SIAM J. COMPUT. © 2000 Society for Industrial and Applied Mathematics
Vol. 29, No. 5, pp. 1422–1448

BINARY SPACE PARTITIONS FOR FAT RECTANGLES[*]

PANKAJ K. AGARWAL[†], EDWARD F. GROVE[‡], T. M. MURALI[§], AND JEFFREY SCOTT VITTER[¶]

Abstract. We consider the practical problem of constructing binary space partitions (BSPs) for a set S of n orthogonal, nonintersecting, two-dimensional rectangles in $\mathbb{R}^3$ such that the aspect ratio of each rectangle in S is at most α, for some constant $\alpha \geq 1$. We present an $n2^{O(\sqrt{\log n})}$-time algorithm to build a binary space partition of size $n2^{O(\sqrt{\log n})}$ for S. We also show that if m of the n rectangles in S have aspect ratios greater than α, we can construct a BSP of size $n\sqrt{m}2^{O(\sqrt{\log n})}$ for S in $n\sqrt{m}2^{O(\sqrt{\log n})}$ time. The constants of proportionality in the big-oh terms are linear in $\log \alpha$. We extend these results to cases in which the input contains nonorthogonal or intersecting objects.

Key words. binary space partitions, rectangles, aspect ratio, computational geometry, computer graphics, solid modelling

AMS subject classifications. 65Y25, 68P05, 68Q20, 68Q25, 68U05

PII. S0097539797320578

1. Introduction.

Rendering a set of opaque or partially transparent objects in $\mathbb{R}^3$ quickly and in a visually realistic way is a fundamental problem in computer graphics [15]. A central component of this problem is *hidden-surface removal*: given a set of objects, a viewpoint, and an image plane, compute the scene visible from the viewpoint as projected onto the image plane. Because of its importance, the hidden-surface removal problem has been studied extensively in both the computer graphics and the computational geometry communities [14, 15, 28]. One of the conceptually simplest solutions to this problem is the z-buffer algorithm [8, 15]. This algorithm sequentially processes the objects; for each object, it updates the pixels of the image plane covered by the object, based on the distance information stored in the z-buffer. A very fast hidden-surface removal algorithm can be obtained by implementing the z-buffer in hardware. However, only special-purpose and costly graphics engines contain fast z-buffers, and z-buffers implemented in software are generally inefficient. Even

[*]Received by the editors May 2, 1997; accepted for publication (in revised form) March 30, 1999; published electronically March 15, 2000. A preliminary version of this paper appeared in the *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 1996.

http://www.siam.org/journals/sicomp/29-5/32057.html

[†]Box 90129, Department of Computer Science, Duke University, Durham, NC 27708–0129 (pankaj@cs.duke.edu). Support of this author was provided by National Science Foundation research grant CCR–93–01259, Army Research Office MURI grant DAAH04–96–1–0013, a Sloan fellowship, a National Science Foundation NYI award with matching funds from the Xerox Corporation, and a grant from the U.S.–Israeli Binational Science Foundation.

[‡]946 Tamarack Lane #13, Sunnyvale, CA 94086 (efg@cs.duke.edu). Support of this author was provided by Army Research Office grant DAAH04–93–G–0076. This work was partially done when the author was at Duke University.

[§]Computer Science Department, Stanford University, Gates 133, 353 Serra Mall, Stanford, CA 94305 (murali@cs.stanford.edu). Support of this author was provided in part by National Science Foundation research grant CCR–9522047 and by Army Research Office MURI grant DAAH04–96–1–0013. This author performed this research when he was affiliated with Brown University and was a visiting scholar at Duke University.

[¶]Box 90129, Department of Computer Science, Duke University, Durham, NC 27708–0129 (jsv@cs.duke.edu). Support of this author was provided in part by National Science Foundation research grant CCR–9522047, by Army Research Office grant DAAH04–93–G–0076, and by Army Research Office MURI grant DAAH04–96–1–0013.

when fast hardware z-buffers are present, they are not fast enough to handle the huge models (containing hundreds of millions of polygons) that often have to be displayed in real time. As a result, other methods have to be developed either to "cull away" a large subset of invisible polygons so as to decrease the rendering load on the graphics pipeline (when models are large; see, e.g., [29]) or to completely solve the hidden-surface removal problem (when there are very slow or no z-buffers).

One technique to handle both of these problems is the *binary space partition* (BSP), a data structure introduced by Fuchs, Kedem, and Naylor [16] that is based on work by Schumacker et al. [27]. Fuchs, Kedem, and Naylor use the BSP to implement the so-called "painter's algorithm" for hidden-surface removal; the painter's algorithm draws the objects to be displayed on the screen in a back-to-front order (in which no object is occluded by any object earlier in the order). In general, it is not possible to find a back-to-front order from a given viewpoint for an arbitrary set of objects. By fragmenting the objects, the BSP ensures that a back-to-front order from *any* viewpoint can be determined for the fragments [16].

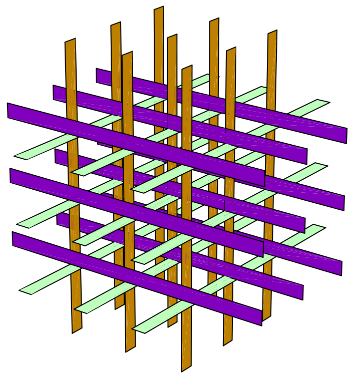
The BSPs have subsequently proven to be versatile, with applications in many other problems—global illumination [6], shadow generation [10, 11], visibility problems [4, 29], solid modeling [22, 24, 30], geometric data repair [19], ray tracing [21], robotics [5], approximation algorithms for network design [17], and surface simplification [3]. Algorithms have also been developed to construct BSPs for moving objects [1, 2, 12, 23, 31].

Informally, a BSP $\mathcal{B}$ for a set of $(d-1)$-dimensional objects in $\mathbb{R}^d$ is a binary tree. Each node v of $\mathcal{B}$ is associated with a convex region $\mathcal{R}_v$. The regions associated with the children of v are obtained by splitting $\mathcal{R}_v$ with a hyperplane. If v is a leaf of $\mathcal{B}$, then the interior of $\mathcal{R}_v$ does not intersect any object. The regions associated with the leaves of the tree form a convex decomposition of $\mathbb{R}^d$. The $(d-1)$-dimensional faces of the cells of this decomposition intersect the objects and divide them into fragments; these fragments are stored at appropriate nodes of the BSP.

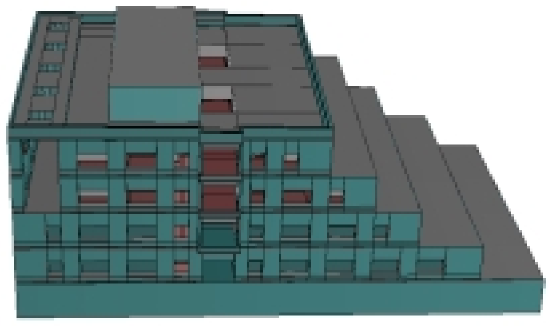
The efficiency of most algorithms that use BSPs depends on the number of nodes in the BSP. As a result, there has been a lot of effort to construct BSPs of small size. Although several simple heuristics have been developed for constructing BSPs of reasonable sizes [4, 7, 16, 20, 29, 30], provable bounds were first obtained by Paterson and Yao. They show that a BSP of size $O(n \log n)$ can be constructed for n disjoint segments in $\mathbb{R}^2$; they also show that a BSP of size $O(n^2)$ can be constructed for n disjoint triangles in $\mathbb{R}^3$, which is optimal in the worst case [25]. But in graphics-related applications, many common environments like buildings are composed largely of orthogonal rectangles, and nonorthogonal objects are approximated by their orthogonal bounding boxes [15]. Paterson and Yao [26] prove that a BSP of size $O(n)$ exists for n nonintersecting, orthogonal segments in $\mathbb{R}^2$, and a BSP of size $O(n\sqrt{n})$ exists for n nonintersecting, orthogonal rectangles in $\mathbb{R}^3$. These bounds are optimal in the worst case.

In all known lower bound examples of orthogonal rectangles in $\mathbb{R}^3$ requiring BSPs of size $\Omega(n\sqrt{n})$, most of the rectangles are "thin." For example, the lower bound proof presented by Paterson and Yao uses a configuration of $\Theta(n)$ orthogonal rectangles, arranged in a $\sqrt{n} \times \sqrt{n} \times \sqrt{n}$ grid, for which any BSP has size $\Omega(n\sqrt{n})$ (see Figure 1.1).All rectangles in their construction have aspect ratio $\Omega(\sqrt{n})$. Such configurations of thin rectangles rarely occur in practice. Many real databases consist mainly of "fat" rectangles; i.e., the aspect ratios of these rectangles are bounded by a constant.

It is natural to ask whether BSPs of near-linear size can be constructed if most



(a)



(b)

FIG. 1.1. (a) *Lower bound for orthogonal rectangles.* (b) *Model of a building—*85% *of the rectangles have aspect ratio at most* 25.

of the rectangles are "fat." We call a rectangle *fat* if its aspect ratio (the ratio of the longer side to the shorter side) is bounded by a fixed constant; for specificity, we use $\alpha \geq 1$ to denote this constant. A rectangle is said to be *thin* if its aspect ratio is greater than α. In this paper, we consider the following problem:

Given a set S of n nonintersecting, orthogonal, two-dimensional rectangles in $\mathbb{R}^3$, of which m are thin and the remaining $n-m$ are fat, construct a BSP for S.

We first show how to construct a BSP of size $n2^{O(\sqrt{\log n})}$ for n fat rectangles in $\mathbb{R}^3$ (i.e., when $m = 0$). We then show that if $m > 0$, a BSP of size $n\sqrt{m}2^{O(\sqrt{\log n})}$ can be built. We also prove a lower bound of $\Omega(n\sqrt{m})$ on the size of such a BSP.

We finally prove two important extensions to these results. If p of the n input objects are nonorthogonal, we show that an $np2^{O(\sqrt{\log n})}$-size BSP exists. Unlike in the case of orthogonal objects, fatness does not help in reducing the worst-case size of BSPs for nonorthogonal objects. In particular, we prove that there exists a set of n fat triangles in $\mathbb{R}^3$ for which any BSP has $\Omega(n^2)$ size. However, nonorthogonal objects can be approximated by orthogonal bounding boxes. The resulting bounding boxes might intersect each other. Motivated by this observation, we also consider the problem in which n fat rectangles contain k intersecting pairs of rectangles, and we show that we can construct a BSP of size $(n+k)\sqrt{k}2^{O(\sqrt{\log n})}$.

In all cases, the constant of proportionality in the big-oh terms is linear in $\log \alpha$, where α is the maximum aspect ratio of the fat rectangles. Our algorithms to construct these BSPs run in time proportional to the size of the BSPs they build. Experiments demonstrate that our algorithms work well in practice and construct BSPs of near-linear size when most of the rectangles are fat, and perform better than most known algorithms for constructing BSPs for orthogonal rectangles [18].

As far as we are aware, ours is the first work to consider BSPs for the practical and common case of (two-dimensional) fat polygons in $\mathbb{R}^3$. De Berg considers a weaker model, the case of (three-dimensional) fat polyhedra in $\mathbb{R}^3$ (a polyhedron is said to be *fat* if its volume is at least a constant fraction of the volume of the smallest sphere enclosing it), although his results extend to higher dimensions [13].

One of the main ingredients of our algorithm is the construction of an $O(n \log n)$-size BSP for a set of n fat rectangles that are "long" with respect to a box B; i.e., none of the vertices of the rectangles lie in the interior of B. To prove this result, we crucially use the fatness of the rectangles. We then develop an algorithm to construct a BSP of size $n2^{O(\sqrt{\log n})}$ for n fat rectangles by simultaneously simulating the algorithm for long rectangles and partitioning the vertices of rectangles in S in a clever manner.

The rest of the paper is organized as follows: section 2 gives some preliminary definitions. In section 3, we show how to build an $O(n \log n)$-size BSP for n long rectangles. Sections 4 and 5 present and analyze our algorithm to construct a BSP of size $n2^{O(\sqrt{\log n})}$ for n fat rectangles. We extend this result in section 6 to construct BSPs for cases in which some objects in the input are thin or nonorthogonal. We conclude in section 7 with some open problems.

2. Geometric preliminaries.

A *binary space partition* $\mathcal{B}$ for a set S of pairwise-disjoint, $(d-1)$-dimensional, polyhedral objects in $\mathbb{R}^d$ is a tree defined as follows: Each node v in $\mathcal{B}$ represents a convex polytope $\mathcal{R}_v$ and a set of objects $S_v = \{s \cap \mathcal{R}_v \mid s \in S\}$ that intersect $\mathcal{R}_v$. The region associated with the root is $\mathbb{R}^d$ itself. If S_v is empty, then node v is a leaf of $\mathcal{B}$. Otherwise, we partition $\mathcal{R}_v$ into two convex polytopes by a *cutting hyperplane* H_v. At v, we store the equation of H_v and $\{s \mid s \in S_v, s \subseteq H_v\}$, the subset of objects in S_v that lie in H_v. If we let H_v^+ be the positive halfspace and H_v^- the negative halfspace bounded by H_v, the polytopes associated with the left and right children of v are $\mathcal{R}_v \cap H_v^-$ and $\mathcal{R}_v \cap H_v^+$, respectively. The left subtree of v is a BSP for the set of objects $S_v^- = \{s \cap H_v^- \mid s \in S_v\}$ and the right subtree of v is a BSP for the set of objects $S_v^+ = \{s \cap H_v^+ \mid s \in S_v\}$. The size of $\mathcal{B}$ is the sum of the number of nodes in $\mathcal{B}$ and the total number of faces of all dimensions of the objects stored at all the nodes in $\mathcal{B}$.

In our case, S is a set of orthogonal rectangles in $\mathbb{R}^3$. In our algorithms, we will use orthogonal cutting planes. Therefore, the region $\mathcal{R}_v$ associated with each node v in $\mathcal{B}$ is a *box* (rectangular parallelepiped). We say that a rectangle r is *long* with respect to a box B if none of the vertices of r lie in the interior of B. Otherwise, r is said to be *short* (see Figure 2.1). A long rectangle is *free* if none of its edges lies in the interior of B; otherwise it is *nonfree*. A *free cut* is a cutting plane that does not cross any rectangle in S and that either divides S into two nonempty sets or contains

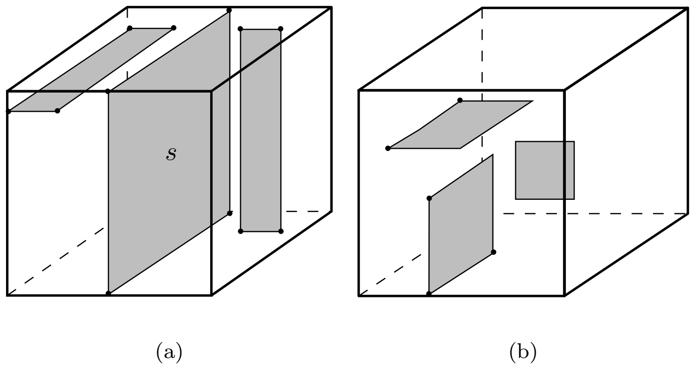


FIG. 2.1. (a) *Long and* (b) *short rectangles. Heavy dots indicate the vertices of these rectangles that lie on the boundary of the box. Rectangle s is a free rectangle.*

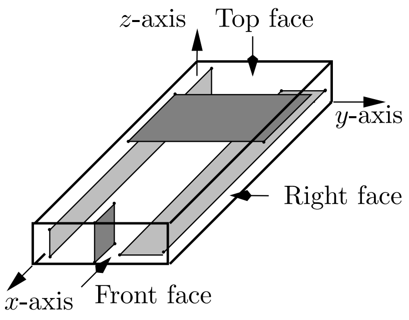


FIG. 2.2. *Different classes of rectangles.*

a rectangle in S. Note that the plane containing a free rectangle is a free cut. Free cuts play a critical role in preventing excessive fragmentation of the rectangles in S.

We will often focus on a box B and construct a BSP for the rectangles intersecting it. Given a set of rectangles R, let

$$R_B = \{s \cap B \mid s \in R\}$$

be the set of rectangles obtained by clipping the rectangles in R within B. For a set of points P, let P_B be the subset of P lying in the interior of B.

A box B has six faces: *top, bottom, front, back, right,* and *left*, as shown in Figure 2.2. We assume, without loss of generality, that the back, bottom, left corner of B is the origin (i.e., the back face of B lies on the yz-plane). A rectangle s that is long with respect to B belongs to the *top class* if two parallel edges of $s \cap B$ are contained in the top and bottom faces of B. We similarly define the *front* and *right* classes. A long rectangle belongs to at least one of these three classes; a nonfree long rectangle belongs to a unique class. See Figure 2.2 for examples of rectangles belonging to different classes.

Although a BSP is a tree, we will often discuss just how to partition the box represented by a node into two boxes. We will not explicitly detail the associated construction of the actual tree itself, since the construction is straightforward once we specify the cutting plane. Sometimes we will abuse notation and use B to refer to

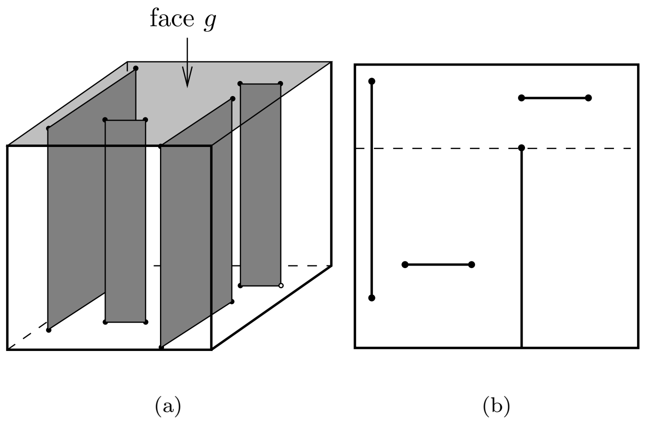


FIG. 2.3. (a) *Long rectangles in the top class.* (b) *Projections of the rectangles in* (a) *onto the top face g; heavy dots indicate the vertices of these rectangles that lie in the interior of g. The dashed line is the cut satisfying* (2.1).

the corresponding node in the BSP as well.

In the rest of the paper, we assume that the vertices of the rectangles in S are sorted by x-, y-, and z-coordinates, and that for each axis, the rectangles perpendicular to that axis are sorted by intercept. The cost of this sort will not affect the asymptotic running times of our algorithms.

We now state two preliminary lemmas that we will use in our algorithms. The first lemma characterizes a set of rectangles that are long with respect to a box and belong to one class. The second lemma applies to two classes of long rectangles.

LEMMA 2.1. *Let C be a box, P a set of points in the interior of C, R a set of rectangles long with respect to C, and $w \geq 1$ a real number. If the rectangles in R_C belong to one class, then the following two conditions hold (see Figure* 2.3*):*

(i) *There exists a face g of the box C that contains exactly one of the edges of each rectangle in R_C. Let V be the set of those vertices of the rectangles in R_C that lie in the interior of g.*

(ii) *We can find a plane that partitions C into two boxes C_1 and C_2 so that for $i = 1, 2$,*

$$|V \cap C_i| + w|P_{C_i}| \leq \frac{|V| + w|P|}{2}. \tag{2.1}$$

If the rectangles in R_C and the points in P are sorted along each of the three axes, the partitioning plane can be computed in $O(|R_C| + |P|)$ time.

Proof. (i) follows from the definition of a class. To prove part (ii) of the lemma, let P^* be the set of projections of the points in P onto g. Assume g is the top face of C. If we associate a weight of 1 with each point in V and a weight w with each point in P^*, the total weight of the points in $V \cup P^*$ is $|V| + w|P|$. By sweeping g, we can find a line l lying in g and parallel to the x-axis that contains a point in $V \cup P^*$ and divides $V \cup P^*$ into two sets, each with weight at most $(|V| + w|P|)/2$. We split C into two boxes C_1 and C_2 using the plane containing l that is orthogonal to g. By construction, C_1 and C_2 satisfy (2.1). The time bound follows easily. ∎

LEMMA 2.2. *Let C be a box, P a set of points in the interior of C, R a set of* *rectangles long with respect to C, and $w \geq 1$ a real number. If the rectangles in R_C belong to two classes, then one of the following two conditions holds (see Figure* 2.4*):*

(i) *We can find one free cut that partitions C into two boxes C_1 and C_2 so that*

$$|R_{C_i}| + w|P_{C_i}| \leq \frac{2\,(|R_C| + w|P|)}{3} \tag{2.2}$$

for $i = 1, 2$.

(ii) *We can find two parallel free cuts that divide C into three boxes C_1, C_2, and C_3 such that all rectangles in R_{C_2} belong to the same class and such that*

$$|R_{C_2}| + w|P_{C_2}| \geq \frac{|R_C| + w|P|}{3}. \tag{2.3}$$

If the rectangles in R_C and the points in P are sorted along each of the three axes, these free cuts can be computed in $O(|R_C| + |P|)$ time.

Proof. We assume without loss of generality that the rectangles in R_C belong to the top and right classes. Let $\bar{r}$ denote the projection of a rectangle $r \in R_C$ onto the x-axis: $\bar{r}$ is either a point or an interval. Similarly, let $\bar{p}$ denote the projection of a point $p \in P$ onto the x-axis. Set

$$U = \left\{ \left(\bigcup_{r \in R_C} \bar{r} \right) \cup \left(\bigcup_{p \in P} \bar{p} \right) \right\}.$$

The set U is a collection of disjoint intervals, some of which may be single points. Let r_1 (resp., r_2) be a rectangle in R_C belonging to the top (resp., right) class. Since the rectangles in R_C are disjoint, it is easily seen that $\bar{r}_1$ and $\bar{r}_2$ are also disjoint. Hence, each connected component of U contains the projections of rectangles belonging to at most one class. For any connected component I of U define

$$\mu(I) = |\{r \in R_C \mid \bar{r} \subseteq I\}| + w|\{p \in P \mid \bar{p} \subseteq I\}|.$$

Set $W = |R_C| + w|P|$. If U contains a connected component $I = [\beta, \gamma]$ with $\mu(I) > W/3$, then the two free cuts are $x = \beta$ and $x = \gamma$. The cuts partition the box C into three boxes C_1, C_2, and C_3, where C_2 denotes the middle box. By construction, all rectangles in R_{C_2} belong to at most one class. Hence, condition (ii) holds. [1]

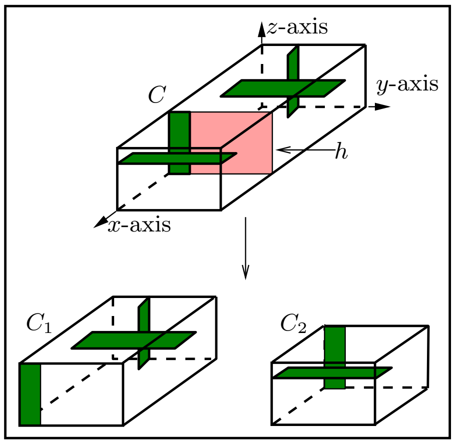
If there is no such connected component of U, then let $I = [\beta, \gamma]$ be the leftmost connected component of U with $\sum_{I' \leq I} \mu(I') > W/3$ (we say that $I' \leq I$ if I' lies to the left of I). Since $\mu(I) \leq W/3$ and $\sum_{I' < I} \mu(I') < W/3$,

$$\sum_{I' \leq I} \mu(I') \leq 2W/3.$$

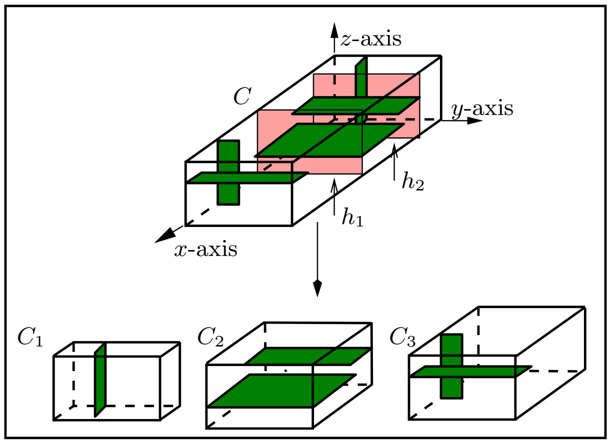
We partition C into two boxes C_1 and C_2 using the cut $x = \gamma$. In this case, condition (i) holds.

If the rectangles in R_C and the points in P are sorted along the x-axis, it is clear that the components in U can be computed and sorted in $O(|R_C| + |P|)$ time. The free cut(s) used to partition C can be found in the same time by sweeping the components of U. ∎

[1] If $\beta = \gamma$, i.e., I is a point, then C_2 is regarded as a degenerate box. If I is the first (resp., last) connected component of U, then C_1 (resp., C_3) may be a degenerate box.



(a)



(b)

FIG. 2.4. (a) *Free cut h partitions box C into two boxes C_1 and C_2.* (b) *Two parallel free cuts h_1 and h_2 partition C into three boxes: C_1, C_2, and C_3.*

3. BSPs for long fat rectangles. Let S be a set of fat rectangles. Assume that all the rectangles in S are long with respect to a box B. In this section, we show how to build a BSP for S_B, the set of rectangles clipped within B. In general, S_B can have all three classes of rectangles. We first exploit the fatness of the rectangles to prove that whenever all three classes are present in S_B, a small number of cuts can divide B into boxes, each of which has only two classes of rectangles. Then we describe an algorithm that constructs a BSP for rectangles belonging to only two classes.

3.1. Reducing three classes to two classes. Assume, without loss of generality, that the longest edge of B is parallel to the x-axis. The rectangles in S_B that

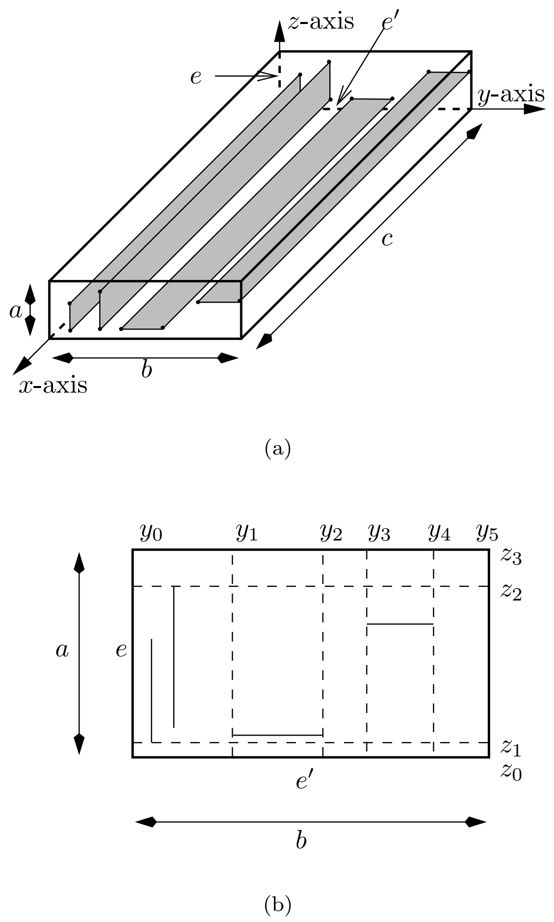


FIG. 3.1. (a) *Rectangles belonging to the sets R and T.* (b) *The back face of B; dashed lines are intersections of the back face with the α-cuts.*

belong to the front class can be partitioned into two subsets: the set R of rectangles that are vertical (and parallel to the right face of B) and the set T of rectangles that are horizontal (and parallel to the top face of B); see Figure 3.1. Let e be the edge of B that lies on the z-axis, and let e' be the edge of B that lies on the y-axis. The intersection of each rectangle in R with the back face of B is a segment parallel to the z-axis. Let $\bar{r}$ denote the projection of such a segment r onto the z-axis, and let $\bar{R} = \{\bar{r} \mid r \in R\}$. Let $z_1 < z_2 < \cdots < z_{k-1}$ be the endpoints of intervals in $\bar{R}$ that lie in the interior of e but not in the interior of any interval of $\bar{R}$. Note that $k-1$ may be less than $2|R|$, as in Figure 3.1, if some of the projected segments overlap. If no two intervals in $\bar{R}$ share an endpoint, then $\{z_1, z_2, \ldots, z_{k-1}\}$ is the set of vertices of the union of the intervals in $\bar{R}$; otherwise, $\{z_1, z_2, \ldots, z_{k-1}\}$ includes endpoints common to two intervals in $\bar{R}$ and not lying in the interior of any other interval in $\bar{R}$. Similarly, for each rectangle t in the set T, let $\bar{t}$ be the projection of t onto the y-axis, and let $\bar{T} = \{\bar{t} \mid t \in T\}$. Let $y_1 < y_2 < \cdots < y_{l-1}$ be the endpoints of intervals in $\bar{T}$ that lie in the interior of e' but not in the interior of any interval of $\bar{T}$.

We divide B into kl boxes by drawing the planes $z = z_i$ for $1 \leq i < k$ and the planes $y = y_j$ for $1 \leq j < l$; see Figure 3.1. This decomposition of B into kl boxes can easily be constructed in a treelike fashion by performing $(k-1)(l-1)$ cuts. We refer to these cuts as *α-cuts*. If any resulting box has a free rectangle, we divide that box into two boxes by applying the free cut along the free rectangle. Let $\mathcal{C}$ be the set of boxes into which B is partitioned in this manner. We can prove the following lemma about the decomposition of B into $\mathcal{C}$.

LEMMA 3.1. *The set $\mathcal{C}$ of boxes formed by the above process satisfies the following properties:*

(i) *Each box C in $\mathcal{C}$ has only two classes of rectangles,*

(ii) *there are at most $26\alpha^2 n$ boxes in $\mathcal{C}$, and*

(iii) $\sum_{C \in \mathcal{C}} |S_C| \leq 16\alpha n$.

Proof. Let z_0 and z_k, where $z_0 < z_k$, be the endpoints of e, the edge of the box B that lies on the z-axis. Similarly, define y_0 and y_l, where $y_0 < y_l$, to be the endpoints of the edge of B that lies on the y-axis.

(i) Let C be a box in $\mathcal{C}$. If C does not contain a rectangle from $T \cup R$, the claim is obvious since the rectangles in T and R together constitute the front class. Suppose C contains rectangles from the set R. Rectangles in R belong to the front class and are parallel to the right face of B. We claim that C cannot have any rectangles from the right class. Indeed, consider an edge of C parallel to the z-axis. The endpoints of this edge have z-coordinates z_i and z_{i+1}, for some $0 \leq i < k$. Since C contains a rectangle from R, by construction, the interval $z_i z_{i+1}$ must be covered by projections of rectangles in R (onto the z-axis). If C also contains a rectangle r belonging to the right class, then let $z_i < z < z_{i+1}$ be the z-coordinate of a point in $r \cap C$. Let r' be a rectangle in R whose projection on the z-axis contains z. Since both r and r' are long with respect to C, the interiors of r and r' intersect, which contradicts the fact that the rectangles in S are nonintersecting. A similar proof shows that if C contains rectangles from T, then C does not contain any rectangle in the top class.

(ii) We first show that both k and l are at most $2\lfloor\alpha\rfloor + 3$. Let a (resp., b, c) denote the length of the edges of B parallel to the z-axis (resp., y-axis, x-axis). By assumption, $a, b \leq c$. Let $r \in R$ be a rectangle with dimensions β and γ, where $\beta \leq \gamma$. Consider $\bar{r}$, the projection of r onto the z-axis. Suppose that $\bar{r} \subseteq z_i z_{i+1}$, for some $0 < i < k-1$, i.e., $\bar{r}$ lies in the interior of the edge e of B lying on the z-axis. Since r is a rectangle in the front class and is parallel to the right face of B, we have $\beta \leq a \leq c = \gamma$. If $\hat{r}$, the rectangle supporting r in the set S, has dimensions $\hat{\beta}$ and $\hat{\gamma}$, where $\hat{\beta} \leq \hat{\gamma} \leq \alpha\hat{\beta}$, we have $\beta = \hat{\beta}$ (since $\bar{r} \subseteq \mathrm{int}(e)$) and $\gamma \leq \hat{\gamma}$. (If i is 0 or $k-1$, we cannot claim that $\beta = \hat{\beta}$; in these cases, it is possible that $\beta \ll \hat{\beta}$.) See Figure 3.2. Thus, we obtain

$$a \leq c = \gamma \leq \hat{\gamma} \leq \alpha\hat{\beta} = \alpha\beta.$$

It follows that the length of the interval $\bar{r}$, and hence the length of $z_i z_{i+1}$, is at least a/α. Since every alternate interval $z_i z_{i+1}, 0 < i < k-1$ contains the projection of at least one rectangle of R, we have $k \leq 2\lfloor\alpha\rfloor + 3$. In a similar manner, $l \leq 2\lfloor\alpha\rfloor + 3$. Hence, the planes $z = z_i, 1 \leq i \leq k-1$ and the planes $y = y_j, 1 \leq j \leq l-1$ partition B into at most $kl \leq (2\lfloor\alpha\rfloor + 3)^2$ boxes. Each such box C can contain at most n rectangles. Hence, at most n free cuts can be made inside C. The free cuts can divide C into at most $n+1$ boxes. This implies that the set $\mathcal{C}$ has at most $kl(n+1) \leq 26\alpha^2 n$ boxes.

(iii) Each rectangle r in S_B is cut into at most kl pieces. The edges of these pieces

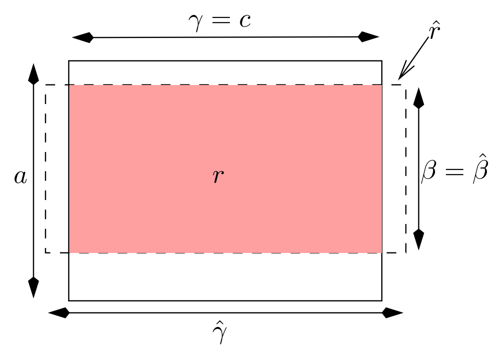


FIG. 3.2. *Projections of $\hat{r}$ (the dashed rectangle), $r = \hat{r} \cap B$ (the shaded rectangle), and the right face of B onto the zx-plane.*

form an arrangement on r. Each face of the arrangement is one of the at most kl rectangles that r is partitioned into. Only $2(k + l - 2)$ faces of the arrangement have an edge on the boundary of r. All other faces can be used as free cuts. Hence, after all possible free cuts are made in the boxes into which B is divided by the $(k-1)(l-1)$ cuts, only $2(k + l - 2)$ pieces of each rectangle in S_B survive. This proves that $\sum_{C \in \mathcal{C}} |S_C| \leq 16\alpha n$. □

Remark. The only place in the whole algorithm where we use the fatness of the rectangles in S is in the proof of Lemma 3.1. If the rectangles in S are thin, then Lemma 3.1(ii) is not true; both k and l can be $\Omega(n)$.

If S_B contains a short rectangle, the α-cuts partition the short rectangle into a constant number of pieces. Hence, Lemma 3.1(ii) and 3.1(iii) hold even when S_B contains short rectangles.

3.2. BSPs for two classes of long rectangles. Let C be one of the boxes into which B is partitioned by the α-cuts. We now present an algorithm for constructing a BSP for the set of clipped rectangles S_C, which has only two classes of long rectangles. We recursively apply the following steps to each of the boxes produced by the algorithm until no box contains a rectangle.

1. If S_C has a free rectangle, we use the free cut containing that rectangle to split C into two boxes.
2. If S_C has two classes of rectangles, we use Lemma 2.2 (with $R = S$ and $P = \emptyset$) to split C into at most three boxes, using at most two parallel free cuts.
3. If S_C has only one class of rectangles, we split C into two by a plane, using Lemma 2.1 (with $R = S$ and $P = \emptyset$).

In steps 2 and 3, since $P = \emptyset$, we can use any value of w in Lemmas 2.2 and 2.1.

We first analyze the algorithm for two classes of long rectangles. The BSP produced has the following structure: If step 3 is executed at a node v, then step 2 is not invoked at any descendant of v. Note that the cutting planes used in steps 1 and 2 do not intersect any rectangle of S_C, so only the cuts made in step 3 increase the number of rectangles. Hence, repeated execution of steps 1 or 2 on S_C constructs a top subtree $\mathcal{T}_C$ of the BSP with $O(|S_C|)$ nodes such that each leaf in $\mathcal{T}_C$ has only one class of rectangles and the total number of rectangles in all the leaves is at most $|S_C|$. The operations at a box D in $\mathcal{T}_C$ involve determining the cuts to be made at D, partitioning D according to these cuts, identifying the resulting free rectangles and applying free cuts containing them, and partitioning the rectangles in S_D into the new boxes.

Since we assume that we have sorted the vertices of the rectangles in S at the very beginning, Lemmas 2.1 and 2.2 imply that the cuts to be made at B can be determined in $O(|S_D|)$ time. The free cuts resulting after partitioning D according to these cuts are parallel to each other. Hence, all free cuts can be applied in $O(|S_D|)$ time. Further, the number of rectangles at each child of D is at most $2|S_D|/3$. Hence, $\mathcal{T}_C$ can be constructed in $O(|S_C|\log|S_C|)$ time. At each leaf v of the tree $\mathcal{T}_C$, recursive invocations of steps 1 and 3 build a BSP of size $O(|S_v|\log|S_v|)$ in $O(|S_v|\log|S_v|)$ time (see [25] for details). Since $\sum_v S_v \leq |S_C|$, where the sum is taken over all leaves v of $\mathcal{T}_C$, the total size of the BSP constructed inside C is $O(|S_C|\log|S_C|)$. It also follows that the BSP inside C can be constructed in $O(|S_C|\log|S_C|)$ time.

We now analyze the overall algorithm for long rectangles. The algorithm first applies the α-cuts to the rectangles in S_B, as described in section 3.1. Consider the set of boxes $\mathcal{C}$ produced by the α-cuts. Each of the boxes in $\mathcal{C}$ contains only two classes of rectangles (by Lemma 3.1(i)). In view of the above discussion, for each box $C \in \mathcal{C}$, we can construct a BSP for S_C of size $O(|S_C|\log|S_C|)$ in time $O(|S_C|\log|S_C|)$. Lemma 3.1(ii) and (iii) imply that the total size of the BSP is

$$O(n)+\sum_{C\in\mathcal{C}} O(|S_C|\log|S_C|) = O(n\log n).$$

The time spent in building the BSP is also $O(n\log n)$. We can now state the following theorem.

THEOREM 3.2. *Let S be a set of n fat rectangles and B a box so that all rectangles in S are long with respect to B. An $O(n\log n)$-size BSP for the clipped rectangles S_B can be constructed in $O(n\log n)$ time. The constants of proportionality in the big-oh terms are linear in α^2, where α is the maximum aspect ratio of the rectangles in S.*

Remark. We can show that the height of the BSP constructed by the above algorithm is $O(\log n)$. We can also modify our algorithm to construct a BSP of size $O(n)$ for n long rectangles as follows: If a box C has two classes of long rectangles, we apply step 1 or 2 of the previous algorithm. If the rectangles in C belong to one class, we use the algorithm of Paterson and Yao for constructing BSPs for orthogonal segments in the plane [26] to construct a BSP of linear size for S_C. However, the height of the BSP can now be $\Omega(n)$ in the worst case.

4. BSPs for fat rectangles.

We now describe our main algorithm for constructing a BSP for a set S of n fat nonintersecting rectangles, in which we simultaneously simulate the algorithm for long fat rectangles presented in section 3 and partition the vertices of the rectangles in S. The algorithm proceeds in rounds. Each round simulates a few steps of the algorithm for long rectangles and partitions the vertices of the rectangles in S into a small number of sets of approximately equal size. At the beginning of the ith round, for $i > 0$, the algorithm has a top subtree $\mathcal{B}_i$ of the BSP for S. Let Q_i be the set of boxes associated with the leaves of $\mathcal{B}_i$ containing at least one rectangle. The initial tree $\mathcal{B}_1$ consists of one node and Q_1 consists of one box that contains all the input rectangles. Our algorithm maintains the invariant that for each box $B \in Q_i$, all long rectangles in S_B are nonfree. If Q_i is empty, we are done. Otherwise, in the ith round, for each box $B \in Q_i$, we construct a top subtree $\mathcal{T}_B$ of the BSP for the set S_B and attach it to the corresponding leaf of $\mathcal{B}_i$. This gives us the new top subtree $\mathcal{B}_{i+1}$. Thus, it suffices to describe how to build the tree $\mathcal{T}_B$ on a box B during a round.

Let $F \subseteq S_B$ be the set of rectangles that are long with respect to B. Set $f = |F|$, and let k be the number of vertices of rectangles in S_B that lie in the interior of B

(note that each such vertex is a vertex of an original rectangle in the input set S). By assumption, all rectangles in F are nonfree. We choose a parameter a, which remains fixed throughout the round. We pick

$$a = 2^{\sqrt{\log(f+k)}}$$

to optimize the size of the BSP that the algorithm creates. We now describe the ith round in detail. See Figure 4.1 for an outline of $\mathcal{B}$'s structure.

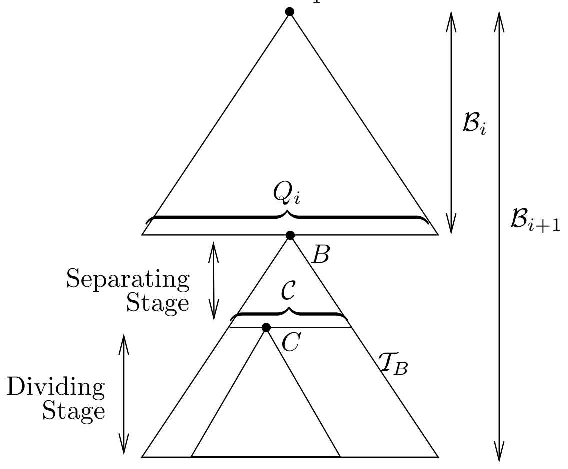


FIG. 4.1. *Overall structure of $\mathcal{B}$.*

If $k = 0$ (i.e., if all rectangles in S_B are long), we use Theorem 3.2 to construct a BSP for S_B. Otherwise, we perform a sequence of cuts in two stages that partition B as follows:

Separating stage. We apply the α-cuts, as described in section 3.1. We make these cuts with respect to the rectangles in F, i.e., we consider only those rectangles of S_B that are long with respect to B. Let $\mathcal{C}$ be the set of boxes into which B is partitioned by the α-cuts.

Dividing stage. We refine each box C in $\mathcal{C}$ by applying cuts similar to the ones made in section 3.2, as described below. Let k_C denote the number of vertices of rectangles in S_C that lie in the interior of C. Recall that F_C is the set of rectangles in F that are clipped within C. We recursively invoke the dividing stage until $|F_C| + 2ak_C \leq (f + ak)/a$ and S_C does not contain any free rectangles.

1. If C has any free rectangle, we use the free cut containing that rectangle to split C into two boxes.
2. If the rectangles in F_C belong to two classes, let P_C denote the set of vertices of the rectangles in S_C that lie in the interior of C. We apply at most two parallel free cuts that satisfy Lemma 2.2, with $R = F, P = P_C$, and $w = 2a$.
3. If the rectangles in F_C belong to just one class, we apply one cut using Lemma 2.1, with $R = F, P = P_C$, and $w = 2a$.

The cuts introduced during the dividing stage can be made in a treelike fashion. At the end of the dividing stage, we have a set of boxes so that for each box D in this set, S_D does not contain any free rectangle and $|F_D| + ak_D \leq (f + ak)/a$. Notice that as we apply cuts in C and in the resulting boxes, rectangles that are short with respect to C may become long with respect to the new boxes. We ignore these new long rectangles until the next round, unless they induce a free cut.

5. Analysis of the algorithm. We now analyze the size of the BSP constructed by the algorithm and the time complexity of the algorithm. In a round, the algorithm constructs a top subtree $\mathcal{T}_B$ of the BSP for the set of clipped rectangles S_B. Recall that F is the set of rectangles that are long with respect to B, $f = |F|$, and k is the number of vertices of rectangles in S_B that lie in the interior of B. For a node C in $\mathcal{T}_B$, recall that k_C denotes the number of vertices of rectangles in S_C that lie in the interior of C.

We now define some more notation that we need for the analysis. For a node C in $\mathcal{T}_B$, let $\mathcal{T}_C$ be the subtree of $\mathcal{T}_B$ rooted at C, L_C be the set of leaves in $\mathcal{T}_C$, ϕ_C be the number of long rectangles in F_C (recall that F_C is the set of rectangles in F that intersect C and are clipped within C), and ν_C be the number of long rectangles in $S_C \setminus F_C$ (recall that a rectangle in $S_C \setminus F_C$ is a portion of a rectangle in S_B that is short with respect to B). For a box D corresponding to a leaf of $\mathcal{T}_B$, let f_D be the number of long rectangles in S_D. Note that f_D counts both the "old" long rectangles in F_D (pieces of rectangles that were long with respect to B) and the "new" long rectangles in $S_D \setminus F_D$ (pieces of rectangles that were short with respect to B, but became long with respect to D due to the cuts made during the round); $f_D = \phi_D + \nu_D$.

In a round, the separating stage first splits B into a set of boxes $\mathcal{C}$. For each box $C \in \mathcal{C}$, F_C has only two classes of long rectangles. The algorithm then executes the dividing stage on each such box C. As in the case of the algorithm for long rectangles (see section 3), the subtree constructed in C has the following property: if step 4 is executed at a node v, then step 4 is not executed at any descendent of v. In Lemma 5.1, we prove a bound on the total number of long rectangles at each leaf of $\mathcal{T}_B$. For a box $C \in \mathcal{C}$, we bound the number of long rectangles at the leaves of $\mathcal{T}_C$ in Lemmas 5.2 and 5.3. In Lemma 5.4, we prove a bound on the size of the tree $\mathcal{T}_B$. Finally, we use these lemmas to establish bounds on the size of the BSP constructed by our algorithm and the running time of our algorithm (see Theorem 5.5).

LEMMA 5.1. *For a box D associated with a leaf of $\mathcal{T}_B$,*

$$f_D + 2ak_D \leq \frac{f + 2ak}{a}.$$

Proof. We know that ν_D is at most k (since a rectangle in $S_D \setminus F_D$ must be a piece of a rectangle short with respect to B, and there are at most k short rectangles in B). Since $f_D + 2ak_D \leq \phi_D + 2ak_D + \nu_D$ and $\phi_D + 2ak_D \leq (f + ak)/a$ (by construction), the lemma follows. ◻

LEMMA 5.2. *Let C be a box associated with a node in $\mathcal{T}_B$. If all rectangles in F_C belong to one class, then*

$$\sum_{D \in L_C} f_D \leq 2\phi_C + 2\nu_C \max\left\{\frac{2\,(\phi_C + ak_C)}{\mu}, 1\right\} + 4k_C\left(\frac{\phi_C + ak_C}{\mu}\right),$$

where $\mu = (f + ak)/a$.

Proof. Assume, without loss of generality, that all rectangles in F_C belong to the top class, and let g be the top face of C. By Lemma 2.1(i), g contains an edge of every rectangle in F_C. Let ρ_C be the number of vertices of the nonfree long rectangles in F_C that lie in the interior of g; obviously, $\phi_C \le \rho_C \le 2\phi_C$. Set

$$\Phi(\rho_C, \nu_C, k_C) = \max \sum_{D \in L_C} f_D,$$

where the maximum is taken over all boxes C and over all sets S of rectangles with ρ_C vertices of rectangles in F_C lying in the interior of the top face of C, ν_C long rectangles in $S_C \setminus F_C$, and k_C vertices in the interior of C. We claim that

$$\Phi(\rho_C, \nu_C, k_C) \le \rho_C + 2\nu_C \max\left\{\frac{\rho_C + 2ak_C}{\mu}, 1\right\} + 2k_C\left(\frac{\rho_C + 2ak_C}{\mu}\right), \tag{5.1}$$

which implies the lemma, because $\rho_C \le 2\phi_C$.

Note that if S_C contains $m \ge 1$ free rectangles, we apply the free cuts containing these rectangles to partition C (by repeatedly invoking step 4 of the dividing stage) until the resulting boxes do not contain any free rectangle. The free cuts partition C into a set $\mathcal{E}$ of $m+1$ boxes. Since we have created the boxes in $\mathcal{E}$ using free cuts,

$$\rho_E + 2ak_E \le \rho_C + 2ak_C, \qquad \text{for any box } E \text{ in } \mathcal{E}, \tag{5.2}$$

$$\sum_{E \in \mathcal{E}} \nu_E \le \nu_C, \qquad \sum_{E \in \mathcal{E}} k_E \le k_C, \qquad \sum_{E \in \mathcal{E}} \rho_E \le \rho_C. \tag{5.3}$$

These inequalities imply that if (5.1) holds for each box in $\mathcal{E}$, then (5.1) holds for C as well. Therefore, we prove (5.1) for all boxes C such that F_C contains only one class of rectangles and S_C does not contain any free rectangle. We proceed by induction on $\rho_C + 2ak_C$.

Base case. $0 \le \rho_C + 2ak_C \le \mu$. Since $0 \le \rho_C + 2ak_C \le \mu$ and S_C does not contain any free rectangle, C is a leaf of $\mathcal{T}_B$, i.e., $L_C = \{C\}$. We have

$$\Phi(\rho_C, \nu_C, k_C) = \sum_{D \in L_C} f_D = f_C = \phi_C + \nu_C \le \rho_C + \nu_C, \tag{5.4}$$

which implies (5.1).

Induction step. $\rho_C + 2ak_C > \mu$. In this case, C is split into two subboxes C_1 and C_2 by a cutting plane h. Since $\sum_{D \in L_C} f_D = \sum_{D \in L_{C_1}} f_D + \sum_{D \in L_{C_2}} f_D$,

$$\Phi(\rho_C, \nu_C, k_C) = \Phi(\rho_{C_1}, \nu_{C_1}, k_{C_1}) + \Phi(\rho_{C_2}, \nu_{C_2}, k_{C_2}),$$

where $k_{C_1} + k_{C_2} \le k_C$ and $\rho_{C_1} + \rho_{C_2} \le \rho_C$.

Note that h does not contain a free rectangle. For $i = 1, 2$, each long rectangle in $S_{C_i} \setminus F_{C_i}$ is contained either in a long rectangle in $S_C \setminus F_C$ or in a short rectangle in S_C. Since h intersects each rectangle in S_C at most once and a short rectangle intersected by h is divided into one short and one long rectangle,

$$\nu_{C_1} + \nu_{C_2} \le 2\nu_C + k_C. \tag{5.5}$$

By Lemma 2.1(ii), we have

$$\rho_{C_i} + 2ak_{C_i} \le \frac{\rho_C + 2ak_C}{2}, \qquad \text{for } i = 1, 2. \tag{5.6}$$

Let $\mathcal{E}_1$ (resp., $\mathcal{E}_2$) be the set of boxes obtained by applying all the free cuts in S_{C_1} (resp., S_{C_2}) in step 4 of the dividing stage. Clearly,

$$\Phi(\rho_C, \nu_C, k_C) = \sum_{E \in \mathcal{E}_1} \Phi(\rho_E, \nu_E, k_E) + \sum_{E \in \mathcal{E}_2} \Phi(\rho_E, \nu_E, k_E).$$

We consider two cases.

Case (i). $\mu < \rho_C + 2ak_C \leq 2\mu$. For each $i = 1, 2$

$$\rho_{C_i} + 2ak_{C_i} \leq \frac{\rho_C + 2ak_C}{2} \leq \mu.$$

As a result, all boxes in $\mathcal{E}_1$ and $\mathcal{E}_2$ are leaves of $\mathcal{T}_B$. Using (5.3) and (5.4), we obtain

$$\begin{aligned}\Phi(\rho_C, \nu_C, k_C) &\leq \sum_{E \in \mathcal{E}_1} f_E + \sum_{E \in \mathcal{E}_2} f_E \leq \sum_{E \in \mathcal{E}_1} (\rho_E + \nu_E) + \sum_{E \in \mathcal{E}_2} (\rho_E + \nu_E) \\ &\leq \rho_{C_1} + \nu_{C_1} + \rho_{C_2} + \nu_{C_2}.\end{aligned}$$

It now follows from (5.5) that

$$\Phi(\rho_C, \nu_C, k_C) \leq \rho_C + 2\nu_C + k_C, \tag{5.7}$$

which implies (5.1), because $\rho_C + 2ak_C > \mu$.

Case (ii). $\rho_C + 2ak_C > 2\mu$. For any box E in $\mathcal{E}_1 \cup \mathcal{E}_2$, by (5.2) and (5.6),

$$\max\left\{\frac{\rho_E + 2ak_E}{\mu}, 1\right\} \leq \max\left\{\frac{\rho_C + 2ak_C}{2\mu}, 1\right\} = \frac{\rho_C + 2ak_C}{2\mu}.$$

By (5.2) and the induction hypothesis,

$$\begin{aligned}\Phi(\rho_C, \nu_C, k_C) \leq& \sum_{E \in \mathcal{E}_1} \left(\rho_E + 2\nu_E \left(\frac{\rho_C + 2ak_C}{2\mu}\right) + 2k_E \left(\frac{\rho_C + 2ak_C}{2\mu}\right)\right) \\ &+ \sum_{E \in \mathcal{E}_2} \left(\rho_E + 2\nu_E \left(\frac{\rho_C + 2ak_C}{2\mu}\right) + 2k_E \left(\frac{\rho_C + 2ak_C}{2\mu}\right)\right) \\ \leq& (\rho_{C_1} + \rho_{C_2}) + 2\left(\nu_{C_1} + \nu_{C_2}\right)\left(\frac{\rho_C + 2ak_C}{2\mu}\right) \\ &+ 2\left(k_{C_1} + k_{C_2}\right)\left(\frac{\rho_C + 2ak_C}{2\mu}\right).\end{aligned}$$

Using (5.5), we obtain

$$\begin{aligned}\Phi(\rho_C, \nu_C, k_C) &\leq \rho_C + 2\left(2\nu_C + k_C\right)\left(\frac{\rho_C + 2ak_C}{2\mu}\right) + 2k_C\left(\frac{\rho_C + 2ak_C}{2\mu}\right) \\ &= \rho_C + 2\nu_C\left(\frac{\rho_C + 2ak_C}{\mu}\right) + 2k_C\left(\frac{\rho_C + 2ak_C}{\mu}\right),\end{aligned}$$

which implies (5.1). ☐

LEMMA 5.3. *Let C be a box associated with a node in $\mathcal{T}_B$. If all rectangles in F_C belong to two classes, then*

$$\sum_{D \in L_C} f_D \leq O\left(\phi_C + (\nu_C + k_C)\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\right),$$

where $\mu = (f + ak)/a$.

Proof. Similar to the proof of Lemma 5.2. See the Appendix for details. ☐

LEMMA 5.4. *The tree $\mathcal{T}_B$ constructed on box B in a round has the following properties:*

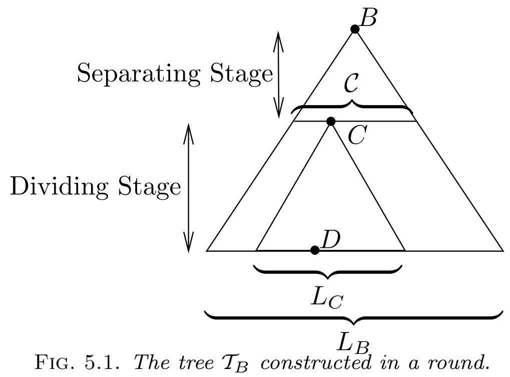


FIG. 5.1. *The tree $\mathcal{T}_B$ constructed in a round.*

(i) $\sum_{D \in L_B} k_D \leq k$,

(ii) $\sum_{D \in L_B} f_D = O(f + a^3 k)$, *and*

(iii) $|\mathcal{T}_B| = O((f + a^3 k) \log a)$.

Proof. The bound on $\sum_{D \in L_B} k_D$ is obvious, since each vertex in the interior of S_B lies in the interior of at most one box of L_B. Next, we use Lemma 5.3 to prove a bound on $\sum_{D \in L_B} f_D$.

Let $\mathcal{C}$ be the set of boxes into which B is partitioned by the separating stage; see Figure 5.1. Obviously, $\sum_{D \in L_B} f_D = \sum_{C \in \mathcal{C}} \sum_{D \in L_C} f_D$. For each box $C \in \mathcal{C}$, Lemma 3.1(i) implies that all rectangles in F_C belong to at most two classes. Hence, by Lemma 5.3,

$$\sum_{D \in L_B} f_D \leq \sum_{C \in \mathcal{C}} O\left(\phi_C + (\nu_C + k_C)\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\right).$$

Arguing as in the proof of Lemma 3.1(3), we can show that $\sum_{C \in \mathcal{C}} \phi_C = O(f)$ and that $\sum_{C \in \mathcal{C}} \nu_C = O(k)$. We also know that $\sum_{C \in \mathcal{C}} k_C \leq k$ and $\mu = (f + ak)/a$. Therefore,

$$\sum_{D \in L_B} f_D = O\left(f + k\left(\frac{f + 2ak}{\mu}\right)^3\right) = O\left(f + a^3 k\right).$$

We now sketch the proof that $|\mathcal{T}_B| = O((f + a^3 k) \log a)$. Following the same argument as in the proof of Lemma 3.1(2), we can show that the separating stage creates a tree with $O(f + k)$ nodes. We now count the number of nodes in $\mathcal{T}_B$ that are created by the dividing stage. Let $D \in \mathcal{T}_B$ be such a node. If D is partitioned by a cut containing a free rectangle (i.e., step 4 of the dividing stage is invoked at D), we charge D to its nearest ancestor $C \in \mathcal{T}_B$ such that C is not partitioned by a free cut (i.e., step 4 or 4 of the dividing stage is executed at C). Otherwise, we charge a cost of 1 to D itself. Let C be a node in $\mathcal{T}_B$ that is not partitioned by a free cut. Since a free rectangle can be created only by partitioning a long rectangle, the cut used to partition C creates $O(\phi_C + \nu_C)$ free rectangles, which implies that C is charged $O(\phi_C + \nu_C + 1)$ times by the above argument. Hence,

$$|\mathcal{T}_B| = O\left(\sum_C (\phi_C + \nu_C + 1)\right),$$

where C ranges over all nodes in $\mathcal{T}_B$ where step 4 or 4 of the dividing stage is executed. By following an inductive argument similar to the ones used to prove Lemmas 5.2 and 5.3, we can show that $|\mathcal{T}_B| = O((f + a^3k)\log a)$. Informally, the charging scheme compresses $\mathcal{T}_B$ by collapsing all nodes that are split by free cuts. Lemma 2.1 and 2.2 imply that the height of the compressed tree is $O(\log a)$. We can show that $\sum_C(\phi_C + \nu_C + 1)$ is roughly the product of the height of the tree and $\sum_D f_D$, the total number of long rectangles intersecting the leaves of the tree. $\square$

We now present our main result regarding the performance of our algorithm.

THEOREM 5.5. *Given a set S of n rectangles in $\mathbb{R}^3$ such that the aspect ratio of each rectangle in S is bounded by a constant $\alpha \geq 1$, we can construct a BSP of size $n2^{O(\sqrt{\log n})}$ for S in time $n2^{O(\sqrt{\log n})}$. The constants of proportionality in the big-oh terms are linear in* $\log \alpha$.

Proof. We first bound the size of the BSP constructed by the algorithm. Let $S(f,k)$ denote the maximum size of the BSP produced by the algorithm for a box B that contains f long rectangles and k vertices in its interior. If $k = 0$, Theorem 3.2 implies that $S(f,k) = O(f \log f)$. For $k > 0$, by Lemma 5.4(iii), we construct the subtree $\mathcal{T}_B$ on B of size $O((f + a^3k)\log a)$ in one round, and recursively construct subtrees for each box in the set of leaves L_B. Therefore, there exist constants $c_1, c_2, c_3 > 0$ so that the size $S(f,k)$ satisfies the following recurrence:

$$S(f,k) \leq \begin{cases} c_1 f \log f & \text{for} \quad k = 0, \\ \displaystyle\sum_{D \in L_B} S(f_D, k_D) + c_2(f + a^3k)\log a & \text{for} \quad k > 0, \end{cases}$$

where

$$f_D + 2ak_D \leq \frac{f + 2ak}{a}$$

for every box D in L_B (by Lemma 5.1), and

$$\sum_D k_D \leq k, \qquad \sum_D f_D \leq c_3(f + a^3k)$$

(by Lemma 5.4(i) and 5.4(ii). Using induction on $f + 2ak$, we can prove that the solution to the above recurrence is

$$S(f,k) = (f + k)2^{O(\sqrt{\log(f+k)})},$$

where the constant of proportionality is linear in $\log \alpha$. Intuitively, the algorithm constructs the BSP for B in $O(\log a) = O(\sqrt{\log(f + k)})$ rounds, and the total number of long rectangles increases roughly by a constant factor in each round. The $n2^{O(\sqrt{\log n})}$ bound on the size of the BSP constructed by the algorithm follows, since $f \leq n$ and $k \leq 4n$ at the beginning of the first round.

We now bound the running time of our algorithm. Recall that we initially sorted the vertices of the rectangles in S by x-, y-, and z-coordinates. Suppose S_B does not contain a free rectangle. By Lemmas 2.1 and 2.2, the cuts to be made at B can be determined in $O(|S_B|)$ time. Suppose C is a box obtained by partitioning B according to these cuts; we can easily obtain the sorted order of the vertices of the rectangles in S_C from the sorted order in B. Let $\mathcal{E}$ be the set of boxes obtained by applying C using all the free rectangles in S_C. Since the free rectangles in S_C are parallel to each other, we can partition the rectangles in S_C among the boxes in $\mathcal{E}$ in $O(|S_C|)$ time. Therefore, we can construct the tree representing the partition of B into the set of boxes E in $O(|S_B|)$ time. Hence, we obtain the same $n2^{O(\sqrt{\log n})}$ bound for the running time of the algorithm. ∎

Remark. We can modify our algorithm to prove that the height of the BSP constructed is $O(\log n)$: if S_B contains free rectangles, we assign appropriate weights to the free rectangles, and partition B using the weighted median of the free rectangles. We leave the details to the reader. Paterson and Yao [25] use a similar idea to bound the height of BSP they construct for segments in the plane.

6. Extensions.

In this section, we extend the algorithm of section 4 to the following two cases: (i) some of the input rectangles are thin and (ii) some of the input polygons are triangles.

6.1. Fat and thin rectangles.

Let us assume that the input $S = F \cup T$ has n rectangles, consisting of $m \geq 1$ thin rectangles in T and $n - m$ fat rectangles in F. We first describe our algorithm and then construct a set of rectangles for which any BSP has size $\Omega(n\sqrt{m})$. The algorithm we use now is very similar to the algorithm for fat rectangles. Given a box B, let f be the number of long rectangles in F_B, k the number of vertices of rectangles in F_B that lie in the interior of B, and t the number of rectangles in T_B. We fix a parameter $a = 2^{\sqrt{\log(f+k)}}$ and perform the following steps:

1. If S_B contains a free rectangle, we use the corresponding free cut to split B into two boxes.
2. If $k = t = 0$, we use the algorithm for long rectangles to construct a BSP for the set of clipped rectangles S_B.
3. If $t \geq (f + k)$, we use the algorithm by Paterson and Yao for orthogonal rectangles in $\mathbb{R}^3$ to construct a BSP for S_B [26].
4. If $(f + k) > t$, we perform one round of the algorithm described in section 4, with the difference that we also use thin rectangles to make free cuts.

This algorithm is recursively invoked on all the resulting subboxes. Let $S(f, k, t)$ be the maximum size of the BSP produced by this algorithm for a box B with k vertices in its interior, f long rectangles in F_B, and t thin rectangles in T_B. Note that in section 5, during the analysis of a round, we did not use the fact that the short rectangles were fat. As a result, Lemmas 5.2 and 5.3 hold for step 6.1 above, with $\nu_C + k_C + t_C$ replacing the term $\nu_C + k_C$. We can similarly extend Lemma 5.4 to obtain the following recurrence for $S(f, k, t)$. Here D ranges over all the boxes that B is divided into by a round of cuts, as described above in step 6.1.

$$S(f,k,t) = \begin{cases} O(f \log f), & \text{for} \quad k = t = 0, \\ O(t\sqrt{t}), & \text{for} \quad t \geq f + k, \\ \sum_D S(f_D, k_D, t_D) + O(f \log a + a^3 k + a^3 t), & \text{for} \quad f + k > t, \end{cases}$$

where $\sum_D k_D \leq k$, $f_D + 2ak_D \leq (f + 2ak)/a$, $\sum_D f_D = O(f + a^3 k)$, and $\sum_D t_D = O(a^3 t)$.

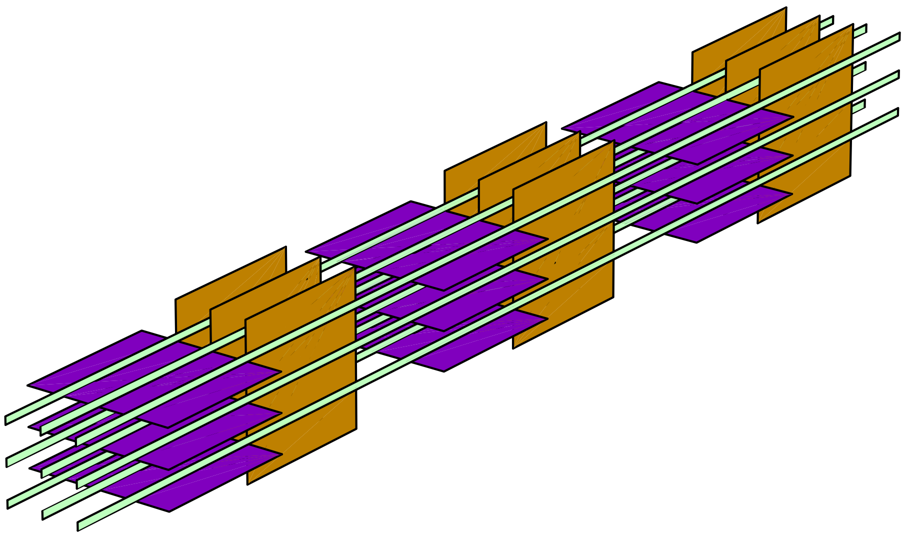


FIG. 6.1. *Lower bound construction for thin and fat rectangles.*

We can analyze this recurrence as in section 5 and show that its solution is

$$S(f,k,t) = (f+k)\sqrt{t}2^{O(\sqrt{\log(f+k)})},$$

where the constant of proportionality is linear in $\log \alpha$. The following theorem is immediate.

THEOREM 6.1. *Let S be a set of n rectangles in $\mathbb{R}^3$, of which $m \geq 1$ are thin. A BSP of size $n\sqrt{m}2^{O(\sqrt{\log n})}$ for S can be constructed in $n\sqrt{m}2^{O(\sqrt{\log n})}$ time. The constants of proportionality in the big-oh terms are linear in $\log \alpha$, where α is the maximum aspect ratio of the fat rectangles.*

We can show that Theorem 6.1 is almost optimal by constructing a set of n rectangles of which m are thin, for which any BSP has size $\Omega(n\sqrt{m})$. Recall that there exists a set of m thin rectangles in $\mathbb{R}^3$ for which any BSP has size $\Omega(m\sqrt{m})$ [26]. To complete the proof of the lower bound, we now exhibit a set $S = T \cup F$ of n rectangles, where T is a set of m thin rectangles and F is a set of $n-m$ fat rectangles, for which any BSP has size $\Omega((n-m)\sqrt{m})$. The rectangles in T are arranged in a $\sqrt{m}\times\sqrt{m}$ grid; each rectangle in T is a segment of length $n-m+5(n-m)/(2\sqrt{m})+1$ perpendicular to the yz-plane. The rectangles in F are divided into $(n-m)/(2\sqrt{m})$ sets, each consisting of $2\sqrt{m}$ squares with side length $\sqrt{m}+2$. Each set consists of $\sqrt{m}$ squares parallel to the xy-plane and $\sqrt{m}$ squares parallel to the xz-plane so that the following three conditions are satisfied (see Figure 6.1).

1. In each set, a square parallel to the xy-plane is at a distance of 2ε from any square parallel to the xz-plane,
2. For any square, the closest square in a different set is at a distance of $1-2\varepsilon$, and
3. For any square, the closest thin rectangle is at a distance of ε.

We can show that there are $(n-m)\sqrt{m}/2$ points such that a cube of side 2ε centered at any such point intersects a thin rectangle t of T and two squares r and s of F. For each such cube ψ, we can show that at least one edge of r, s, or t is crossed in the interior of ψ by a cutting plane of $\mathcal{B}$, which implies that the cutting planes in $\mathcal{B}$ and the edges of the rectangles in S cross at $\Omega((n-m)\sqrt{m})$ points, thus proving the lower bound on the size of $\mathcal{B}$.

6.2. Fat rectangles and triangles. Suppose that $p \geq 1$ polygons in the input S are (nonorthogonal) triangles and that the rest are fat rectangles. To construct a BSP for S, we use nonorthogonal cutting planes; hence, each region is a convex polytope and the intersection of a triangle with a region is a polygon, possibly with more than three edges. We can extend the algorithm of section 6.1 to this case as follows: In step 6.1, we check whether we can make free cuts through the nonorthogonal polygons too. In step 6.1, if the number of triangles at a node is greater than the number of fat rectangles, we use the algorithm of Agarwal et al. for triangles in $\mathbb{R}^3$ to construct a BSP of size quadratic in the number of triangles in near-quadratic time [2]. Proceeding as in the previous section, we can prove the following theorem.

THEOREM 6.2. *A BSP of size* $np2^{O(\sqrt{\log n})}$ *can be constructed in* $np2^{O(\sqrt{\log n})}$ *time for n polygons in $\mathbb{R}^3$, of which $p \geq 1$ are nonorthogonal and the rest are fat rectangles. The constants of proportionality in the big-oh terms are linear in* $\log \alpha$, *where α is the maximum aspect ratio of the fat rectangles.*

Unlike the case of rectangles, the fatness assumption does not help in constructing BSPs of small size for triangles. More specifically, we can show that there exists a set of n fat triangles in $\mathbb{R}^3$ such that any BSP for these triangles has $\Omega(n^2)$ size by modifying Chazelle's construction for proving a quadratic lower bound on the size of convex decompositions of polyhedra in $\mathbb{R}^3$ [9].

7. Conclusions. In this paper, we have studied the problem of constructing BSPs for orthogonal rectangles under the natural assumption that most rectangles are fat. Our result shows that this assumption allows a smaller worst-case size of BSPs. Our algorithm constructs a BSP for any set of orthogonal rectangles; it is only the analysis of the algorithm that depends on the fatness of the input rectangles. We have implemented a variant of our algorithm and compared its performance to that of other known algorithms [18]. Our algorithm is indeed practical: it constructs a BSP of near-linear size on real data sets. It performs better than not only Paterson and Yao's algorithm [26] but also most heuristics described in the literature [4, 16, 30].

We now briefly mention another extension to our algorithms. If the n fat rectangles contain $k \geq 1$ crossing pairs, we can construct a BSP of size $(n+k)\sqrt{k}2^{O(\sqrt{\log n})}$ for these rectangles as follows: for each crossing pair r and s, we partition one of the rectangles (say, r) into two smaller rectangles that do not intersect s. We construct a BSP for the resulting $n + O(k)$ rectangles by invoking our algorithm for a set of fat and thin triangles. We can also construct a set of n fat rectangles with k crossing pairs for which any BSP has size $\Omega(n + k\sqrt{k})$.

It seems very probable that BSPs of a size smaller than $n2^{O(\sqrt{\log n})}$ can be built for n fat rectangles in $\mathbb{R}^3$. The only lower bound we have is the trivial $\Omega(n)$ bound. It would be interesting to see if simple heuristics (e.g., choose the next splitting plane to be one that intersects the smallest number of rectangles) can be proven to construct BSPs of (close to) optimal size. An even more challenging open problem is determining the right assumptions that should be made about the input objects and the graphics display hardware so that provably fast and *practically efficient* algorithms can be developed for doing hidden-surface elimination of these objects.

Appendix. Proof of Lemma 5.3.

LEMMA 5.3. *Let C be a box associated with a node in $\mathcal{T}_B$. If all rectangles in F_C belong to two classes, then*

$$\sum_{D \in L_C} f_D \le O\left(\phi_C + (\nu_C + k_C)\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\right),$$

where $\mu = (f + ak)/a$.

Proof. Let

$$\Psi(\phi_C, \nu_C, k_C) = \max \sum_{D \in L_C} f_D,$$

where the maximum is taken over all boxes C and over all sets S of rectangles with ϕ_C long rectangles in F_C, ν_C long rectangles in $S_C \setminus F_C$, and k_C vertices in the interior of C. The rectangles in F_C belong to at most two classes. We claim that

$$\Psi(\phi_C, \nu_C, k_C) \le 2\phi_C + 5\nu_C \max\left\{\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3, 1\right\} + 6k_C\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3, \tag{A.1}$$

which proves the lemma.

If S_C contains $m \ge 1$ free rectangles, we apply the free cuts containing these rectangles to partition C (by repeatedly invoking step 4 of the dividing stage) until the resulting boxes do not contain any free rectangles. Let $\mathcal{E}$ be the set of boxes into which C is so partitioned. Then

$$\phi_E + 2ak_E \le \phi_C + 2ak_C, \qquad \text{for any box } E \text{ in } \mathcal{E}, \tag{A.2}$$

$$\sum_{E \in \mathcal{E}} \phi_E \le \phi_C \qquad \sum_{E \in \mathcal{E}} \nu_E \le \nu_C \qquad \sum_{E \in \mathcal{E}} k_E \le k_C. \tag{A.3}$$

These inequalities imply that if (A.1) holds for each box in $\mathcal{E}$, then (A.1) holds for C as well. Therefore, we prove (A.1) for all boxes C such that F_C contains at most two classes of rectangles and S_C does not contain any free rectangles. We proceed by induction on $\phi_C + 2ak_C$.

Base case. $\phi_C + 2ak_C \le \mu$. Since $\phi_C + 2ak_C \le \mu$ and S_C does not contain any free rectangles, C is a leaf of $\mathcal{T}_B$. We have

$$\Psi(\phi_C, \nu_C, k_C) = \sum_{D \in L_C} f_D = f_C = \phi_C + \nu_C, \tag{A.4}$$

which implies (A.1).

Induction step. $\phi_C + 2ak_C > \mu$. The cuts made in step 4 of the dividing stage fall into one of two categories (see Lemma 2.2). Note that none of these cuts contains a free rectangle.

Case (i). We divide C into two boxes, C_1 and C_2, using a plane h that does not cross any rectangle in F_C. As a result,

$$\phi_{C_1} + \phi_{C_2} \le \phi_C \qquad \text{and} \qquad k_{C_1} + k_{C_2} \le k_C.$$

Since h intersects each rectangle in S_C at most once, (5.5) holds in this case too; i.e.,

$$\nu_{C_1} + \nu_{C_2} \le 2\nu_C + k_C. \tag{A.5}$$

Lemma 2.2 implies that

$$\phi_{C_i} + 2ak_{C_i} \le \frac{2(\phi_C + 2ak_C)}{3} \qquad \text{for } i = 1, 2. \tag{A.6}$$

Let $\mathcal{E}_1$ (resp., $\mathcal{E}_2$) be the set of boxes obtained by applying all the free cuts in S_{C_1} (resp., S_{C_2}) in step 4 of the dividing stage. Clearly,

$$\Psi(\phi_C, \nu_C, k_C) \le \sum_{E \in \mathcal{E}_1} \Psi(\phi_E, \nu_E, k_E) + \sum_{E \in \mathcal{E}_2} \Psi(\phi_E, \nu_E, k_E).$$

We consider two cases.

(a) $\mu \le \phi_C + 2ak_C \le 3\mu/2$. In this case, by (A.2) and (A.6),

$$\phi_E + 2ak_E \le \frac{2(\phi_C + 2ak_C)}{3} \le \mu,$$

for each box E in $\mathcal{E}_1$ and $\mathcal{E}_2$. Since E does not contain a free rectangle, E is a leaf of $\mathcal{T}_B$. Using (A.5), (A.3), and (A.4), we obtain

$$\begin{aligned}
\Psi(\phi_C, \nu_C, k_C) &\le \sum_{E \in \mathcal{E}_1} f_E + \sum_{E \in \mathcal{E}_2} f_E \\
&\le \sum_{E \in \mathcal{E}_1} (\phi_E + \nu_E) + \sum_{E \in \mathcal{E}_2} (\phi_E + \nu_E) \\
&\le \phi_{C_1} + \nu_{C_1} + \phi_{C_2} + \nu_{C_2} \\
&\le \phi_C + 2\nu_C + k_C,
\end{aligned}$$

which implies (A.1), because $\phi_C + 2ak_C > \mu$.

(b) $\phi_C + 2ak_C > 3\mu/2$. For a box E in $\mathcal{E}_1 \cup \mathcal{E}_2$, by (A.2) and (A.6), we have

$$\max\left\{\left(\frac{\phi_E + 2ak_E}{\mu}\right)^3, 1\right\} \le \max\left\{\left(\frac{2}{3}\left(\frac{\phi_C + 2ak_C}{\mu}\right)\right)^3, 1\right\} = \frac{8}{27}\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3.$$

By the induction hypothesis, and by using (A.3) and (A.5),

$$\begin{aligned}
\Psi(\phi_C, \nu_C, k_C) \le{}& \sum_{E \in \mathcal{E}_1} \left(2\phi_E + 5\nu_E \left(\frac{8}{27}\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\right) + 6k_E \left(\frac{8}{27}\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\right)\right) \\
&+ \sum_{E \in \mathcal{E}_2} \left(2\phi_E + 5\nu_E \left(\frac{8}{27}\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\right) + 6k_E \left(\frac{8}{27}\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\right)\right) \\
\le{}& 2\left(\phi_{C_1} + \phi_{C_2}\right) + 5\left(\nu_{C_1} + \nu_{C_2}\right)\left(\frac{8}{27}\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\right) \\
&\quad + 6\left(k_{C_1} + k_{C_2}\right)\left(\frac{8}{27}\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\right) \\
\le{}& 2\phi_C + 5 \cdot \frac{8}{27}\left(2\nu_C + k_C\right)\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3 + 6 \cdot \frac{8}{27} k_C \left(\frac{\phi_C + 2ak_C}{\mu}\right)^3 \\
\le{}& 2\phi_C + 5\nu_C \left(\frac{\phi_C + 2ak_C}{\mu}\right)^3 + 6k_C \left(\frac{\phi_C + 2ak_C}{\mu}\right)^3,
\end{aligned}$$

which implies (A.1).

Case (ii). We find two parallel planes h_1 and h_2 that divide C into three boxes C_1, C_2, and C_3 (in this order) so that all rectangles in F_{C_2} belong to one class. Lemma 2.2 implies that the rectangles in F_C are partitioned among C_1, C_2, and C_3. Moreover, h_1 and h_2 partition the vertices in the interior of C. Thus,

$$\phi_{C_1} + \phi_{C_2} + \phi_{C_3} \le \phi_C \qquad \text{and} \qquad k_{C_1} + k_{C_2} + k_{C_3} \le k_C.$$

The planes h_1 and h_2 can intersect the rectangles in $S_C \setminus F_C$. Each long rectangle in $S_C \setminus F_C$ is partitioned into at most three long rectangles. Each short rectangle in S_C is partitioned into at most three rectangles; if two of these rectangles are long, then one of the long rectangles must be in S_{C_2}, since C_2 is sandwiched between C_1 and C_3 (see the proof of Lemma 2.2). In other words,

$$\nu_{C_1} + \nu_{C_3} \le 2\nu_C + k_C \qquad \text{and} \qquad \nu_{C_2} \le \nu_C + k_C. \tag{A.7}$$

Lemma 2.2 also implies that

(A.8)

$$\phi_{C_i} + 2ak_{C_i} \le \frac{2(\phi_C + 2ak_C)}{3}, \qquad \text{for } i = 1, 3, \text{ and } \phi_{C_2} + 2ak_{C_2} \le \phi_C + 2ak_C.$$

Let $\mathcal{E}_i, 1 \le i \le 3$ be the set of boxes obtained by applying all the free cuts in S_{C_i} in step 4 of the dividing stage. Note that for each box $E \in \mathcal{E}_2$, F_E contains only one class of rectangles. It is clear that

$$\Psi(\phi_C, \nu_C, k_C) \le \sum_{E \in \mathcal{E}_1} \Psi(\phi_E, \nu_E, k_E) + \sum_{E \in \mathcal{E}_2} \Phi(2\phi_E, \nu_E, k_E) + \sum_{E \in \mathcal{E}_3} \Psi(\phi_E, \nu_E, k_E),$$

where $\Phi(\,)$ is as defined in the proof of Lemma 5.2. We again consider two cases.

(a) $\mu < \phi_C + 2ak_C \le 3\mu/2$. By (A.8), each box in $\mathcal{E}_1$ and $\mathcal{E}_3$ is a leaf of $\mathcal{T}_B$. Therefore, by (A.4),

$$\begin{aligned}
\Psi(\phi_C, \nu_C, k_C) &\le \sum_{E \in \mathcal{E}_1} f_E + \sum_{E \in \mathcal{E}_2} \Phi\left(2\phi_E, \nu_E, k_E\right) + \sum_{E \in \mathcal{E}_3} f_E \\
&\le \sum_{E \in \mathcal{E}_1} (\phi_E + \nu_E) + \sum_{E \in \mathcal{E}_3} (\phi_E + \nu_E) + \sum_{E \in \mathcal{E}_2} \Phi\left(2\phi_E, \nu_E, k_E\right) \\
&\le (\phi_{C_1} + \phi_{C_3}) + (\nu_{C_1} + \nu_{C_3}) + \sum_{E \in \mathcal{E}_2} \Phi\left(2\phi_E, \nu_E, k_E\right).
\end{aligned}$$

For each box $E \in \mathcal{E}_2$, $\phi_E + 2ak_E \le 3\mu/2$. Hence, by (5.7), we have

$$\Phi\left(2\phi_E, \nu_E, k_E\right) \le 2\phi_E + 2\nu_E + k_E.$$

As a result,

$$\begin{aligned}
\Psi(\phi_C, \nu_C, k_C) &\le (\phi_{C_1} + \phi_{C_3}) + (\nu_{C_1} + \nu_{C_3}) + \sum_{E \in \mathcal{E}_2} (2\phi_E + 2\nu_E + k_E) \\
&\le (\phi_{C_1} + \phi_{C_3}) + (\nu_{C_1} + \nu_{C_3}) + (2\phi_{C_2} + 2\nu_{C_2} + k_{C_2}) \\
&\le 2\phi_C + 4\nu_C + 4k_C,
\end{aligned}$$

where the last inequality follows from (A.7). This inequality implies (A.1).

(b) $\phi_C + 2ak_C > 3\mu/2$. For a box $E \in \mathcal{E}_1 \cup \mathcal{E}_3$,

$$\max\left\{\left(\frac{\phi_E + 2ak_E}{\mu}\right)^3, 1\right\} \le \max\left\{\left(\frac{2}{3}\left(\frac{\phi_C + 2ak_C}{\mu}\right)\right)^3, 1\right\} = \frac{8}{27}\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3.$$

Similarly, for a box $E \in \mathcal{E}_2$,

$$\max\left\{2\left(\frac{\phi_E + ak_E}{\mu}\right), 1\right\} \le 2\left(\frac{\phi_C + 2ak_C}{\mu}\right).$$

By the induction hypothesis and (5.1),

$$\begin{aligned}\Psi(\phi_C, \nu_C, k_C) \le &\sum_{E\in\mathcal{E}_1}\left(2\phi_E + 5\cdot\frac{8}{27}\nu_E\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3 + 6\cdot\frac{8}{27}k_E\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\right)\\ &+ \sum_{E\in\mathcal{E}_2}\left(2\phi_E + 4\nu_E\left(\frac{\phi_C + 2ak_C}{\mu}\right) + 4k_E\left(\frac{\phi_C + 2ak_C}{\mu}\right)\right)\\ &+ \sum_{E\in\mathcal{E}_3}\left(2\phi_E + 5\cdot\frac{8}{27}\nu_E\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3 + 6\cdot\frac{8}{27}k_E\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\right).\end{aligned}$$

Since $\phi_C + 2ak_C > 3\mu/2$,

$$\frac{\phi_C + 2ak_C}{\mu} \le \frac{4}{9}\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3.$$

Therefore, using (A.7), we have

$$\begin{aligned}\Psi(\phi_C, \nu_C, k_C) &\le 2\left(\phi_{C_1} + \phi_{C_2} + \phi_{C_3}\right) + 5\cdot\frac{8}{27}\left(\nu_{C_1} + \nu_{C_3}\right)\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\\ &\qquad + \frac{16}{9}\nu_{C_2}\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3 + 6\cdot\frac{8}{27}\left(k_{C_1} + k_{C_3}\right)\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\\ &\qquad + \frac{16}{9}k_{C_2}\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3\\ &\le 2\phi_C + 5\nu_C\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3 + 6k_C\left(\frac{\phi_C + 2ak_C}{\mu}\right)^3,\end{aligned}$$

which implies (A.1). $\square$

Acknowledgments. We would like to thank Seth Teller for providing us with the Soda Hall dataset (shown in Figure 1.1) created at the Department of Computer Science, University of California at Berkeley.

REFERENCES

[1] P. K. Agarwal, J. Erickson, and L. J. Guibas, *Kinetic binary space partitions for intersecting segments and disjoint triangles*, in Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, pp. 107–116.

[2] P. K. Agarwal, L. J. Guibas, T. M. Murali, and J. S. Vitter, *Cylindrical static and kinetic binary space partitions*, in Proceedings of the 13th ACM Symposium on Comput. Geom., ACM, New York, 1997, pp. 39–48.

[3] P. K. Agarwal and S. Suri, *Surface approximation and geometric partitions*, in Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms, Arlington, VA, 1994, pp. 24–33.

[4] J. M. Airey, *Increasing Update Rates in the Building Walkthrough System with Automatic Model-space Subdivision and Potentially Visible Set Calculations*, Ph.D. thesis, Dept. of Computer Science, University of North Carolina, Chapel Hill, NC, 1990.

[5] C. Ballieux, *Motion Planning Using Binary Space Partitions*, Technical Report inf/src/93-25, Utrecht University, Utrecht, The Netherlands, 1993.

[6] A. T. Campbell, *Modeling Global Diffuse Illumination for Image Synthesis*, Ph.D. thesis, Department of Computer Sciences, University of Texas, Austin, TX 1991.

[7] T. Cassen, K. R. Subramanian, and Z. Michalewicz, *Near-optimal construction of partitioning trees by evolutionary techniques*, in Proceedings of Graphics Interface '95, Quebec, Canada, Canadian Human-Computer Communications Society, 1995, pp. 263–271.

[8] E. Catmull, *A subdivision algorithm for computer display of curved surfaces*, Technical Report UTEC-CSC-74-133, Department Computer Sciences, University of Utah, Salt Lake City, UT, 1974.

[9] B. Chazelle, *Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm*, SIAM J. Comput., 13 (1984), pp. 488–507.

[10] N. Chin and S. Feiner, *Near real-time shadow generation using BSP trees*, in Proceedings of the ACM International Conference on Computer Graphics and Interactive Techniques, Boston, MA, 1989, pp. 99–106.

[11] N. Chin and S. Feiner, *Fast object-precision shadow generation for area light sources using BSP trees*, in Proceedings of the ACM Symposium on Interactive 3D Graphics, Cambridge, MA, 1992, pp. 21–30.

[12] Y. Chrysanthou, *Shadow Computation for 3D Interaction and Animation*, Ph.D. thesis, Queen Mary and Westfield College, University of London, London, UK, 1996.

[13] M. de Berg, *Linear size binary space partitions for fat objects*, in Proceedings of the 3rd Annual European Symposium on Algorithms, Lecture Notes in Comput. Sci. 979, Springer-Verlag, Berlin, 1995, pp. 252–263.

[14] S. E. Dorward, *A survey of object-space hidden surface removal*, Internat. J. Comput. Geom. Appl., 4 (1994), pp. 325–362.

[15] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, 2nd ed., Addison-Wesley, Reading, MA, 1990.

[16] H. Fuchs, Z. M. Kedem, and B. Naylor, *On visible surface generation by a priori tree structures*, in Proceedings of the ACM International Conference on Computer Graphics and Interactive Techniques, ACM, New York, 1980, pp. 124–133.

[17] C. Mata and J. S. B. Mitchell, *Approximation algorithms for geometric tour and network design problems*, in Proceedings of the 11th Annual ACM Symposium on Comput. Geom., ACM, New York, 1995, pp. 360–369.

[18] T. M. Murali, P. K. Agarwal, and J. S. Vitter, *Constructing binary space partitions for orthogonal rectangles in practice*, in Proceedings of the 6th Annual European Symposium on Algorithms, Lecture Notes in Comput. Sci. 1461, Springer, Berlin, 1998, pp. 211–222.

[19] T. M. Murali and T. A. Funkhouser, *Consistent solid and boundary representations from arbitrary polygonal data*, in Proceedings of the ACM Symposium on Interactive 3D Graphics, Providence, RI, 1997, pp. 155–162.

[20] B. Naylor, *Constructing good partition trees*, in Proceedings of Graphics Interface, Toronto, Ontario, Canada, Canadian Human-Computer Communications Society, 1993, pp. 181–191.

[21] B. Naylor and W. Thibault, *Application of BSP Trees to Ray-Tracing and CSG Evaluation*, Technical Report GIT-ICS 86/03, School of Information and Computer Science, Georgia Institute of Technology, 1986.

[22] B. F. Naylor, *SCULPT: An interactive solid modeling tool*, in Proceedings of Graphics Interface, Halifax, Nova Scotia, Canada, Canadian Human-Computer Communications Society, 1990, pp. 138–148.

[23] B. F. Naylor, *Interactive solid geometry via partitioning trees*, in Proceedings of Graphics Interface, Vancouver, B.C., Canada, Canadian Human-Computer Communications Society, 1992, pp. 11–18.

[24] B. F. Naylor, J. Amanatides, and W. C. Thibault, *Merging BSP trees yields polyhedral set operations*, in Proceedings of the ACM International Conference on Computer Graphics and Interactive Techniques, Dallas, TX, 1990, pp. 115–124.

[25] M. S. Paterson and F. F. Yao, *Efficient binary space partitions for hidden-surface removal and solid modeling*, Discrete Comput. Geom., 5 (1990), pp. 485–503.

[26] M. S. Paterson and F. F. Yao, *Optimal binary space partitions for orthogonal objects*, J.

Algorithms, 13 (1992), pp. 99–113.

[27] R. A. Schumacker, R. Brand, M. Gilliland, and W. Sharp, *Study for Applying Computer-Generated Images to Visual Simulation*, Technical Report AFHRL–TR–69–14, U.S. Air Force Human Resources Laboratory, Brooks Air Force Base, San Antonio, TX, 1969.

[28] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, *A characterization of ten hidden-surface algorithms*, ACM Comput. Surv., 6 (1974), pp. 1–55.

[29] S. J. Teller, *Visibility Computations in Densely Occluded Polyhedral Environments*, Ph.D. thesis, Department of Computer Science, University of California at Berkeley, Berkeley, CA 1992.

[30] W. C. Thibault and B. F. Naylor, *Set operations on polyhedra using binary space partitioning trees*, in Proceedings of the ACM International Conference on Computer Graphics and Interactive Techniques, Anaheim, CA, 1987, pp. 153–162.

[31] E. Torres, *Optimization of the binary space partition algorithm (BSP) for the visualization of dynamic scenes*, in Proceedings of Eurographics, European Association for Computer Graphics, Aire-la-Ville, Switzerland, 1990, pp. 507–518.