gamma \cap R(v)$, arranged in two sorted lists. The first list contains the "bounded" subfacets, i.e., those whose boundary contains no part of an original edge from Γ, and the second list contains the other subfacets. Each list is sorted according to π and the first list is regarded as preceding the second.

One step of the algorithm consists of the following procedure. Choose any frontier node v such that $\Gamma(v)$ is nonempty, and suppose that u is the first subfacet associated with v. We set H_v to be plane(u) and proceed to "cut" $\Gamma(v)$ by H_v. In sorted order, each subfacet w of $\Gamma(v)$ is processed in turn, by testing each of its edges or vertices against H_v. If w is contained in H_v it remains associated with v. If w does not intersect H_v, then it is assigned to the appropriate new successor node (v_0 or v_1) of v. If w is cut by H_v, then it is divided into two new subfacets w_0, w_1, which are assigned to v_0 and v_1, respectively. (When u is a bounded facet no such division can

occur.) If w_i is a bounded subfacet it is appended to the first list at v_i, otherwise it is appended to the second list.

The algorithm terminates when, for all frontier nodes v, the set $\Gamma(v)$ is empty.

The running time can be bounded by the depth of the BSP tree multiplied by the description size of P since each edge or vertex is tested at most once at each level of the tree. By Lemma 1, the total time is $O(n \times n^2)$. □

The previous theorem can be generalized to higher dimensions.

Theorem 9. *An autopartition P_π^* of size $O(n^{d-1})$ for n facets in R^d can be constructed in time $O(n^{d+1})$. There are sets of n facets in R^d for which the size of every autopartition is $\Omega(n^{d-1})$.*

Proof. The proof of the size bounds is analogous to that of Theorem 8. We use the following fact, proved in [4], regarding an arrangement A of n hyperplanes in R^d where $d \geq 2$. The total number of boundary hyperplanes summed over all regions of A that are intersected by any other hyperplane is $O(n^{d-1})$. The lower bound is given by Theorem 1.

We have relaxed the time bound to $O(n^{d+1})$ since, for $d > 3$, the facial structure of subfacets is more complex and the simple algorithm for R^3 described above is not adequate. We would therefore maintain the complete arrangement for each facet at each node instead of just keeping the external regions. □

We expect that the time bound in the above theorem can be improved to $O(n^d)$ by the use of a more elaborate data structure.

7. A Lower Bound

Under the restriction to autopartitions, Theorem 9 already gives matching upper and lower bounds for all $d \geq 2$. For the unrestricted case, we present a lower bound on the partition complexity in three dimensions which is due to Eppstein [5].

Example 3. Consider first a planar square grid formed by n parallel red lines segments intersecting n parallel green lines at right angles. Next we skew the square arrangement a little to form part of a hyperbolic paraboloid, with the red and green lines belonging to its two families of generators. Finally, the red lines are all moved "up" very slightly so that the surface containing the red lines is above that containing the green lines (and the lines no longer intersect).

A coordinate representation of this configuration is

$$\{y = j, z = xj | 1 \leq j \leq n\} \cup \{x = i, z = iy + \varepsilon | 1 \leq i \leq n\}$$

and an impression of the configuration with a close-up view of one of the "squares" is given in Fig. 11.

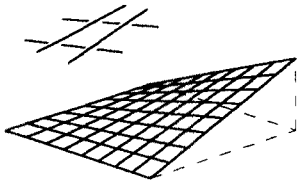

Fig. 11

Theorem 10. *In* R^3, $p(n) = \Omega(n^2)$.

Proof. Consider Example 3. Provided that the separation distance between the two families of lines is sufficiently small, any complete partitioning of the configuration by planes must cut at least one of the four line segments in the neighborhood of each skew square. Hence the total number of cuts made by the partition is $\Omega(n^2)$, and our lower bound is proved. □

An example by Thurston giving a lower bound of $\Omega(n^{3/2})$ for orthogonal line segments in three dimensions is presented in [14].

8. Applications

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

8.1. *Hidden-Surface Removal*

To speed up hidden-surface removal when a three-dimensional scene is viewed from different positions, Fuchs *et al.* [6] proposed preprocessing the scene into a BSP tree. The fact that finding efficient BSPs for general three-dimensional scenes remained an open problem served as our initial motivation for studying BSPs.

We first outline the relation between visibility computation and BSPs as presented in [6], and then state the new result.

Definition. Let $\Gamma = \{u_1, u_2, \ldots, u_n\}$ be n facets in R^3, and let $w \in R^3$ be a viewpoint. A permutation π of $\{1, 2, \ldots, n\}$ is said to be a *visibility ordering* of Γ 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 $\ell(w, q) \cap u_{\pi(i)} = \varnothing$ 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.

A visibility ordering is a prerequisite for many hidden-surface removal algorithms. For example, the "painter's algorithm" paints each facet in low-to-high priority order onto the screen's image buffer, whereas the output-sensitive algorithm of Overmars and Sharir [12] processes the facets in the opposite order. Note that a visibility ordering depends on the viewing position; also such an ordering

may not always exist, as we can easily find an example where three facets block each other cyclically. However, if a BSP of the input is made then the resulting set of subfacets 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 BSP tree of Γ. 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 Γ' denote the set of subfacets produced by P.

Lemma 3. *Let P be a BSP of Γ with output Γ'. A visibility ordering of Γ' 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 subfacet which lies completely in H^- can obstruct any subfacet 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 $O(|P|)$. □

For the BSP illustrated in Figure 2 the visibility ordering generated for the indicated viewpoint is $e, f_0, d, f_1, a_1, c, b, a_0$.

Thus, in applying the scheme of [6] 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 [6]; our results reduce this to $O(n^2)$.

8.2. *Constructive Solid Geometry*

Another application of BSPs is to generate a constructive-solid-geometry (CSG) representation of an object from its boundary representation. For polyhedral objects, Peterson [15] 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 a short Peterson-style formula be generated? In two dimensions, it is known that a formula of size $O(n)$ can be found for a simple polygon of n sides ([3]; also see [15]). 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 an 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 $f_v(D) = (H^+ \cap f_1) \cup (H^- \cap f_2)$, where f_1 and f_2 are the formulae corresponding

to the two subtrees of v. For the polygon shown in Fig. 2 the given BSP yields the formula

$$(d^+ \cap ((c^+ \cap b^- \cap a^-) \cup (c^- \cap a^- \cap f^+))) \cup (d^- \cup e^- \cap f^+).$$

Theorem 8 implies the following result.

Theorem 11. *Every polyhedron in R^3 with n facets has a Peterson-style CSG formula of size $O(n^2)$.*

9. Conclusion and Open Problems

Our principal results are summarized here.

Main Theorem. *The worst-case BSP complexity $p(n)$ is bounded in different dimensions as follows:*

(i) *in R^2, $p(n) = O(n \log n)$,*
(ii) *in R^3, $p(n) = \Theta(n^2)$,*
(iii) *in R^d, for $d \geq 3$, $p(n) = O(n^{d-1})$.*

Several important questions remain open. Example 3 does not extend immediately to higher dimensions, and we have no good lower bounds for $p(n)$ in dimensions greater than three. Also we have no tight lower bound for the CSG application, so here too a gap remains.

The technical requirement of a fixed bound on the number of boundary elements of each facet could be removed if the following question were resolved.

Question. Given an arrangement of n hyperplanes and a convex region C in R^d, what is the total number of bounding facets summed over all those regions of the arrangement which intersect the boundary of C?

It is easy to see that, in two dimensions, the sequence of boundary edges corresponds to a Davenport–Schinzel sequence of order 3, and hence the total number of such edges is at most $O(n\alpha(n))$ (see [4] and [9]).

In two dimensions our bounds for $p(n)$ are given by $\Omega(n) \leq p(n) \leq O(n \log n)$. Progress could be made by extending the construction for orthogonal sets given in [14], or by finding linear-size partitions for other special situations.

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

Acknowledgments

We thank Marshall Bern, Dan Greene, Ketan Mulmuley, and Bruce Naylor for helpful discussions on binary partitions. We are also grateful to Jack Snoeyink for calling our attention to the application to constructive solid geometry.

References

1. B. Chazelle, Intersecting is easier than sorting, *Proc. 16th Ann. ACM Symp. 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**, 1988, 31–40.
4. H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, New York, 1987.
5. D. Eppstein, Private communication.
6. H. Fuchs, Z. Kedem, and B. Naylor, On visible surface generation by a priori tree structures, *Computer Grahics* (SIGGRAPH '80 Conference Proceedings), 1980, 124–133.
7. E. Gilbert and E. Moore, Variable-length binary encoding, *Bell System Technical Journal* **38**, 1959, 933–968.
8. L. Guibas and F. Yao, On translating a set of rectangles, in *Advances in Computing Research*, Vol. 1, edited by F. Preparata, JAI Press, Greenwich, CT, 1983, 61–77.
9. S. Hart and M. Sharir, Nonlinearity of Davenport–Schinzel sequences and of a generalized path compression scheme, *Combinatorica* **6**, 1986, 151–177.
10. D. Knuth, *The Art of Computer Programming*. Vol. 3, Addison-Wesley, Reading, MA, 1973.
11. B. Naylor, *A priori* based techniques for determining visibility priority for 3-d scenes, Ph.D. dissertation, Univ. of Texas at Dallas, 1981.
12. M. Overmars and M. Sharir, Output-sensitive hidden surface removal, *Proc. 30th IEEE Symp. on Foundations of Computer Science*, 1989, 598–603.
13. M. Paterson and F. Yao, Optimal binary partitions with applications to hidden-surface removal and solid modelling, *Proc. 5th Ann. ACM Symp. on Computational Geometry*, 1989, 23–32 (also Dept. of Computer Science Research Report RR139, Univ. of Warwick, March 1989).
14. M. Paterson and F. Yao, Binary space partitions for orthogonal objects, *Proc. 1st Annual ACM–SIAM Symp. on Discrete Algorithms*, 1990, 100–106.
15. D. Peterson, Halfspace representations of extrusions, solids of revolution, and pyramids, SANDIA Report SAND84-0572, Sandia National Laboratories, 1984.
16. F. Preparata, A new approach to planar point location, *SIAM Journal on Computing* **10**, 1981, 473–482.
17. W. Thibault and B. Naylor, Set operations on polyhedra using binary space partitioning trees, *Computer Graphics* **21**, 1987, 153–162.

Received June 1, 1989, and in revised form March 19, 1990.