# Research report 158

## OPTIMAL BINARY SPACE PARTITIONS FOR ORTHOGONAL OBJECTS

Michael S Paterson, F Frances Yao*

(RR158)

A *binary space partition*, or BSP, is a scheme for recursively dividing a configuration of objects by hyperplanes until all objects are separated. BSPs are widely used in computer graphics as the underlying data structure for computations such as real-time hidden-surface removal, ray tracing, and solid modelling. In these applications, the computational cost is directly related to the *size* of the BSP, i.e., the total number of fragments of the objects generated by the partition. Until recently, the question of minimizing the size of BSPs for given inputs had been studied only empirically. We concentrate here on orthogonal objects, a case which arises frequently in practice and deserves special attention. We construct BSPs of linear size for any set of orthogonal line segments in the plane. In three dimensions, BSPs of size $O(n^{3/2})$ for any set of $n$ mutually orthogonal line segments or rectangles are constructed. These bounds are optimal and may be contrasted with the $Q(n^2)$ bound for general polygonal objects in $R^3$.

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom

* Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto CA 94304
USA

April 1990

# Optimal Binary Space Partitions for Orthogonal Objects

*Michael S. Paterson* *        *F. Frances Yao* #

**Abstract**

A *binary space partition*, or BSP, is a scheme for recursively dividing a configuration of objects by hyperplanes until all objects are separated. BSPs are widely used in computer graphics as the underlying data structure for computations such as real-time hidden-surface removal, ray tracing, and solid modelling. In these applications, the computational cost is directly related to the *size* of the BSP, i.e., the total number of fragments of the objects generated by the partition. Until recently, the question of minimizing the size of BSPs for given inputs had been studied only empirically. We concentrate here on orthogonal objects, a case which arises frequently in practice and deserves special attention. We construct BSPs of linear size for any set of orthogonal line segments in the plane. In three dimensions, BSPs of size $O(n^{3/2})$ for any set of $n$ mutually orthogonal line segments or rectangles are constructed. These bounds are optimal and may be contrasted with the $\Theta(n^2)$ bound for general polygonal objects in $R^3$.

## 1 Introduction

For geometric problems where the input is a set of objects in the plane or in space, efficient algorithms are often based on recursive partitioning. The input is divided into two parts by splitting the objects with a line (in the 2-D case) or with a plane (in the 3-D case). The two resulting sets are then divided recursively until finally subproblems of some trivial size are obtained. Since each division may split some of the objects into two parts, the process described above can lead to a proliferation of objects. For efficiency, the dividing cuts must be chosen carefully so that fragmentation of the input objects is minimized.

The partitioning method described above was called a *binary space partition* (or BSP) by Fuchs, Kedem and Naylor [3]. They used BSPs to solve hidden-surface removal with changing

*Department of Computer Science, University of Warwick, Coventry, CV4 7AL, England. The work was done while this author was visiting 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).

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

viewpoints. Other applications of BSPs have been discussed in [5], [7] and [9]. In these applications, the space and time bounds of the computations are proportional to the size of the BSPs constructed. Questions of optimality for BSPs were first studied by Paterson and Yao [7]. It was shown in [7] that the size of an optimal BSP for a general polygonal scene in $R^3$ is $\Theta(n^2)$.

In this paper, we consider BSPs for orthogonal objects. These are useful in practice for representing scenes such as architectural models that are naturally orthogonal, or orthogonal approximations to more complex scenes. In two dimensions, we produce BSPs of linear size for sets of orthogonal line segments. In three dimensions, we show how to construct BSPs of size $O(n^{3/2})$ for any set of $n$ mutually orthogonal rectangles or line segments. The techniques used in obtaining the $O(n^{3/2})$ partitions are quite different from those for the general (polygonal) case in [4]. These bounds are optimal in the worst case. Our construction algorithm can be implemented in time $O(n^{3/2})$.

We mention some applications of our results.

1) From a BSP of size $O(n^{3/2})$ representing an input scene, a correct visibility ordering for any viewing position can be obtained in time $O(n^{3/2})$ via a generalized in-order traversal of the BSP tree (see [2], [7]).

2) Given a rectangular polyhedron described by its $n$ faces, one can generate a CSG (*constructive-solid-geometry*) formula of size $O(n^{3/2})$ for the polyhedron (see [1], [7]).

3) For the *art gallery* problem (see O'Rourke [6]), $O(n^{3/2})$ guards are sufficient to cover the interior of any rectangular polyhedron with $n$ faces. This matches the $\Omega(n^{3/2})$ lower bound given by Seidel (see [6]).

For all three applications mentioned above, the best previous bounds were $O(n^2)$ (for the general polyhedral case).

The definitions and basic properties of BSPs are reviewed in the next section. In Section 3, we show how to construct a partition of size $O(n)$ for any set of $n$ mutually orthogonal line segments in $R^2$. In Section 4, we give a partition of size $O(n^{3/2})$ for orthogonal line segments in $R^3$, and apply this result in Section 5 to obtain an $O(n^{3/2})$ partition for any set of $n$ mutually orthogonal rectangles. Generalizations to higher dimensions are considered in Section 6, and we conclude with some open problems in Section 7.

## 2 Preliminaries

In practice, a solid object in $R^3$ is often represented by its boundary elements, i.e., by a set of polygons approximating its surface. Thus, in the orthogonal formulation of our problem, we take the input $\Gamma$ to consist of a set of $n$ rectangles in $R^3$ with disjoint interiors and with edges parallel to the axes. Since 'orthogonal rectangles' sounds odd, we coin a new term 'orthothetic' which literally means 'placed at right angles'. We thus refer to the input $\Gamma$ as a configuration of *orthothetic rectangles.* In the degenerate case, when each rectangle is a line segment (parallel to one of the axes), we refer to $\Gamma$ as a configuration of *orthothetic line segments.*

The concept of a binary space partition as described in the previous section is intuitively clear; formal definitions of a BSP and the associated cost measures are given below.

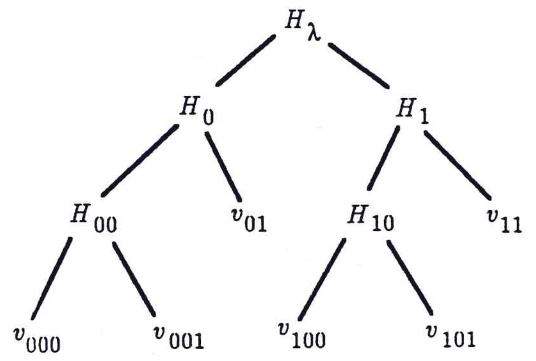


Figure 1.

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 $H_\lambda, H_0, H_1, H_{00}, H_{01}, \ldots$ (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 (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 $\Gamma$ 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 convex polygons. A binary partition $P$ naturally induces a decomposition of $\Gamma$. For any node $v$ of $P$, let $\Gamma(v)$ denote the collection of subfacets, $\Gamma \cap R(v)$. For a given $\Gamma$, we shall be 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 *binary space partition* (or BSP) *of* $\Gamma$. 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 binary space partition of $\Gamma$ is the total weight of its internal nodes, which is also the total number of subfacets generated by $P$. The *partition complexity* of $\Gamma$, denoted by $p(\Gamma)$, is $\min\{|P| \mid P \text{ is a binary space partition of } \Gamma\}$. Define $p_d(n) = \max\{p(\Gamma) \mid |\Gamma| = n, \Gamma \subseteq R^d\}$. In this paper we consider only orthothetic configurations and define

$$\bar{p}_d(n) = \max\{p(\Gamma) \mid |\Gamma| = n, \Gamma \subseteq R^d \text{ and } \Gamma \text{ orthothetic}\}.$$

A simple yet useful device which prevents excessive fragmentation is the concept of a 'bounded cut'. Assume that at some stage of a partition we have a region $R(v)$ which is completely separated by some facet $A$ of $\Gamma$. In such a situation, an immediate partition of $R(v)$ along $A$ is advantageous, since it does not cut through any elements of $\Gamma(v)$, and it will prevent $A \cap R(v)$ itself from ever being cut. We refer to such a cut by $A$ as a *bounded cut.*

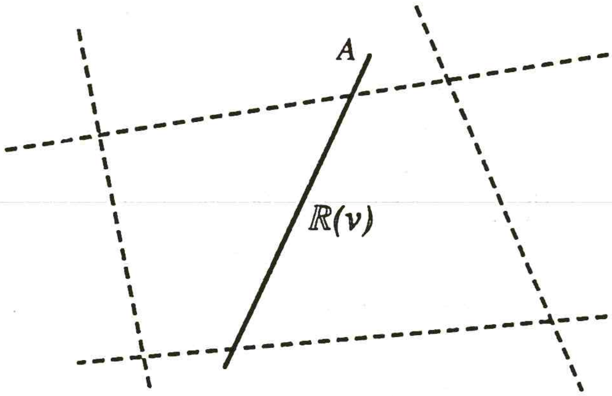


Figure 2.

# 3 Optimal Orthogonal Partitions

In this section we consider the case where $\Gamma$ consists of $n$ horizontal or vertical segments in the plane, and show that an $O(n)$ partition for $\Gamma$ 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 $\Gamma$, that is, all segments of $\Gamma$ 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 $A_L$ and $A_R$ denote the sets of horizontal segments anchored on the sides $(x = x_0)$ and $(x = x_1)$ respectively. Similarly define the sets, $A_B$ and $A_T$, of vertical segments anchored at the bottom and top of $R$.

We define a T-*decomposition* of $R$ as follows. Let $s$ be a longest segment in $A_L$, and suppose line($s$) intersects some segment of $A_B \cup A_T$. 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 line($t$), and then splitting the area to the left of $t$ along line($s$). Any anchored segment which is intersected permits a bounded cut. The T-decomposition is completed by making all such bounded cuts. (See Figure 3.) The following fact is easy to verify.

**Fact.** In a T-decomposition, the only anchored segments that are cut belong to $A_R$, and all bounded cuts occur in $R_3$.

We shall define a partition for $\Gamma$ by recursively applying T-decompositions. Before applying the recursion, we may need to rotate the rectangles to ensure that each segment of $\Gamma$ is cut

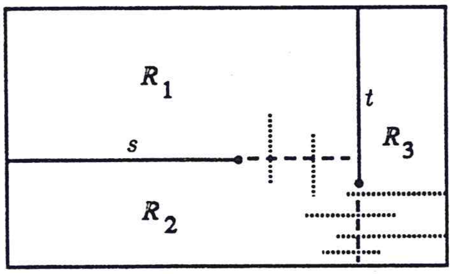


Figure 3.

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.

After the first cut of an unanchored segment, we will color one of the two resulting anchored segments *green* and the other one *red*. When a *green* segment is cut, both subsegments will be made *red*. (Figure 4.) We shall ensure that no *red* segment is ever cut and so each original segment can be cut at most twice. A side of a rectangle $R$ is considered *green* if all segments anchored on that side are *green*.

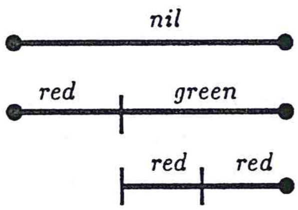


Figure 4.

**Lemma 1.** Let $R$ be a rectangle with a green side.

(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) At least one of the resulting rectangles inherits a green side from the original rectangle. We ensure that the cut edge is *green* for the other rectangle by coloring *its* end of each cut segment *green*.

(ii) For each (unanchored) segment cut by line($s$), we color its upper part *red* and its lower part *green*. Then the side of $R_1$ defined by line($t$) and the side of $R_2$ defined by line($s$) are *green*. For each subrectangle of $R_3$, its right side is still *green*. □

We describe our partition algorithm recursively.

**Orthogonal Partition Algorithm (OPA)**

$\Gamma$ is a set of segments, $R$ is a bounding rectangle for $\Gamma$, and $R$ has a *green* side.

(i) If $R$ is empty then we are finished.

(ii) If there is some segment $s$ such that line($s$) cuts no anchored segment, then partition $R$ along 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. Apply OPA recursively on the the resulting set of subrectangles.

**Theorem 1.** Algorithm OPA finds a partition of size $O(n)$ in $O(n\log^2 n)$ time.

**Proof.** By Lemma 1, the invariant that $R$ has a *green* side is preserved. Since any original segment is cut at most twice, the size of the resulting partition is at most $3n$.

To analyze the running time of OPA, we first describe the data structures needed for carrying out the T-decomposition on a rectangle $R$. Since separate, but symmetric, data structures will be maintained for the set of horizontal segments and the set of vertical segments, we will concentrate on the former case only. The set of horizontal segments is composed of $A_L$ and $A_R$, the sets of segments anchored on the left and on the right respectively, and $U_H$, the set of unanchored segments. We use separate data structures for these three sets. To represent $A_L$ (and $A_R$, similarly), we take the set $V$ of right endpoints of segments in $A_L$, and create 1) a search tree $S(A_L)$ for the $x$-coordinates of $V$, and 2) a priority search tree $Pr(A_L)$ for $V$. The structure $Pr(A_L)$ stores $V$ as a search tree by $y$-value and maintains a priority queue of maximal values of $x$. For the set $U_H$, we will use a search tree $S(U_H)$ for its $y$-coordinates, and a segment tree Seg($U_H$) with respect to its $x$-coordinates. (The set of segments stored at a node of Seg($U_H$) will be ordered by $y$-coordinates.)

To carry out a T-decomposition, we first find the segment $s$ by searching $S(A_L)$, and then find the segment $t$ by searching $Pr(A_T)$ and $Pr(A_B)$.

The partition by line($t$) is carried out as follows. We search $S(A_R)$ and Seg($U_H$) to find the segments intersected by line($t$). Those segments of $A_R$ that are intersected by line($t$) define the bounded cuts for $R_3$; they are deleted from the set $A_R$ for $R_3$, and added to the set $A_R$ for $R_2$. Those segments of $U_H$ that are cut by line($t$) are added to the set $A_R$ for $R_2$, and also to the set $A_L$ for $R_3$. The search tree and the priority search tree associated with $A_R$ for $R_2$ (and also those with $A_L$ for $R_3$) are then rebuilt. The remaining uncut segments of $U_H$ are split into two sets, to the left and to the right of line($t$), respectively, with their associated search tree and segment tree extracted from those of $U_H$. Finally, each remaining tree structure of $R$, except for those representing $A_L$, is split into two subtrees as a result of the partition by line($t$). The partition by line($s$) is performed in a similar way; it is somewhat simpler since only segments of $U_V$ can be cut by line($s$), and the data structures associated with $A_T$ and $A_B$ are unaffected by the cut.

We now analyze the running time of Algorithm OPA. Since each recursive call removes at least one subsegment from further consideration, the total number of calls is $O(n)$ by Lemma 1. The cost of step (i) of the algorithm is $O(1)$, while the test in step (ii) can be done in $O(\log n)$ time by finding the required information from the priority search trees associated with the four bounding edges of $R$. We next consider the total cost over all recursive calls for performing the T-decompositions in step (iii). It is easy to account for the cost of most operations by allowing $O(\log n)$ time for the generation and removal of each subsegment in the course of the partition. The only operation whose cost cannot be accounted for this way is the splitting of the segments trees "along the grain", that is, the splitting of Seg($U_V$) by line($s$) and the splitting of Seg($U_H$) by line($s$). By maintaining sorted order for all segments stored at the same node of a segment tree (and also using the search tree associated with the set), we can carry out the splitting operation in time $O((m_1+1)\log m)$, if an original segment tree for $m$ segments is divided into two subtrees for $m_1$ and $m_2$ segments, with $m_1+m_2=m$ and $m_1 \leq m_2$. The recurrence relation $f(m) \leq f(m_1)+f(m_2)+cm_1\log m$ has a solution $f(m) \leq cm\log^2 m$. We thus conclude that Algorithm OPA has a total running time of $O(n\log^2 n)$. □

**Corollary 1.** $\bar{p}_2(n) = \Theta(n)$.

## 4 Partitions of Orthothetic Lines

A configuration $\Gamma$ in $R^3$, consisting of $p_i$ line segments parallel to the $x_i$-axis for $i=1,2,3$, is said to have *type* $t(\Gamma)=(p_1,p_2,p_3)$ and *size* $\Sigma(\Gamma)=p_1+p_2+p_3$. Our design of an efficient BSP for $\Gamma$ makes crucial use of the parameter $\Pi(\Gamma)=p_1p_2p_3$, referred to as the *measure* of $\Gamma$.

We shall use the following simple lemma.

**Lemma 2.** If $a_i, b_i$ are non-negative reals for $1 \leq i \leq r$, then

$$\min\{\Pi_{i=1}^{r} a_i,\ \Pi_{i=1}^{r} b_i\} \leq \tfrac{1}{2}\Pi_{i=1}^{r}(a_i+b_i).$$

**Proof.**

$$\begin{aligned}\min\{\Pi_i a_i,\ \Pi_i b_i\} &\leq \sqrt{\Pi_i a_i \Pi_i b_i}\\ &= \Pi_i\sqrt{a_i b_i}\\ &\leq \tfrac{1}{2}\Pi_i(a_i+b_i).\end{aligned}$$

□

An *i-cut* ($1 \leq i \leq 3$) is a partition of $\Gamma$ by a plane perpendicular to the $x_i$-axis into two subconfigurations $\Gamma', \Gamma''$ with $t(\Gamma')=(p_1',p_2',p_3')$ and $t(\Gamma'')=(p_1'',p_2'',p_3'')$. We will have $p_i' \leq p_i$, $p_i'' \leq p_i$, $p_j'+p_j'' \leq p_j$ and $p_k'+p_k'' \leq p_k$, where $\{i,j,k\}=\{1,2,3\}$. At least one of the two final inequalities is strict if we choose a cutting plane which contains some line segment.

**Lemma 3.** Given a configuration $\Gamma$ with $m(\Gamma) > 0$, for any $i$ there is an $i$-cut producing $\Gamma'$ and $\Gamma''$ such that $max\{\Pi(\Gamma'), \Pi(\Gamma'')\} \leq \frac{1}{4}\Pi(\Gamma)$.

**Proof.** Suppose the cutting plane is '$x_i = c$', oriented so that $t(\Gamma')$ increases with $c$. Choose a maximum $c$ such that $\Pi(\Gamma') \leq \frac{1}{4}\Pi(\Gamma)$. We may assume that the $i$-cut passes through some segment so $p'_i p'_j p'_k \leq \frac{1}{4}\Pi(\Gamma) < p'_i(p_j - p''_j)(p_k - p''_k)$. The right side is the measure of $\Gamma'$ when $c$ is increased slightly. Therefore $(p_j - p''_j)(p_k - p''_k) > \frac{1}{4}p_j p_k$. It follows from Lemma 2 with $r = 2$ that $p''_j p''_k \leq \frac{1}{4}p_j p_k$. The last inequality proves the Lemma. □

For the partitioning of rectangles in the next section, we will need to cycle through the three coordinates rather than choose an arbitrary direction to cut at each step. Therefore we present a line-partitioning algorithm which proceeds in 'rounds'. A *round* is a sequence of up to three cuts corresponding to distinct values of i, where each cut satisfies the inequality of Lemma 3. An *S-round*, $S \subseteq \{1, 2, 3\}$, is a round consisting of $|S|$ cuts with indices given by the elements of $S$ in any order.

**Constructing a BSP for $\Gamma$**

Given $\Gamma$ with $t(\Gamma) = (p_1, p_2, p_3)$, assume without loss of generality that $p_1 \geq p_2 \geq p_3$. Provided that $p_3 > 0$, let $u_i = \lfloor \log p_i \rfloor$[1] for $i = 1, 2, 3$, and define the (infinite) ternary sequence

$$\sigma \equiv \sigma_0\sigma_1\sigma_2\ldots = \underline{3}^{u_2-u_3}(\underline{2}\ \underline{3})^{u_1-u_2}(\underline{1}\ \underline{2}\ \underline{3})^{\infty}.$$

We use $\sigma$ to define a partition $P_\sigma$ of $\Gamma$, described in the form of a binary tree:

> **Stage 1.** If the configuration at a node on the $r^{\text{th}}$ level (where the root is on the $0^{\text{th}}$ level) has nonzero measure, a $\sigma_r$-cut satisfying the inequality of Lemma 3 is made.
>
> **Stage 2.** If the configuration at a node has measure zero, then for at least one of the axes, say $x_i$, there are no segments parallel to it. We separate the configuration as much as possible with suitable $i$-cuts, and then apply optimal 2-D partitions as provided by Theorem 1.

Thus, in each path of the tree, the partition $P_\sigma$ performs (a prefix of) a sequence of $u_2 - u_3$ $\{3\}$-rounds, $u_1 - u_2$ $\{2, 3\}$-rounds, and arbitrarily many $\{1, 2, 3\}$-rounds until the measure has been reduced to zero; the process is then finished off with optimal two-dimensional partitions.

**Lemma 4.** Let $S(\Gamma, \sigma)$ be the total number of line segments generated by Stage 1 of $P_\sigma$ from $\Gamma$. Then $S(\Gamma, \sigma) = O(\sqrt{p_1 p_2 p_3} + p_1)$.

**Proof.** Let $w$ be the maximum number of $\{1, 2, 3\}$-rounds used in any branch of the partition tree $P_\sigma$. By Lemma 3, each cut reduces the measure by a factor of at least 4, hence the depth of cutting is at most $1 + \frac{1}{2}\log \Pi(\Gamma)$. If $w \geq 0$, we have

$$(u_2 - u_3) + 2(u_1 - u_2) + 3(w - 1) \leq \tfrac{1}{2}\log \Pi(\Gamma)$$

[1] All logarithms here are to the base 2.

and hence

$$\begin{aligned}
3w &\leq 3+\tfrac{1}{2}\log(p_1p_2p_3)-2u_1+u_2+u_3\\
&< 5+\tfrac{3}{2}\log(p_2p_3/p_1)\\
&< \tfrac{3}{2}(\log(11p_2p_3/p_1)).
\end{aligned}$$

Thus $w \leq max\{0, \frac{1}{2}(\log(11p_2p_3/p_1))$. The total depth of 1-cuts, 2-cuts and 3-cuts is at most $w$, $u_1-u_2+w$ and $u_1-u_3+w$ respectively, so

$$\begin{aligned}
S(\Gamma,\sigma) &\leq 2^w(p_1+2^{u_1-u_2}p_2+2^{u_1-u_3}p_3)\\
&= O(2^w p_1)\\
&= O(\sqrt{p_1p_2p_3}+p_1).
\end{aligned}$$

This proves the Lemma. □

**Theorem 2.** For any configuration $\Gamma$ of $n$ orthothetic line segments in $R^3$, a BSP of size $O(n^{3/2})$ can be found in time $O(n^{3/2})$. In particular, $p(\Gamma)=O(\sqrt{\Pi(\Gamma)}+\Sigma(\Gamma))$. Furthermore there are $\Gamma$ for which $p(\Gamma)=\Omega(\sqrt{\Pi(\Gamma)}+\Sigma(\Gamma))$.

**Proof.** We note that, in Stage 2 of $P_\sigma$, the configuration is separated into disjoint 2-D subconfigurations without cutting any segments, and the optimal 2-D partitions increase the total size by at most a constant factor. The upper bound now follows from Lemma 4.

The partition $P_\sigma$ can be constructed in time $O(n^{3/2})$. A naive algorithm is adequate to achieve this time bound. A configuration $\Gamma$ is represented by six sorted lists, $L_{ij}$ where $i \neq j, 1 \leq i,j \leq 3$. The list $L_{ij}$ contains all the segments of $\Gamma$ parallel to the $x_i$-axis, sorted in increasing order of $x_j$-coordinates. To determine the proper $i$-cut, as described in Lemma 3, it is sufficient to step through the pair of lists $L_{ji}, L_{ki}$ in increasing $x_i$-order until the appropriate balance point is reached. The corresponding value of $x_i$ is then used for a cut plane. To construct representations for the succeeding configurations $\Gamma', \Gamma''$, a linear pass through the representation of $\Gamma$ is sufficient. Thus the time for performing an $i$-cut on $\Gamma$ is linear in the size of $\Gamma$.

The total running time of our algorithm is therefore bounded by the sum of the configuration sizes at all the nodes of the partitioning tree. Throughout the initial sequence of $\{3\}$-rounds and $\{2,3\}$-rounds (of length $O(\log p_1)$), the $p_1$ segments parallel to the $x_1$-axis are never cut and the total number of subsegments generated is $O(p_1)$. Then for each $\{1,2,3\}$-round performed, the total number of subsegments at most doubles. The number of these rounds is $\frac{1}{2}\log(p_2p_3/p_1)+O(1)$. Hence the total size summed over all rounds is

$$O(p_1\log p_1+\sqrt{p_1p_2p_3})=O\left(\sqrt{\Pi(\Gamma)}+\Sigma(\Gamma)\log\Sigma(\Gamma)\right)=O(n^{3/2}).$$

The lower bound is shown by extending an example of Thurston [10], mentioned in [7]+. We take a rectangular parallelepiped of size $a_1\times a_2\times a_3$ in a three-dimensional grid, and connect corresponding grid points on opposite faces of the parallelepiped with line segments. If we move

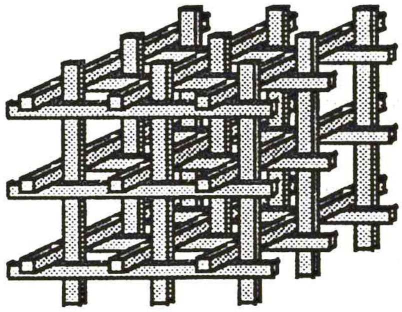

Figure 5.

the three families of lines slightly so that they become all disjoint, we obtain a configuration $\Gamma$ of type $t(\Gamma) = (p_1, p_2, p_3)$ where $p_1 = a_2a_3$, $p_2 = a_3a_1$, $p_3 = a_1a_2$. Figure 5 illustrates the $3 \times 3 \times 3$ example of this. For clarity, the line segments are represented as rectangular rods. It can be argued that any BSP for $\Gamma$ must cut at least one of the three line segments in the neighborhood of each of the $a_1a_2a_3$ grid points internal to the parallelepiped. Thus at least $a_1a_2a_3 + (p_1 + p_2 + p_3) = \sqrt{p_1p_2p_3} + (p_1 + p_2 + p_3)$ segments will be generated by any BSP. □

## 5 Partitions of Orthothetic Rectangles

We shall represent each rectangle by the set of four segments comprising its boundary, and apply the partitioning algorithm $P_\sigma$ to this configuration of segments. The resulting partition is almost a BSP for the set of rectangles also. Any subregion corresponding to a leaf of the BSP contains no edge, but may contain subrectangles internal to some original rectangle. However, all such subrectangles can be removed using bounded cuts.

It remains to analyze the sizes of these BSPs.

**Lemma 5.** Any orthothetic rectangle subjected to $d$ rounds of cut (with bounded cuts taken whenever possible) generates at most $3 \cdot 2^{d+2}$ subrectangles.

**Proof.** Each round at most subdivides any rectangle into a $2 \times 2$ array of subrectangles. To establish the Lemma we prove, by induction on $d$, a more detailed bound in which we take account of how many edges of the rectangle are original edges, as opposed to cut edges created by earlier cuts.

For $0 \leq g \leq 4$ and $d \geq 0$, let $F(g, d)$ be the maximum number of subrectangles, generated by $d$ rounds and bounded cuts, that originate from one rectangle with $g$ original edges.

**Claim.**

$$F(g, d) \leq 3g(2^d - 1) + 1$$

**Proof of Claim.** If $d = 0$ the result is trivial, while if $g = 0$ the rectangle may be removed with a bounded cut. For $d \geq 1, g \geq 1$, we may suppose that the first round divides the rectangle into

4 rectangles with $g_1, g_2, g_3, g_4$ original edges respectively, where $\Sigma_{i=1}^4 g_i \leq 2g$. Hence

$$\begin{aligned} F(g,d) &\leq \Sigma_i F(g_i, d-1) \\ &\leq 3\Sigma_i g_i(2^{d-1}-1)+4 \text{ by induction} \\ &\leq 6g(2^{d-1}-1)+4 \\ &\leq 3g(2^d-1)+1 \text{ since } g \geq 1. \end{aligned}$$

This proves the Claim, and hence the Lemma. □

**Theorem 3.** For a set $\Gamma$ of $n$ orthothetic rectangles, $p(\Gamma) = O(n^{3/2})$. In addition, if two of the three classes of rectangles have at most $b$ members ($b > 0$) then $p(\Gamma) = O(n\sqrt{b})$.

**Proof.** Suppose the number of rectangles of $\Gamma$ perpendicular to the $x_i$-axis is $a, b, c$ for $i = 1, 2, 3$ respectively. If any of $a, b, c$ is zero the result follows from the 2-D case. Otherwise we may assume that $0 < a \leq b \leq c$.

The type of $\Gamma$, represented by the perimeter edges is $t(\Gamma) = (p_1, p_2, p_3) = (2(b+c), 2(a+c), 2(a+b))$. Hence $p_1 = O(c)$, $p_2 = O(c)$, $p_3 = O(b)$, $p_2/p_3 < c/b$ and $p_1/p_2 < 2$.

In applying the partitioning algorithm $P_\sigma$ to $\Gamma$, the numbers $v, v', w$ of $\{3\}$-rounds, $\{2,3\}$-rounds and $\{1,2,3\}$-rounds respectively satisfy

$$v \leq \log\frac{c}{b} + 1,\ v' \leq 1,\ w \leq \frac{1}{2}\log b + O(1).$$

Therefore, by using Lemma 5, it can be shown that the resulting BSP has size at most

$$O(b \cdot 2^{v+v'+w} + c \cdot 2^{v'+w}) \leq O(c\sqrt{b}).$$

This inequality proves the Theorem. □

**Corollary 2.** $\bar{p}_3(n) = \Theta(n^{3/2})$. (The lower bound was proved in Theorem 2.)

## 6 Higher Dimensional Binary Space Partitions

We obtain analogous results in higher dimensions for configurations of orthothetic line segments, although we have not been able to extend these results to orthothetic hyper-rectangles as in the three-dimensional case. In dimensions higher than three, it may be the case that some region of a partition contains no edges, and yet the subconfiguration in it requires nontrivial partitioning.

For higher dimensions, alternative definitions for a BSP may be proposed, depending on the treatment of lower-dimensional subconfigurations. If we require that a BSP completely decompose lower-dimensional subconfigurations, a different complexity function $p^*$ is obtained. Whether or not such complete decomposition is appropriate depends of course on the particular application. In three dimensions, the difference between $p$ and $p^*$ is not significant since, by

Theorem 1, a two-dimensional configuration occurring in $\Gamma(v) \cap H_v$ at some node $v$ can be completely partitioned with at most a linear increase in its size. Theorem 4 shows that if $\Gamma$ is a $d$-dimensional configuration of $n$ orthothetic line segments then $p(\Gamma) = O(n^{d/(d-1)})$. However, if $\Gamma$ contains a configuration such as that of Figure 5 lying in a three-dimensional subspace, then clearly $p^*(\Gamma) = \Omega(n^{3/2})$.

**Theorem 4.** For a configuration $\Gamma$ of orthothetic line segments in $d$ dimensions

$$p(\Gamma) = O\left((\Pi(\Gamma))^{\frac{1}{d-1}} + \Sigma(\Gamma)\right).$$

There are $\Gamma$ for which

$$p(\Gamma) = \Omega\left((\Pi(\Gamma))^{\frac{1}{d-1}} + \Sigma(\Gamma)\right).$$

**Proof.** We assume that $p_1 \geq \cdots \geq p_d$, and let $u_i = \lfloor \log_2 p_i \rfloor$. If $p_d = 0$ then $\Gamma$ is contained in some finite set of $x_d$-hyperplanes and $p(\Gamma) = \Sigma(\Gamma)$ by definition of $p(\Gamma)$. Otherwise, we use a sequence of rounds of cuts as before. The defining index sequence is

$$\sigma = \underline{d}^{u_{d-1}-u_d}\ (\underline{d\text{-}1}\ \underline{d})^{u_{d-2}-u_{d-1}} \cdots (\underline{2}\ \underline{3}\cdots\underline{d})^{u_1-u_2}(\underline{1}\ \underline{2}\cdots\underline{d})^{\infty}.$$

Let $w$ be the maximum number of $\{1,\cdots d\}$-rounds encountered in any path of the partition tree. By a generalisation of Lemma 3, each cut generates two subconfigurations with measure decreased by a factor of at least $2^{d-1}$, hence if $w > 0$ we must have

$$\begin{aligned}
\log \Pi(\Gamma) &\geq (d-1)((u_{d-1}-u_d) + 2(u_{d-2}-u_{d-1}) + \cdots + (d-1)(u_1-u_2) + d(w-1)) \\
&= (d-1)(d(w+u_1-1) - \Sigma_{i=1}^{d} u_i) \\
&> (d-1)(d(w+\log p_1 - 2) - \log \Pi(\Gamma)).
\end{aligned}$$

Therefore, if $w > 0$ then

$$\log \Pi(\Gamma) \geq (d-1)(d \log p_1 + w - 2). \qquad (*)$$

Since

$$\begin{aligned}
S(\Gamma,\sigma) &\leq 2^w(p_1 + p_2 2^{u_1-u_2} + p_3 2^{u_1-u_3} + \cdots + p_d 2^{u_1-u_d}) \\
&\leq 2^w p_1 2d,
\end{aligned}$$

it follows from $(*)$ that

$$\begin{aligned}
S(\Gamma,\sigma) &= O\left(\left((\Pi(\Gamma))^{\frac{1}{d-1}} + p_1\right) d\right) \\
&= O\left((\Pi(\Gamma))^{\frac{1}{d-1}} + \Sigma(\Gamma)\right)
\end{aligned}$$

for fixed $d$.

The lower bound is proved by an example based on an $a_1 \times a_2 \times \cdots \times a_d$ parallelepiped. analogous to that in the proof of Theorem 2. Let $A = \Pi_j a_j$. Then $p_i = A/a_i$ for all $i$, and $\Pi(\Gamma) = A^{d-1}$. Since at least one cut is needed in the vicinity of each grid point, we have

$$p(\Gamma) \geq A + \Sigma(\Gamma) = \Omega\left((\Pi(\Gamma))^{\frac{1}{d-1}} + \Sigma(\Gamma)\right). \qquad \square$$

## 7 Open Problems

We mention two directions in which our present results might be generalized.

1. Extend Theorem 4 from line segments to $k$-facets for $2 \leq k < d$; that is, find efficient BSPs for higher dimensional facets in $R^d$ for $d \geq 4$ and establish corresponding lower bounds.

2. Prove an analogue of Theorem 1 for planar configurations consisting of three or more families of parallel line segments, and consider similar generalizations in higher dimensions.

## References


[1] 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.

[2] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987.

[3] H. Fuchs, Z. Kedem and B. Naylor, "On visible surface generation by a priori tree structures", *Computer Graphics* (SIGGRAPH '80 Conference Proceedings), 124-133.

[4] E. McCreight, "Priority Search Trees", *SIAM J. Comput.* **14**, 1985, 257-276.

[5] B. Naylor, "*A priori* based techniques for determining visibility priority for 3-d scenes". Ph. D. dissertation, Univ. of Texas at Dallas, 1981.

[6] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford Univ. Press, 1987.

[7] M. Paterson and F. Yao, "Binary partitions with applications to hidden-surface removal and solid modelling", Xerox PARC Technical Report CSL-89-6 (also to appear in *Discrete & Computational Geometry*). An earlier version [7]$^+$ appeared in *Proceedings of the Fifth Annual ACM Symposium on Computational Geometry*, Saarbruchen, West Germany, June 1989, pp. 23-32.

[8] D. Peterson, "Halfspace representations of extrusions, solids of revolution, and pyramids", SANDIA Report SAND84-0572, Sandia National Laboratories, 1984.

[9] W. Thibault and B. Naylor, "Set operations on polyhedra using binary space partitioning trees", *Computer Graphics* **21**, 1987, 153-162.

[10] W. Thurston, private communication.