Combinatorial and Computational Geometry
MSRI Publications
Volume **52**, 2005

Binary Space Partitions: Recent Developments

CSABA D. TÓTH

Abstract. A binary space partition tree is a data structure for the representation of a set of objects in space. It found an increasing number of applications over the last decades. In recent years, intensifying research focused on its combinatorial properties, which affect directly the efficiency of applications. Important advances were made on binary space partitions for disjoint line segments in the plane and for axis-aligned objects in higher dimensions. New research directions were also initiated on some realistic polygonal scenes and on kinetic binary space partitions. This paper attempts to give an overview of these results and reiterates some of the most pressing open problems.

1. Introduction

The *binary space partition tree* is a geometric data structure obtained by a recursive partitioning scheme, called *binary space partition* (for short, *BSP*) over a set of input objects: The space is partitioned along a hyperplane into two half-spaces, then either half-space is partitioned recursively until every subproblem contains only a trivial fraction of the input objects. The concept of BSP has emerged from the computer graphics community in the seventies. It was originally designed to assist efficient hidden-surface removal algorithms for moving viewpoints, but it has later found widespread applications in many areas of computational and combinatorial geometry.

In many of the applications, the bottle neck of the space complexity is the size of the BSP tree they rely on. Combinatorial research focused on determining the worst case complexity of BSPs for certain classes of inputs. Despite the simplicity of the BSP algorithm, it is often challenging to determine the so-called *partition complexity* even for simple object classes such as disjoint line segments in the plane, or axis-aligned boxes in higher dimensions. Ideally, the partition hyperplanes do not split the input objects, and the size of the BSP tree is linear in terms of the input size. In many cases, however, it is inevitable that

input objects are also fragmented, and the size of the data structure becomes superlinear.

This paper surveys results on the combinatorial properties of BSPs from the last four-five years. Inevitably, we include less recent results as well because early ideas and observations are often used, sometimes in an enhanced form, to obtain new results. Before we move on to the latest developments, let us define the binary space partition and the partition complexity, and recall a few applications and early results.

Definitions. A binary space partition tree is a recursive partition scheme for an input set of pairwise interior disjoint objects in $\mathbb{R}^d$, $d \in \mathbb{N}$. If the input contains two full-dimensional objects or a lower-dimensional object, we partition the space by a hyperplane h and recursively apply two binary space partitions for the objects clipped in each of the two open half-spaces of h. If the input is at most one full-dimensional object (and no lower-dimensional object), we stop.

The partition algorithm naturally corresponds to a binary tree: Every node corresponds to the input of a recursive call of the BSP: The root corresponds to the initial input set, the two children of a non-leaf node correspond to the inputs of its two subproblems. The BSP tree *data structure* is based on this binary tree: Every leaf stores at most one full-dimensional object which is the input of the corresponding subproblem; and every non-leaf node stores the splitting hyperplane and the (lower-dimensional) objects of the corresponding subproblem that lie on the splitting hyperplane. As a convention, the non-leaf nodes store only k-dimensional fragments of k-dimensional objects lying on the splitting hyperplane in $\mathbb{R}^d$, $0 \leq k \leq d$. For example, if a splitting hyperplane h crosses an input segment s then the point $h \cap s$ is never stored. Figure 1 depicts an example for a recursive partitioning and corresponding BSP tree for four input objects.

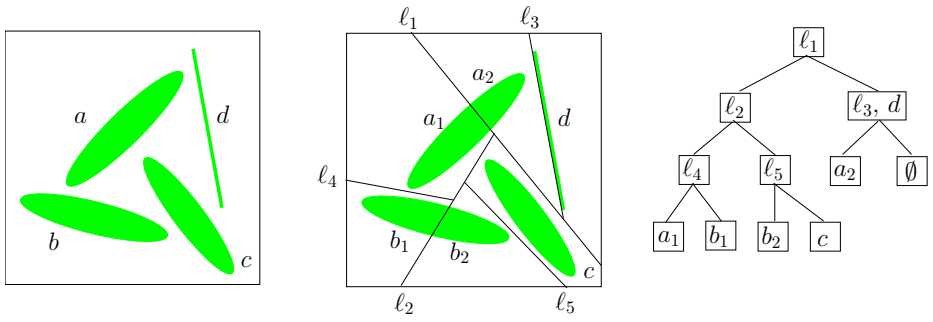

Figure 1. Three disjoint ellipses and a line segment (left), binary space partition with five partition lines (middle), and the corresponding BSP tree (right).

The binary space partition algorithm also defines a convex space partition: Every node of the BSP tree corresponds to a convex polytope, which is the

intersection of the halfspaces bounded by the splitting hyperplanes of its ancestors. The root corresponds to the space $\mathbb{R}^d$. A splitting hyperplane stored at a non-leaf node splits the corresponding convex polytope into the two polytopes corresponding to the two child nodes. The polytope corresponding to a node is also called the *cell* of the (sub)problem because it contains every input object of the (sub)problem. Observe that the set of all leaves of a BSP tree correspond to a convex partition of the space.

An important special BSP, called *autopartition*, is defined for at most $(d-1)$-dimensional input objects in $\mathbb{R}^d$. An autopartition is a BSP where every splitting hyperplane contains an input object of the corresponding (sub)problem.

A BSP has two important parameters: Its *size* is the total number of fragments of objects stored at the nodes of the BSP tree; its *height* is the height of the BSP tree. The BSP in Figure 1 has size 6 and height 3. An easy way to compute the size of the BSP is to sum the number of input objects and the number of cuts, the events when a splitting hyperplane partitions (a fragment of) an input object. If the BSP does not make "useless cuts", that is, if every splitting plane partitions the convex hull of the input objects, then the BSP size is an upper bound on the number of non-leaf nodes of the BSP tree.

There are many BSPs for every input set depending on the choice of the splitting hyperplane at each subproblem. The *partition complexity* of a set S of objects is the size of the smallest BSP for S. The minimal size BSP does not necessarily comes with minimal heights, though. The *autopartition complexity* of a set S is the size of the smallest autopartition for S. Clearly, the autopartition complexity is never smaller than the partition complexity.

The convention that objects lying in a partition hyperplane are not partitioned further leads to somewhat counterintuitive phenomenon: A set of objects in $\mathbb{R}^d$ with superlinear partition complexity will have linear partition complexity when embedded into $\mathbb{R}^{d+1}$ because a single splitting hyperplane contains them all. One may define another binary partitioning scheme which would partition recursively the fragments of objects lying in each splitting hyperplane. The minimal number of fragments under such an alternative partition scheme may be much higher than the partition complexity. In this survey, we focus on combinatorial properties of the partition complexity.

1.1. Applications. The initial and most prominent applications, where the BSP tree itself is stemming from, lie in computer graphics: BSPs support fast *hidden-surface removal* [Schumacker et al. 1969; Fuchs et al. 1980; Murali 1998] and *ray tracing* [Naylor and Thibault 1986] for moving viewpoints. Rendering is used for visualizing spatial opaque surfaces on the screen. A common and efficient rendering technique is the so-called *painter's method*. It draws every object sequentially according to the *depth order* or *back-to-front* order, starting with the deepest object and proceeding with the objects closer to the viewer.

When all the objects have been drawn, every pixel represents the color of the object closest to the viewpoint.

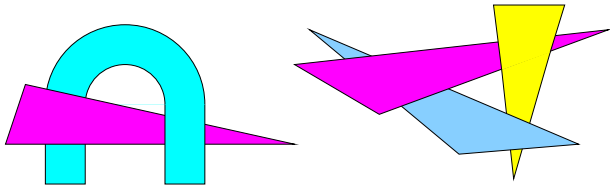

Figure 2. Objects may have cyclic overlaps.

Unfortunately, some sets of objects have cyclic overlaps, where no valid depth order exists (Figure 2). A BSP partitions the input objects into fragments for which the depth order always exists for every viewpoint. It is easy to extract the depth order of the fragments by a simple traversal of the BSP tree. An additional benefit of BSPs is that the depth order can easily be updated as the viewpoint moves continuously: It is enough to swap the two children of a node of the BSP tree when the viewpoint crosses the corresponding splitting hyperplane. (Computing the depth order, if it exists, for a *fixed* viewpoint may be easier than constructing a BSP, however; see [de Berg 1993].)

The first applications of BSPs were soon followed by many others, e.g., *constructive solid geometry* [Thibault and Naylor 1987; Naylor et al. 1990; Buchele 1999] and *shadow generation* [Chin and Feiner 1989; Batagelo and Jr. 1999]. The BSP for the faces of a full dimensional polyhedral solid can represent the solid and its boundary with Boolean set operations. This, in turn, can be used for efficient real-time shadow generation by computing the union of shadow volumes obtained from traversals of the BSP tree in depth order for each light source position. BSPs were also used in *surface simplification* [Agarwal and Suri 1998; Shaffer and Garland 2001; Pauly et al. 2002] for clustering sample points.

Other applications of BSP trees include *range counting* [Agarwal and Matoušek 1992], *point location* [Arya et al. 2000; de Berg 2000], *collision detection* [Ar et al. 2000; Ar et al. 2002], *robotics* [Ballieux 1993], *graph drawing* [Asano et al. 2003], and *network design* [Mata and Mitchell 1995]. BSP trees generalize some of the most commonly used geometric search structures such as quadtrees/octrees, Kd-trees, and BAR-trees [Duncan et al. 2001], each of which has numerous applications on its own right.

Early research. Research on combinatorial properties of BSPs began with two influential papers of Paterson and Yao [1990; 1992]. They considered BSPs for disjoint $(d-1)$-dimensional hyper-rectangles in $\mathbb{R}^d$ (including line segments in the plane). Many of their initial observations (about free cuts, anchored segments, and round-robin algorithms) are key elements of the most recent partition algorithms, as well.

Paterson and Yao [1990] gave a randomized and a deterministic algorithm to construct a BSP of size $O(n \log n)$ and height $O(\log n)$ for n disjoint line segments in the plane. The height bound is optimal apart from a constant factor, but the size bound is not known to be tight. A lower bound construction, where the partition complexity of n disjoint line segments is $\Omega(n \log n / \log \log n)$, is discussed in Section 3.

They also proved that the partition complexity of n disjoint $(d-1)$-dimensional simplices in $\mathbb{R}^d$, $d \geq 3$, is $O(n^{d-1})$, which is best possible for $d = 3$ and tight for autopartitions in every dimension [Paterson and Yao 1990]. In three-space, there are n disjoint line segments, lying along a hyperbolic paraboloid, whose partition complexity is $\Theta(n^2)$. The same lower bound carries over to hyper-rectangles in $\mathbb{R}^d$ with the same $\Theta(n^2)$ partition complexity. The gap between the upper and lower bounds in higher dimensions has resisted research efforts so far.

OPEN PROBLEM. *What is the partition complexity of n disjoint $(d-1)$-dimensional simplices in $\mathbb{R}^d$, for $d \geq 4$?*

Every BSP computes a convex partition of the free space around the input objects. The size of the convex partition is the number of leaves of the BSP tree. The partition complexity of $(d-1)$-dimensional simplices is, therefore, cannot be smaller than the minimum convex partition of a polyhedron with n faces in $\mathbb{R}^d$. It would be tempting to derive lower bounds on the BSP size from the convex partitioning, unfortunately, no super-quadratic lower bound is known for this problem [Chazelle 1984] for any $d \in \mathbb{N}$.

OPEN PROBLEM. *What is the size of a minimum convex partition of a polyhedron with n faces in $\mathbb{R}^d$, $d > 3$?*

In this survey we focus on the worst case (maximum) partition complexity of a set of objects of a certain type. Computation or approximation of the *optimal* size BSP for *specific* instances seems to be an elusive open problem (see e.g., [Agarwal et al. 2000c]).

OPEN PROBLEM. *Is it NP-hard or is there a polynomial algorithm to compute the partition complexity of n given disjoint line segments in the plane?*

OPEN PROBLEM. *Is it APX-hard or is there a polynomial time approximation scheme to compute the partition complexity of n given disjoint line segments in the plane?*

Road map. In Section 2, we continue with recalling earlier ideas and observations which proved to be omnipresent in later results on BSPs. This section can be considered a short warm up course on the basics on binary space partitioning, which is essential in understanding the sometimes intricate partition schemes and lower bound constructions. Section 3 sketches the proof for two closely related results on disjoint line segments in the plane: An $\Omega(n \log n / \log \log n)$ lower bound construction and a constructive upper bound on segments with k

distinct orientations, $k \in \mathbb{N}$. Section 4 gathers a series of results about BSPs for axis-aligned input objects, where all splitting hyperplanes are also axis-aligned. We conclude the paper with two short sections related to other fields of computational geometry: Section 5 reviews results on realistic object classes which sometimes allow for linear partition complexity. Section 6 deals with kinetic BSPs for continuously moving input objects.

2. Preliminaries

Many of the resent results on BSPs are based on or are extending basic concepts known for decades. In this section, we present the most important ideas with references the their recent extensions or enhancements.

Free cuts. A hyperplane h that does not split any input object but partitions the *set* of objects into smaller sets (i.e., at least two of $\text{int}(h^-)$, h, and $\text{int}(h^+)$ intersect input objects), is called a *free cut*. A minimum size BSP should use free cuts whenever they are available [Paterson and Yao 1990]. A BSP that uses free cuts only is called *perfect*. De Berg, de Groot, and Overmars [de Berg et al. 1997a] designed an $O(n^2 \log n)$ time algorithm to detect if a perfect BSP exists for n disjoint lien segments in the plane.

Cutting along a $(d-1)$-dimensional object whose (relative) boundary lies on the boundary of the cell is clearly a free cut (Figure 3). In the plane, this implies, for instance, that if segment s is cut already at points p and q, then the subproblem containing the middle fragment $pq \subset s$ can be partitioned along pq without cutting any input segment. Therefore, if we track the cuts of a single input segment through an optimal binary space partitioning, then only the two *endings* (the portion between the segment endpoint and the first or last cut) can be further fragmented. This observation leads to an $O(n \log n)$ upper bound on the partition complexity of n disjoint line segments [Paterson and Yao 1990], but it is a basic element in all BSP algorithms.

Free cuts along objects. Consider a $(d-1)$-dimensional object s in $\mathbb{R}^d$. If a fragment $\hat{s}$ of s is bounded by cuts (previous splitting planes), then a hyperplane along $\hat{s}$ is a free cut for the subproblem containing $\hat{s}$. As a consequence, we can assume that whenever a splitting hyperplane cuts a fragment $\hat{s}$ of s then $\hat{s}$ is adjacent to the (relative) boundary of s. If a splitting hyperplane h partitions $\hat{s}$ but does not partition the common (relative) boundary of s and $\hat{s} \subset s$ (e.g. in Figure 3), then this commmon boundary is entirely on one side of h and the fragment of $\hat{s}$ on the other side of h is bounded by cuts and so it is a free cut. For each final fragment of s adjacent to the boundary of s, there is one such cut on every level of the BSP tree. Therefore the partition complexity of n disjoint $(d-1)$-dimensional simplices in $\mathbb{R}^d$ is upper bounded by the product of the size and the height of a BSP for their $(d-2)$-dimensional boundary simplices. For

example, we obtain an $O(n \log n)$ size BSP for n disjoint line segments in the plane from a $2n$ size and $\lceil \log 2n \rceil$ height Kd-tree of the $2n$ segment endpoints.

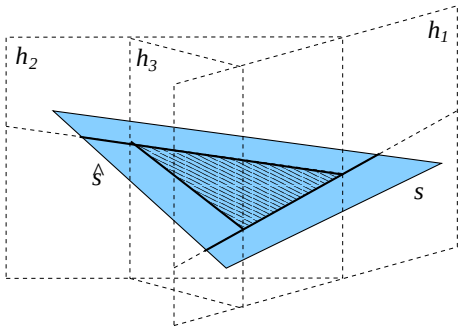

Figure 3. An input triangle s in $\mathbb{R}^3$ is cut by three vertical splitting planes h_1, h_2, and h_3. The striped triangle fragment is a free cut for the cell bounded by the three splitting planes.

Paterson and Yao [1990] showed that cutting along all input objects in a random order and applying free cuts whenever possible results in an $O(n^{d-1})$ expected size BSP for n disjoint $(d-1)$-dimensional simplices in $\mathbb{R}^d$, $d \geq 3$. This BSP, however, can have linear expected height in the worst case, e.g., if the input is the face set of a convex polytope. In three-space, this gives a randomized BSP of $O(n^2)$ size and $O(n)$ height for n disjoint triangles.

We can construct deterministically a BSP of $O(\log^2 n)$ height and $O(n^2 \log^2 n)$ size: Project all triangle edges to the xy-plane, and cover them with $O(n^2)$ pairwise non-crossing segments. By [Paterson and Yao 1990], there is a $O(n^2 \log(n^2))$ size and $O(\log(n^2))$ height BSP for these segments. The lifting of the planar splitting lines to vertical splitting planes in $\mathbb{R}^3$ along with all possible free cuts gives a BSP for the triangles. Refining this idea, Agarwal et al. [2000c] designed a BSP of size $O(n^2 \log^2 n)$ size and $O(\log n)$ height for n disjoint triangles; while Agarwal, Erickson, and Guibas [Agarwal et al. 1998] reported a randomized BSP of expected $O(n^2)$ size and $O(\log n)$ height.

Cycles. Assume that we are given disjoint $(d-1)$-dimensional objects in R^d. If the hyperplane spanned by any object is disjoint from all other objects, then every hyperplane through an object is a free cut and we obtain a perfect BSP of size n by partitioning along them in an arbitrary order. We can define a binary relation between two objects a and b saying that $a \succ b$ if and only if the hyperplane through a splits b. A BSP of size n using free cuts only still exists if $\succ$ is an acyclic relation: No object is cut if we always split the space along a minimum element with respect to $\succ$. Sets with large partition complexity ought to have many *cycles* w.r.t. $\succ$. Figure 4 depicts examples of cycles in the plane and in three space. Cycles are vital in the analysis of BSPs for disjoint line segments in the plane.

Anchored segments. A line segment is *anchored* if one of its endpoints lies on the boundary of the cell. An anchored segment has only a uni-directional extension that might split other segments. Paterson and Yao [1992] found a small *BSP for axis-aligned anchored segments* in the plane that cuts non-anchored segments at most four times. Using this BSP as a subroutine inductively, they obtain an $O(n)$ size BSP for n disjoint axis-parallel line segments: If a segment is partitioned into c pieces during a BSP for anchored segments, then $c-2$ pieces are free-cuts in their proper subproblems and are not fragmented any further, the remaining two pieces are anchored and are taken care of by the next call of the subroutine for anchored segments. This idea was extended and turned out to be extremely fruitful in a number of recent results [Tóth 2003b; Tóth 2003a]

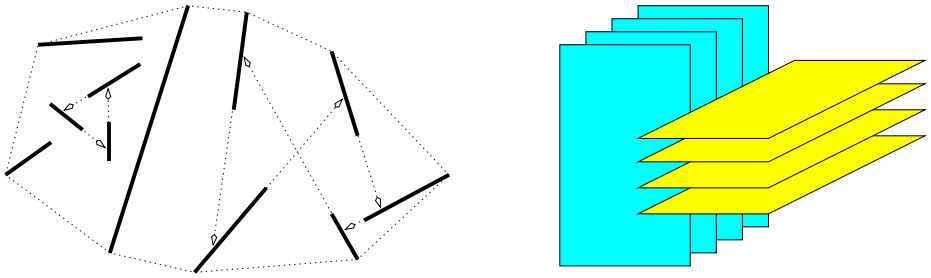

Figure 4. A set of line segments with a free cut and two cycles, one of which is an anchored cycle (left). Axis-aligned rectangles where $\succ$ defines a complete bipartite graph (right).

De Berg et al. [1997b] noticed a useful property of *cycles of anchored segments*. Consider a cycle $(a_1, a_2, \ldots, a_k)$ of anchored segments, where $a_i \succ a_{i+1}$ for $i = 1, 2, \ldots, k-1$ and $a_k \succ a_1$. Assume, furthermore, that the first anchored segment hit by the extension of a_i is a_{i+1} for $i = 1, 2, \ldots k-1$, and a_1 for $i = k$ (Figure 4). A *cycle cut* for $(a_1, a_2, \ldots, a_k)$ is a partition subroutine where we cut sequentially along the segments $a_k, a_{k-1}, \ldots, a_1$ in this order. Note that only the first splitting line can cut anchored segments. Indeed, $a_{k-1} \succ a_k$, so the extension of a_{k-1} hits a_k. The split along a_{k-1} does not extend beyond a_k, along which we have made a previous cut. Similarly, partitions along $a_{k-2}, \ldots, a_2, a_1$ are all free cuts w.r.t. anchored segments of the initial problem. This property of cycles of anchored segments in the plane unfortunately does not generalize to higher dimensions.

One possible higher dimensional generalization of anchored segments was introduced by Dumitrescu et al. [2004] in the context of BSPs for axis-aligned k-flats in $\mathbb{R}^d$. An axis-aligned k-flat b is called an *ℓ-stabber*, $0 \leq \ell \leq k$, if two opposite $(k-\ell)$-faces of b lie on the boundary of the convex hull. Tóth [2003a] defined another generalization: A 1-stabber b is a *shelf* with respect to a cell C if $C \setminus b$ is simply connected. He showed that the partition complexity of shelves is $O(n \log n)$ in $\mathbb{R}^3$.

Round-robin partition. Researchers tried to construct small size BSPs by computing, for each subproblem, a splitting plane that optimizes some objective function [Paterson and Yao 1992; Nguyen 1996; Nguyen and Widmayer 1995]. Intuitively, an "optimal" cut should cut few objects and partition the input into almost equal size subproblems. Local optimization rarely lead to near optimal BSPs, one notable exception is the case of axis-parallel segments in $\mathbb{R}^d$, $d \geq 3$. Paterson and Yao [1992] considered the product of the number of segments for each axis-parallel direction as an objective function. They showed that locally optimal axis-aligned cuts (i.e., cuts decreasing maximally this value) build up to an $O(n^{d/(d-1)})$ size and $O(\log n)$ height BSP, which is best possible. The orientation of optimal axis-aligned cuts in recursive calls varies in an irregular order.

A simple but efficient *round-robin BSP* scheme for axis-aligned objects in $\mathbb{R}^d$ works in rounds: Each round partitions the space into 2^d pieces in d recursive steps along axis-aligned hyperplanes of all d orientations. The round-robin BSP scheme for points in $\mathbb{R}^d$ is the well known Kd-tree. For axis-aligned line segments in $\mathbb{R}^d$, a round-robin BSP makes a cut recursively along the median hyperplanes of the segment endpoints. This BSP has $O(n^{(d-1)/d})$ size and $O(\log n)$ height, which is best possible for n axis-aligned segments [Paterson and Yao 1992]. Recently, round-robin schemes [Dumitrescu et al. 2004; Hershberger et al. 2004] lead to non-trivial bounds on the axis-aligned partition complexity of (disjoint) k-flats in $\mathbb{R}^d$ for certain values of k.

3. Line Segments in the Plane

The best known upper bound on the partition complexity of n disjoint line segments in the plane is $O(n \log n)$. Paterson and Yao [1990] found an astonishingly simple randomized BSP of expected $O(n \log n)$ size. The trapezoid decomposition method of Preparata [1981] (see also [Preparata and Shamos 1985]) gives a deterministic BSP of this size as well. It was widely believed that disjoint segments in the plane have linear partition complexity, this was supported by experiments and by linear upper bounds for certain input classes such as axis-parallel segments [Paterson and Yao 1992], anchored segments, and segments of similar length [de Berg et al. 1997b]. Tóth [2003c] gave a family of n disjoint line segments for every n, $n \in \mathbb{N}$, whose partition complexity is $\Theta(n \log n / \log \log n)$. An extension of these ideas lead to an $O(n \log k)$ bound on the partition complexity of n disjoint line segments with k distinct orientations.

Lower bound construction. We show how the idea of cycles over segments in the plane can be developed into a lower bound construction whose partition complexity is $\Theta(n \log n / \log \log n)$. For the sake of simplicity, we focus on autopartitions, where every cut is made along an input segment. We start with the simple observation that a BSP cuts at least one segment of every cycle. If every

segment appears in a unique cycle of size 3, then this already implies that the partition complexity of n segments is at least $n + n/3$. To guarantee a higher rate of fragmentation we design a recursive construction on k levels: The construction S_i on level i, $1 < i \leq k$, consists of copies of the construction S_{i-1} of level $i-1$. On the lowest level, S_0 consists of a single line segment.

We squeeze S_{i-1} into a long and skinny rectangle, and build congruent copies of this deformed S_{i-1} into S_i. Note that the partition complexity is invariant under affine transformations. The copies of S_{i-1} are so skinny that their position in S_i can be described by disjoint line segments, which we call *sticks*. The lower bound on the partition complexity depends on three elements: *(a)* If S_i consists of x copies of S_{i-1} arranged in cycles then we can guarantee that any BSP makes at least $\Omega(x)$ cuts on sticks. *(b)* If a cut of a stick (that is, a cut through a copy of S_{i-1}) implies that $\Omega(|S_{i-1}|)$ true segments of S_{i-1} are cut, then the arrangement of sticks at level i actually guarantees $\Omega(n)$ cuts on the input segments. *(c)* Once we make sure that cycles of sticks at distinct levels induce distinct cuts on each input segment, then the entire construction gives a lower bound of $\Omega(kn)$ on the partition complexity, where k is the number of levels. It remains to show how to find a construction satisfying all three conditions with $k = \Omega(\log n / \log\log n)$ levels.

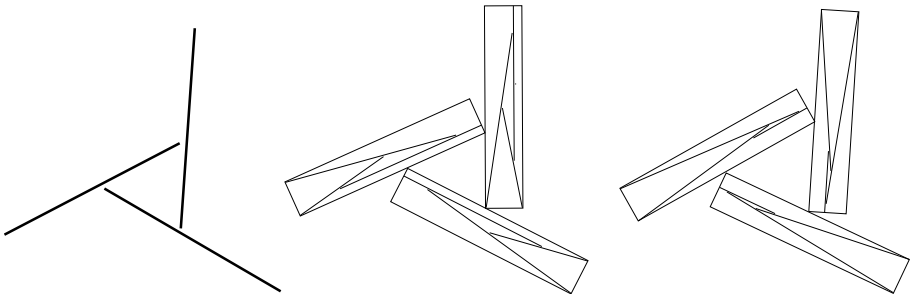

Figure 5. A cycle (left) and two cycles composed of cycles enclosed in long and skinny boxes (middle and right). A line spanned by segments in one rectangle either pierces a cycle in another rectangle (middle) or not (right).

A splitting line ℓ intersecting a copy of S_{i-1} does not necessarily cut *all* the true segments in S_{i-1}. If ℓ cuts all sub-constructions S_{i-2} within S_{i-1}, then it destroys the cycle arrangement of the sticks in S_{i-1} (Figure 5, middle) and so this cycle arrangement does not induce any cuts on segments of S_{i-1}. Therefore we assume that the sticks of S_{i-1} are arranged so that no line ℓ spanned by an input segment in another copy of S_{i-1} destroys their cycle (Figure 5, right). If, however, a line ℓ cuts only a fixed fraction c_{i-1}, $0 < c_{i-1} < 1$, of the sticks of S_{i-1}, then it cuts only $\left(\prod_{j=1}^{i} c_j\right) \cdot n$ true segments. Since we want $\prod_{i=1}^{k} c_i$ to be a constant independent of $k = \Omega(\log n / \log\log n)$, a cut through S_{i-1} must

actually cut the vast majority of the sticks of S_{i-1}, which requires an asymmetric cyclic arrangement of sticks.

Consider an asymmetric cycle in S_i with three sticks such that two sticks correspond to i^2 copies of S_{i-1} (lying in disjoint skinny rectangles along the same segment), and the third stick corresponds to only one copy of S_{i-1} (Figure 6, left). If every line ℓ cutting S_i cuts through both heavy sticks, then ℓ cuts $\prod_{i=1}^{k} c_i = \prod_{i=1}^{k} 2i^2/(i^2+1) = O(1)$ fraction of the true segments. Recall that s cycle cut may split only one sticks in the cycle. This means that an acyclic cycle could be partitioned efficiently by cutting the light stick only, and none of the $2i^2$ copies of S_{i-1} along the heavy sticks (Figure 6, left).

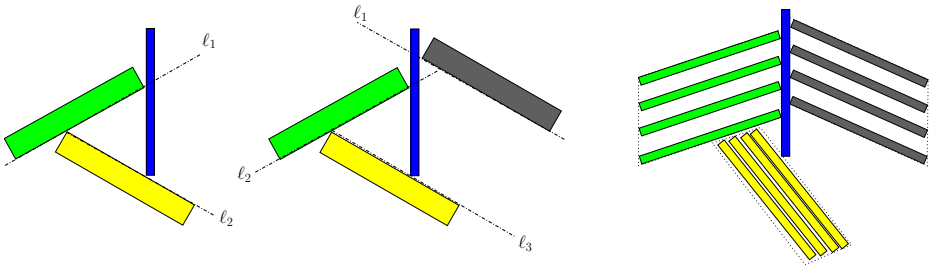

Figure 6. An asymmetric cycle (left), an asymmetric cycle with a one-step staircase structure on either side of the smallest rectangle (middle), and with a four step staircase (right).

We can overcome this difficulty with one additional idea: We want that if neither of the heavy sticks is cut by a line ℓ, then the light stick should be cut *many times*. Specifically, we want that any BSP of the stick arrangement at level i either cuts a heavy stick once or it cuts the light stick i^2 times. Both scenarios incur the same number of cuts on sticks, and on true segments, too. We can force the light stick to be cut many times by arranging i^2 copies of S_{i-1} in a staircase fashion on either side of the light stick. (Figure 6, middle and right). Technically, we break each heavy stick into i^2 distinct sticks in a staircase-like structure as depicted in Figure 6, right).

Tóth [2003c] shows that both the autopartition and partition complexity of this construction is $\Omega(n \log n / \log\log n)$. The partition complexity of n disjoint line segments in the plane is, thus, $O(n \log n)$ and $\Omega(n \log n / \log\log n)$.

Open Problem. *What is the partition complexity of n disjoint line segments in the plane*?

Convex cycles. The importance of cycles of anchored segments was already noted by de Berg, de Groot, and Overmars [de Berg et al. 1997b]. The segments of a cycle $(a_1, a_2, \ldots, a_k)$, however, can be quite irregular: A cycle cut can partition a non-anchored segment into k fragments (Figure 4). Tóth [2003b] defined a *convex cycle*, which is a cycle of anchored segments lying along sides of a con-

vex polygon (Figure 7). Cutting sequentially along segments of a convex cycle partitions any non-anchored segment into at most four pieces. To make sure that every set of segments in a cycle contains a subset forming a convex cycle, we *drop* the condition that $a_k \succ a_1$ in the definition of convex cycles (Figure 7, right).

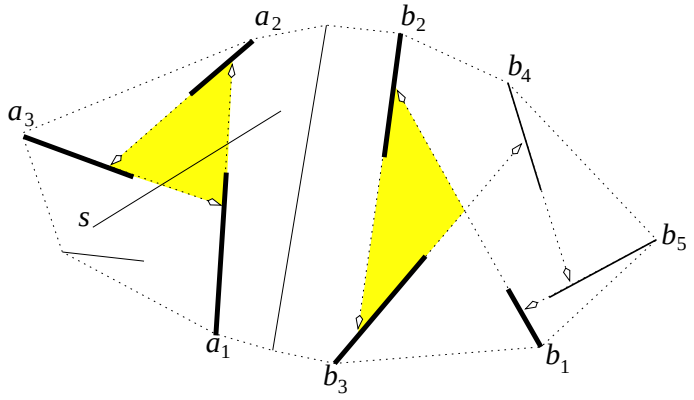

Figure 7. A convex cycle (a_1, a_2, a_3) and a cycle $(b_1, b_2, \ldots, b_5)$ that contains a convex cycle (b_1, b_2, b_3) with $b_3 \not\succ b_1$.

Limited number of line directions. Paterson and Yao [1992] proved, using anchored segments, that the partition complexity of n disjoint axis-parallel segments in the plane is at most $3n$. This was further improved to $2n - 1$ [Dumitrescu et al. 2004] (see Section 4). The proof techniques, however, do not generalize for cases where the segments have three or more distinct orientations. Tóth [2003b] proved recently that for any k, $k \in \mathbb{N}$, the autopartition complexity of n disjoint line segments with k distinct orientations is $O(nk)$, while their partition complexity is $O(n \log k)$. The latter bound is asymptotically tight for $k = O(1)$ and also for the lower bound construction described above, that is, where n disjoint segments have $k = 2^{O(\log n / \log \log n)}$ distinct orientations.

The partitioning algorithm for a set S of disjoint segments works in *phases*: Every phase i performs a BSP for a fixed set A_i of anchored segments such that no segment of S is cut more than $O(\log k)$ times (respectively, $O(k)$ times in the case of autopartitions). At an odd phase $2i - 1$, we let A_{2i-1} be the set of *lower anchored segments* at the current subproblem, that is, all (fragments of) segments whose lower endpoint is on the boundary of a cell. Similarly, at an even phase $2i$, A_{2i} is the set of all upper anchored segments. The result follows immediately, since if a segment $s \in S$ is cut in phase i then it is split into $c \log k$ fragments, $c \log k - 2$ of which are free cuts in their subproblem and 2 are anchored fragments. The two endings of s may be cut again $O(\log k)$ times in

phase $i+1$ and $i+2$ during a BSP for upper and lower anchored segments, but s is no longer cut in later phases.

A phase i calls a BSP subroutine P for the set A_i in each cell. $P(A_i)$ makes two types of cuts: (1) A *segment-cut* splits a cell along a line ℓ spanned by a segment $s \in A_i$. It is applied if ℓ does not cut any anchored segment in the cell. (2) A *cycle-cut* partitions along a convex cycle of anchored segments (as described in Section 2). Every cycle-cut is followed by a *cleanup step*, where we apply the subroutine P for subsets $B_i \subset A_i$ of those anchored segments of A_i which are split by the cycle cut. One can ensure that B_i has at most half as many distinct orientations as A_i in each resulting subcell (in case of autopartitions, B_i has at least one fewer distinct orientations than A_i in each subcell).

For every phase, the segment-cut and cycle-cut steps can be assigned into different levels according to recursive calls where subroutine P applied them. Since the number of distinct orientations decreases by a factor of two in each level, there are at most $\log k$ levels (at most k levels for autopartitions). One can also ensure by carefully isolating the region in which a cycle-cut splits anchored segments of A_i that every input segment is cut $O(1)$ times in each level. This implies that every segment is cut at most $O(\log k)$ times in total ($O(k)$ times for autopartitions).

4. Axis-aligned Binary Space Partitions

A k-dimensional object is *axis-aligned*, if each of its ℓ-dimensional faces is parallel to ℓ coordinate axes, $1 \le \ell \le k$. In an orthogonal coordinate system, axis-aligned objects are also called orthogonal, rectilinear, or isothetic. Note that BSPs are invariant under affine transformations of the Euclidean space, and so the partition complexity of axis-aligned objects is independent of the angles among the coordinate axes. BSPs for axis-aligned objects are important, because in many applications input objects are axis-aligned by nature or are modeled by their axis-aligned bounding boxes.

The simplest axis-aligned objects are the boxes, defined as a cross product $\prod_{i=1}^{d}[a_i, b_i]$ of d intervals in $\mathbb{R}^d$. The *extent dimensions* of a box B are the coordinates i where $a_i < b_i$, while in every non-extent dimension, $a_i = b_i$. An axis-aligned k-flat is a box with k extent dimensions. An axis-parallel line segment, for example, is a 1-flat.

Every partition algorithm in this section uses axis-aligned splitting hyperplanes only, such BSPs are said to be *axis-aligned*. Note that every cell in an axis-aligned BPS is an axis-aligned box. For most axis-aligned input classes, the axis-aligned BSPs are best possible *among all BSP*s ignoring constant or logarithmic factors, because they match the partition complexity known for the class. In some cases, lower bounds known for *axis-aligned BSP*s are higher by a constant or logarithmic factor than those for the general BSPs. Of course, it is

easy to construct instances where the smallest BSP and the smallest axis-aligned BSP have significantly different sizes.

Segments and rectangles in the plane. After Paterson and Yao [1992] showed that the partition complexity of n disjoint axis-parallel line segments in the plane is $O(n)$, it remained to determine the exact value of the coefficient hidden in the asymptotic notation. A partition algorithm due to d'Amore and Franciosa [1992], originally designed for axis-aligned boxes, always computes an axis-aligned BSP of size at most $2n-1$. Dumitrescu, Mitchell, and Sharir [2004] almost matched this bound with a construction (Figure 8, middle) whose axis-aligned partition complexity is $2n - o(n)$.

For disjoint axis-aligned rectangles, Berman, DasGupta, and Muthukrishnan [Berman et al. 2002] showed that the partition complexity is at most $3n-1$. Their partition algorithm is a blend of previous algorithms [Paterson and Yao 1992; d'Amore and Franciosa 1992], combined with a charging scheme. In [Dumitrescu et al. 2004] a construction is given where the axis-aligned partition complexity is at least $\frac{7}{3}n - o(n)$. The authors point out, however, that their construction cannot grant a lower bound that would match $3n - 1$, since it does have an axis-aligned BSP of size $2.444n + o(n)$.

Open Problem. *What is the coefficient of the linear term in the (axis-aligned) partition complexity of disjoint axis-aligned boxes in the plane*?

A tradeoff between size and height. Arya [2002] noticed that the height of a linear size *axis-aligned* BSP for disjoint axis-parallel segments cannot always be logarithmic. Examining the construction on Figure 8, left, he showed that there is a tradeoff between the size and the height of axis-aligned BSP trees. If the height of such a tree is h then its size is $\Omega(n \log n / \log h)$. A more complicated, but stronger formulation of his result says that the BSP-tree of height h must have size $\Omega(nr)$ in the worst case where $r \in \mathbb{N}$, $\sum_{i=0}^{r} \binom{h}{i} \leq n/8$.

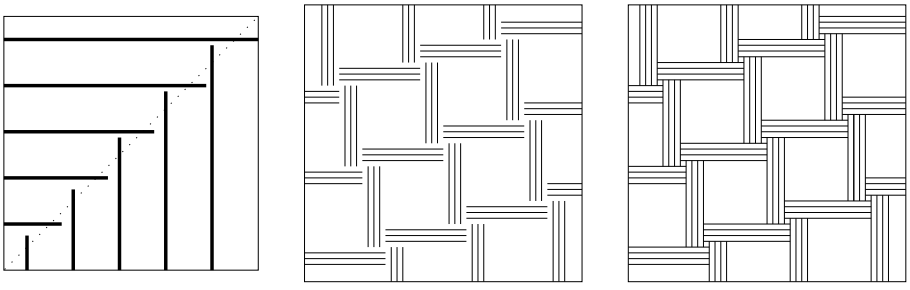

Figure 8. Arya's construction (left), Dumitrescu's construction with $2{\cdot}3{\cdot}3{\cdot}4 = 72$ disjoint axis-parallel line segments (middle), and the corresponding axis-aligned tiling of size $72 + 3^3 + (3+1)^2$ (right).

On the two extremities, this implies that an axis-aligned BSP-tree of size $O(n)$ has height $n^{\Omega(1)}$; and one of height $O(\log n)$ has size $\Omega(n \log n)$. The set of segments in Arya's construction, however, has linear partition complexity: All segments become anchored after a diagonal cut (dotted line on Figure 8, left). No tradeoff is known so far for the size and height of general BSP trees.

OPEN PROBLEM. *Is there a tradeoff between the size and the height of the BSP trees for disjoint line segments (with arbitrary orientations)?*

k-flats in d-dimensions. Paterson and Yao [1992] proved that the axis-aligned partition complexity of n disjoint line segments in $\mathbb{R}^d$, $d > 2$, lies between $(n/d)^{d/(d-1)} + n$ and $d \cdot (n/d)^{d/(d-1)} + n$. There has been no improvement on the coefficients on the main terms so far. Their lower bound construction is a cubic grid $\prod_{i=1}^{d} \{1, 2, \ldots, (n/d)^{1/(d-1)}\}$ where d lines of distinct orientation stab each grid cell. Their upper bound can be obtained by a round-robin partition algorithm applying median cuts in all d directions.

Dumitrescu, Mitchell, and Sharir obtained an $O(n^{d/(d-k)})$ upper bound for the partition complexity of n axis-aligned k-flats in $\mathbb{R}^d$ [Dumitrescu et al. 2004]. It is not difficult to see that a round-robin cutting scheme delivers this bound, too: Let every round consist of d recursive cuts in all d directions such that every splitting hyperplane cuts through the median of the vertex set of the fragments clipped to the cell. The total number of fragments may increase by a factor of 2^k in each round, because every (fragment of a) k-flat can be split along each of its k extents. At the same time, the number of fragments in one subproblem decreases by a factor of 2^{d-k}. So there are at most $(\log_2 n)/(d-k)$ rounds, and the BSP size is $O(n \cdot n^{k/(d-k)})$. Dumitrescu et al. [2004], in fact, applied a somewhat different partition scheme where all cuts for each hyperplane direction are made simultaneously. They also give a matching lower bound of $\Omega(n^{d/(d-k)})$ for the case where $k < d/2$.

The formula $\Theta(n^{d/(d-k)})$ is no longer valid for disjoint objects if $k \geq d/2$ (the axis-aligned partition complexity is $\Theta(n^{5/3})$ for $k = 2$ and $d = 4$ [Dumitrescu et al. 2004]). The reason for this is that *disjointness* plays no role if $k < d/2$, but it is restrictive for $k \geq d/2$. Indeed, if two objects whose extent dimensions together do not contain all d dimensions intersect, then they have a common non-extent dimension where their coordinates coincide. Any system of k-flats in $\mathbb{R}^d$, $k < d/2$, can be perturbed into a pairwise disjoint set by translating each object independently by a small random vector. If the random vectors are sufficiently small, then this perturbation does not decrease the partition complexity (every BSP for the perturbed set translates into a BSP of less than or equal size for the original input). Two flats whose extent dimensions contain all d dimensions may not always be separated by an arbitrarily small perturbation. The simplest examples are a set of axis-parallel segments in the plane and in $\mathbb{R}^3$, respectively: A small random perturbations removes all segment-segment intersection in three-space, but intersections may prevail in the plane.

The disjointness requirement is more apparent for the following generalization of free cut segments in higher dimensions: An axis-aligned box is a *stabber* in dimension i if its extent contains the extent of the bounding box of the input in dimension i. It is an *M-stabber* for a set $M \subseteq \{1, 2, \ldots, d\}$, if it is a stabber in every dimension $i \in M$; we can also say that it is a k-stabber if $k = |M|$. An M_1-stabber and an M_2-stabber intersect if $M_1 \cup M_2 = \{1, 2, \ldots, d\}$. Stabbers were defined by Dumitrescu, Mitchell, and Sharir [Dumitrescu et al. 2004]. Later Hershberger, Suri, and Tóth [Hershberger et al. 2004] gave an alternative definition saying that a box is k-stabber, if all its j-faces, $j < k$, lie on the boundary of the bounding box.

The concept of stabbers is crucial in any axis-aligned BSP algorithm. Let us mention just two of their useful properties: (i) A $(d-1)$-stabber is a free-cut in $\mathbb{R}^d$. (ii) Apart from at most 2^k fragments incident to the vertices of b, every fragment of a k-flat b is a ℓ-stabber for some $1 \leq \ell \leq k$. A set of disjoint stabbers corresponds to a pairwise intersecting set system: Let us map every stabber b to the set $\varphi(b) \subset \{1, 2, \ldots, d\}$ of dimensions in which b is *not* a stabber. $\{\varphi(b) : b\}$ is a pairwise intersecting set system; and vice versa: Given a pairwise intersecting set system D on $\{1, 2, \ldots, d\}$, there is a set of stabbers in $\mathbb{R}^d$ whose image under φ is D.

Tilings. In every lower bound construction where the partition complexity is high, the free space surrounding the input objects is also complex in the sense that its smallest convex partition is often superlinear. Researchers expected that if there is no space among the (convex) input objects (or, if the convex partition of the free space has the same order of magnitude as the input), then the partition complexity would be significantly smaller. A *tiling* in $\mathbb{R}^d$ is a set of d-dimensional objects that partition the space. An *axis-aligned tiling* is a set of full-dimensional axis-aligned boxes that partition the space.

Already in the plane, the worst case partition complexity of axis-aligned tilings is smaller than that of disjoint boxes. Berman, DasGupta, and Muthukrishnan [Berman et al. 2002] showed that every axis-aligned tiling of size n has an axis-aligned BSP of size at most $2n$ (recall that the axis-aligned partition complexity of disjoint rectangles is at least $\frac{7}{3}n - o(n)$ [Dumitrescu et al. 2004]). This is asymptotically optimal, since a construction from [Dumitrescu et al. 2004], originally designed for disjoint line segments, can be converted to an axis-aligned tiling. Their lower bound construction for axis-parallel line segments consists of $2k(k+1)$ bundles of k parallel lines (Figure 8, middle). These $2k^3 + 2k^2)$ segments can be blown up to skinny axis-aligned boxes such that each bundle fills an axis-aligned rectangle, and the free space between the rectangles can be covered by $k^2 + (k+1)^2 = O(k^2)$ interior-disjoint axis-aligned boxes (Figure 8, right). A total of $n = 2k^3 + O(k^2)$ boxes tile the plane, and their axis-aligned BSP cannot be smaller than that of the original k^3 segments we have started with, that is, at least $4k^3 - O(k^2)$.

The difference in the partition complexity of disjoint and plane filling rectangles was only the coefficient of the linear term. In three- and higher dimensions the upper bounds on the partition complexity of tilings is smaller in exponent than that of disjoint boxes. Hershberger and Suri [2003] observed that every tiling where all $(d-3)$-dimensional faces of every tile lie on the convex hull has a linear size axis-aligned BSP. (This is not true for disjoint axis-aligned boxes, in general: In the known worst-case constructions, all $(d-3)$-dimensional faces lie on the convex hull.) This implies that it is enough to partition an axis-aligned tiling until every cell is empty of $(d-3)$-dimensional faces, and then a linear size BSP in each cell results in a BSP for the input tiling.

Hershberger, Suri, and Tóth [Hershberger et al. 2004] apply a round-robin BSP for the $(d-3)$-dimensional faces of an axis-aligned tiling in $\mathbb{R}^d$. One can show that the number of $(d-3)$-faces in each subproblem decreases by a factor of 2^3. Therefore the round-robin scheme terminates in $\frac{1}{3}\log n + O(1)$ rounds. In each round of the round-robin scheme, every $(d-2)$-face is split into at most 2^{d-2} fragments. In the course of $\frac{1}{3}\log n + O(1)$ rounds, every $(d-2)$-face is partitioned into $(2^{d-2})^{\frac{1}{3}\log n + O(1)} = O(n^{(d-2)/3})$ fragments. Since the $(d-1)$-faces that become free-cut in their subproblem are not fragmented any further, every $(d-1)$-face and every tile is partitioned into $O(n^{(d-2)/3})$ fragments, as well. The round-robin scheme for $(d-3)$-faces partitions the axis-aligned tiling in $\mathbb{R}^d$ into $n \cdot O(n^{(d+1)/3}) = O(n^{(d+1)/3})$ fragments such that no $(d-3)$-face lies in the interior of a cell. By applying a linear size BSP for the tiling clipped in each resulting cell, we obtain a BSP of size $O(n^{(d+1)/3})$ for an axis-parallel tiling with n tiles in $\mathbb{R}^d$. By contrast, the best known upper bound on the partition complexity of n disjoint full-dimensional boxes in $\mathbb{R}^d$ is $O(n^{d/2})$ only.

In dimensions $d = 3$, the partition complexity of axis-aligned tilings of size n is $O(n^{4/3})$, which is tight by a construction of Hershberger and Suri [2003]. They start with the lower bound construction of Paterson and Yao [1992] for axis-parallel line segments in three space, which consists of $3k^2$ axis-parallel line segments such that three segments of distinct directions stab every unit cube in $[0,k] \times [0,k] \times [0,k]$ grid. Then they replace every segment by an air-tight bundle of k axis-aligned skinny boxes, and fill the free space between the bundles by $2k^3$ pairwise disjoint boxes. They obtain a tiling with $n = 5k^3$ boxes. By the argument of Paterson and Yao, each grid cell corresponds to a cut through a bundle of k skinny boxes, which gives a total of $k \cdot k^3 = \Theta(n^{4/3})$ cuts of tiles. Generalizing this idea and ideas of Dumitrescu et al. [2004], Hershberger, Suri, and Tóth [Hershberger et al. 2004] gave a construction in d-dimensions for which the axis-aligned partition complexity is $\Omega(n^{\beta(d)})$, where $\lim_{d\to\infty} \beta(d) = (1+\sqrt{5})/2 = 1.618$. Apart from the staggering gap between the upper and lower bounds for $d \geq 4$, many problems remain open on BSPs of axis-aligned and of general tilings.

Open Problem. *What is the (axis-aligned) partition complexity of n space filling axis-aligned boxes in $\mathbb{R}^d$, $d \geq 4$?*

Open Problem. *What is the partition complexity of n space filling simplices in $\mathbb{R}^d$, $d \geq 3$?*

Open Problem. *What is the partition complexity of n tetrahedra that form the triangulation of a convex polytope in d-space, $d \geq 3$?*

Fat rectangles. Another characteristics of the lower bound constructions for n axis-aligned rectangles in $\mathbb{R}^3$ is that the rectangles are long and skinny (they are axis-parallel line segments fattened by a small $\varepsilon > 0$ in all extent dimensions). It was reasonable to expect that disjoint fat objects have small partition complexity. An axis-aligned k-flat is *fat* if the ratio of a largest inscribed and a smallest circumscribed cubes[1] in the k-dimensional space spanned by the object is bounded by a constant.

De Berg [1995] proved that the partition complexity of *full dimensional* disjoint fat objects is linear in every dimension. This is true for any set of disjoint fat objects even if they are not axis-aligned (Section 5 deals with further extensions of this result). On the other hand, n disjoint fat but not axis-aligned rectangles may have $\Theta(n^2)$ partition complexity [Paterson and Yao 1990].

Agarwal et al. [2000b; 1998] considered n disjoint axis-aligned fat 2-flats in $\mathbb{R}^3$ and proved that their partition complexity is $n \cdot 2^{O(\sqrt{\log n})}$. Tóth [2003a] improved this upper bound and gave an algorithm to compute a BSP of $O(n \log^8 n)$ size and $O(\log^4 n)$ height. He also gave a lower bound construction for which the axis-aligned partition complexity is $\Omega(n \log n)$.

The main difficulty in dealing with a fat rectangle r in three-space is that the fragments of r clipped to a cell of a subproblem is not necessarily fat anymore. Every fragment, however, belongs to one of the following three types, each of which has some useful properties that help our partitioning algorithm.

For an input rectangle r and a cell C, we say that the fragment $r' = r \cap C$ is
- a *corner* if r' is incident to a vertex of the original axis-parallel rectangle r,
- *bridge* if one extent of r' is the same as the corresponding extent of C (r' is a 1-stabber) and the other lies in the interior of the corresponding extent of C (see Figure 9), or
- a *shelf* if one extent of r' is the same as the corresponding extent of C and the other is incident to one endpoint of the other extent of C (see Figure 9).

Every rectangle has only four vertices, therefore in every level of the BSP tree, at most four sub-problems contain corners of an input rectangle r. This implies that the number of cuts on corners of an input r is bounded by (four times) the height of the BSP. A *bridge* fragment r' of a fat rectangle r is not necessarily

[1]Traditionally, an object is called *fat* if the ratio of the radii of a circumscribed and an inscribed *ball* is bounded by a constant. Our definition, in terms of cubes, is equivalent to the standard definition (with a different constant threshold).

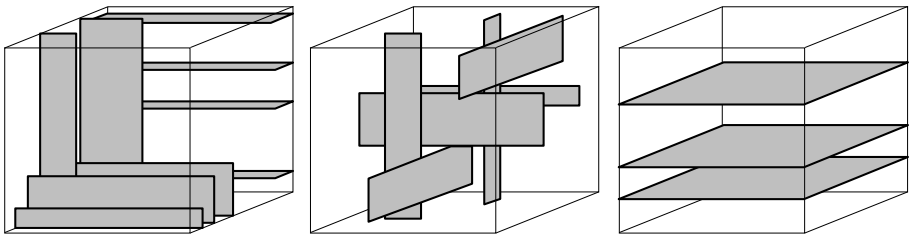

Figure 9. Shelves, bridges, and free-cuts clipped to a cell C in $\mathbb{R}^3$.

fat. It still preserves part of the fatness of r, a property we call *semi-fatness*: If r is α-fat then the extent of r' in the interior of that of C is at most α times longer than the extent containing the corresponding extent of C. The semi-fatness allows us to separate bridges of different orientations (see subroutine P_3 below). Finally, there is an $O(n \log n)$ size BSP for n disjoint shelves such that it partitions *every* rectangle into at most $O(\log^2 n)$ fragments.

We outline the main ideas to construct a $O(\log^4 n)$ height BSP for disjoint axis-aligned fat rectangles. The partition algorithm is composed of a four level hierarchy of BSPs: We give a glimpse of each level and explain the lowest level (a BSP for shelves) in detail below.

The main algorithm, P_1, makes cuts recursively along the median xy-plane of the vertices of the input rectangles. Since the number of rectangle vertices in a cell is halved in each step, P_1 terminates in $O(\log n)$ rounds. After every median cut of P_1, a *cleanup step* is applied in each cell C, which is a BSP *for the stabbers* w.r.t. C and partitions *every rectangle* into $O(\log^6 n)$ fragments. If the cleanup step P_2 makes a cut along a stabber $r \cap C$, then subsequent median cuts of P_1 cannot partition $r \cap C$ anymore. This ensures that the median cuts performed by P_1 may split any rectangle $O(\log n)$ times only.

The BSP P_2 for a set S of stabbers makes cuts recursively along the median xz-plane of the vertex set of S. Each median cut of P_2 is followed by another cleanup subroutine, P_3, which in each cell separates the stabbers of S from other stabbers (i.e., fragments that became stabbers due to the last median cut by P_2). Finally, P_3 makes cuts along yz-planes, and calls a BSP P_4 for all shelves after every such step. We describe the BSP P_4 for a set S of n shelves in a cell. P_4 has $O(n \log n)$ size and it partitions every rectangle into $O(\log n)$ fragments.

Consider a set $S_\perp$ of $n_\perp$ disjoint shelves in a cell C. Assume for simplicity that they are all adjacent to the bottom side of C. The subroutine BSP for $S_\perp$ is a simple recursive algorithm: We split C by a horizontal plane h through the highest upper edge of a shelf in $S_\perp$, then we split the subcell below h along the highest shelf (a free cut) and along the median shelf, and recursively call this subroutine for each resulting subproblem (Figure 10 shows an example). The algorithm terminates in $\log n_\perp$ rounds because the number of shelves in a subproblem is halved in each round. None of the shelves is split, and we show that

every axis-parallel segment disjoint from these shelves is cut $O(\log n_\perp)$ times: A segment parallel to stabbing extent of the shelves is never cut, a vertical segment is cut at most once in every round, which amounts to $O(\log n_\perp)$ cuts. Lastly, consider a segment s orthogonal to the shelves. In subproblems where s is a stabbing segment, it lies above the highest shelf and so it cannot be cut at all. In a subproblem where an endpoint of s lies in the interior of a cell, s be cut by the splitting planes along the median shelf, which totals to at most $2\log n_\perp$ cuts during the entire subroutine.

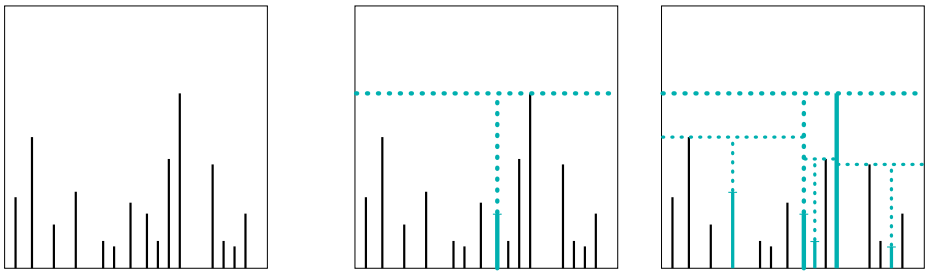

Figure 10. First two rounds of the BSP for shelves.

This was a BSP for shelves along the one side of a cell C. We still need to integrate six BSPs for each of the six sides of C: If we naïvely call the subroutine six times sequentially for the subproblems and for each side then the number of cuts on axis-parallel segments might be $(O(\log n))^6$. We use, instead, the concept of *overlay of BSPs* to keep the number of cuts on every segment $O(\log n)$. We compute independently six BSPs for the shelves adjacent to each side, then we "overlay" them one after another: A cut along a splitting plane made in one BSP may incur splits in several cells resulting from previous BSPs. The combination is still a valid BSP because every subproblem is split by a hyperplane. The number of cuts on each segment is the sum of the cuts made by each of the six BSPs rather than their product. The overlay of BSPs is a key concept to keep the number of cuts low throughout this algorithm.

In general we would like know the partition complexity of n disjoint axis-aligned fat k-flats in $\mathbb{R}^d$, $1 < k < d$. It is easy to show a lower bound of $\Omega(n^{(d-k+1)/(d-k)})$. Indeed, consider a worst case construction for n axis-parallel line segments in $\mathbb{R}^{d-k+1}$, whose partition complexity is $\Omega(n^{(d-k+1)/(d-k)})$ by Paterson and Yao [1992]. Embed it into a $(d-k+1)$-dimensional affine subspace H of $\mathbb{R}^d$ and expand it to a k-dimensional square orthogonally to H. We obtain n pairwise disjoint k-dimensional squares in $\mathbb{R}^d$. The size of any BSP for this set is $\Omega(n^{(d-k+1)/(d-k)})$. We expect that this bound can be attained apart from a polylogarithmic factor, that is, out of the k extents of a k-dimensional fat rectangle, $k-1$ extents are essentially redundant with respect to the partition complexity.

OPEN PROBLEM. *What is the (axis-aligned) partition complexity of n disjoint fat axis-aligned k-flats in $\mathbb{R}^d$, $1 < k < d$?*

5. From Fatness to Guardable Scenes

The worst case constructions for the partition complexity of general and axis-aligned disjoint objects in 3-space use long and skinny input objects. This observation suggests that disjoint fat objects must have small partition complexity. A full dimensional object is fat if the ratio of an circumscribed and inscribed ball is bounded by a constant. Fat objects are often easier to handle. Among other things, the union of fat objects have small combinatorial complexity in many cases [Efrat and Sharir 2000; Pach and Tardos 2002; Pach et al. 2003]. A result that every set of disjoint fat objects has linear partition complexity [de Berg 1995] initiated a flurry of work on extensions defining new object classes which also have linear partition complexity by analogous arguments.

Fat full dimensional objects De Berg, de Groot, and Overmars [1997b] have proved that the partition complexity of n disjoint convex fat objects in the plane is $O(n)$. De Berg [1995] has later found an elegant proof that establishes linear partition complexity for disjoint full dimensional fat polyhedra with constant number of vertices (e.g., fat tetrahedra) in $\mathbb{R}^d$, $d \in \mathbb{N}$, as well. His partition algorithm generates the BSP tree in $O(n \log n)$ time for any $d \in \mathbb{N}$, where the constant of proportionality depends on the dimension. We briefly describe his method below.

Consider an orthogonal coordinate system and then compute the axis-aligned bounding box of every fat object. Generate a BSP *for the set V of vertices of all bounding boxes* by repeating one of the following two steps starting with a bounding cube C of V: (i) Partition the cubic cell C into 2^d congruent cubes if at least two sub-cubes have non-empty intersection with V. (ii) Otherwise every point of $V \cap C$ lies in one of the subcubes C'. In this case, partition C along the sides of the bounding cube of $(V \cap C) \cup c'$ and (subsets of) where c' is the common vertex of C and C'. Every subproblem that still contains a vertex of V is a cube (allowing recursion). The subcells of the complement of the bounding box of $(V \cap C) \cup c'$ in C are not necessarily fat but they can be expressed as the union of $O(1)$ cubes. Every cell of the resulting partition is empty of bounding box vertices of the fat input objects. This implies that every resulting cell intersects only a constant number of input objects. In each subproblem, one can complete the BSP with a constant number of cuts if every fat object is either convex or polyhedral with constant combinatorial complexity.

Note that the argument above used only that the input objects are fat, and that they are pairwise disjoint. The extensions of disjoint fat scenes define properties of the whole input scene rather than properties of individual objects.

Relaxations. De Berg [2000] noticed that the above argument goes through verbatim if, instead of disjoint fat objects, he requires a much weaker property: uncluttered. A set of objects is *uncluttered* if every axis-aligned cube intersecting more than a constant number of input objects contains a vertex of the bounding box of an input object. It is easy to see that every set of disjoint fat objects is uncluttered. The converse is false and so the class of uncluttered sets is strictly larger than that of fat objects.

De Berg et al. [2003] further explored possible relaxations of fatness and uncluttered that still grants linear partition complexity. A set of n objects is *guardable* if there are $O(n)$ *guard* points such that every axis-aligned box that intersects more than a constant number of objects must contain a guard point. Every uncluttered set is guardable with the obvious choice for guard points being the bounding box vertices. On the other hand, there are guardable sets that are cluttered (Figure 11, right). Unlike fatness and uncluttered [de Berg et al. 2002], however, there is no polynomial time algorithm known to decide if a set of objects is guardable or not, let alone finding a guard points for a given guardable set [de Berg et al. 2003].

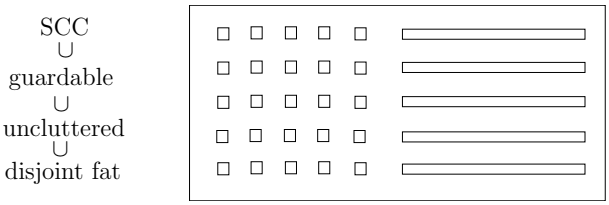

Figure 11. The relations between the classes of polygonal scenes (left), and a guardable but cluttered set of boxes (right).

Another property likely to imply low partition complexity is the *small simple-cover complexity* (SSC), defined originally by Mitchell, Mount, and Suri [Mitchell et al. 1997]. We cite the definition from [de Berg et al. 2002]: A set of n objects has *SSC*, if their convex hull can be covered by $O(n)$ balls such that the number of objects intersecting each ball is bounded by a small constant. De Berg et al. [2003] proved that in the plane a set of objects has SCC if and only if it is guardable. In higher dimensions, however, SCC is a strictly broader property than guardability. It is not known if there is a small size BSP for an SCC set.

Open Problem. *What is the partition complexity of a set of objects with small simple-cover complexity in* $\mathbb{R}^d$, $d \geq 3$?

Tobola and Nechvile [2003] extended the definition of uncluttered to sets of axis-aligned objects where the neighborhood of every object is uncluttered after a linear transformation. A set S of axis-aligned objects is *locally uncluttered* if for every object $s \in S$ the fragments of objects clipped in a neighborhood $N(s)$

of s are uncluttered after a linear transformation T_s. The neighborhood $N(s)$ is the axis-aligned box obtained from s by fattening each extent with a multiple of the longest extent of s. Tobola and Nechvile [2003] show that every locally uncluttered set of n objects in $\mathbb{R}^d$ has an $O(n)$ size axis-aligned BSP.

6. Kinetic Binary Space Partitions

The first motivation for binary space partitions was efficient hidden-surface removal from a *moving* viewpoint. The input objects of the BSP were, however, assumed to be static. Recent research on BSPs for moving objects was set in the model of *kinetic data structures* (KDS) of [Basch et al. 1999]. In this model, objects move continuously along a given trajectory (flight plan), typically along a line or a low degree algebraic curve. The splitting hyperplanes are defined by faces of the input objects, and so they move continuously, too. The BSP is updated only at discrete *events*, though, when the combinatorial structure of the BSP changes.

In the KDS paradigm, the goal is to maintain at all times a BSP whose size and height is below a reasonable threshold, which is usually close to the partition complexity available for static input. Efficiency of the algorithms are measured by additional parameters: the total *space* required during the algorithm, *total time*, which is broken down into the *total number of events* and the maximum *update time* for an event.

First Agarwal et al. [2000c] studied the BSP in the KDS setting. They consider n moving line segments in the plane such that the trajectory of every segment endpoint is a constant degree polynomial and the segments are pairwise disjoint at all times. They design a randomized algorithm to maintain a BSP of expected size $O(n \log n)$ in $O(n \log n)$ space, the number of events is $O(n^2)$ and each event requires an expected $O(\log n)$ update time.

Instead of adapting a previously known static BSP algorithm [Paterson and Yao 1990] to KDS paradigm, they start out from a randomized incremental BSP, which is an adaptation of the *vertical (trapezoid) decomposition* of Mulmuley [1990] and Seidel [1991]. They fix a random permutation of the segments, which is the only random choice in the algorithm. In their BSP, every splitting line is either vertical or a free cut (i.e., lies along an input segment); they call such BSPs *cylindrical*. By restricting the possible splitting planes, they reduce the number of possible event types: In fact, the only possible combinatorial event is that a segment endpoint starts or stops crossing a vertical splitting line. A careful analysis of the effects of these events on the BSP tree establishes the bounds on the update time and the total space requirement.

All known kinetic BSPs are based on vertical decomposition. This choice keeps the *types* of the possible combinatorial events under control, but the *number* of events can be suboptimal. If the x coordinates of every two segment endpoints are swapped during the motion, then the number of events can be $\Theta(n^2)$ for any

cylindric kinetic BSP. For arbitrary kinetic BSPs, no such lower bound is known. Agarwal et al. [2000a] established lower bounds on the combinatorial changes every kinetic BSP has to go through: They give a set P of n points in the plane all moving along axis-parallel lines with constant velocity such that any kinetic BSP for P experiences $\Omega(n^{3/2})$ combinatorial changes.

Open Problem. *How many combinatorial changes occur in the kinetic BSP of n points moving with constant velocity in the plane*?

Agarwal, Erickson, and Guibas [Agarwal et al. 1998] adapted the cylindrical BSP method to potentially intersecting line segments. They maintain a BSP of size $O(n \log n + k)$ and height $O(\log n)$ in total time $O(n \log^2 n + k \log n)$, all in expectation, where k is the number of intersecting segment pairs. They apply this algorithm to derive a kinetic BSP for n disjoint triangles in $\mathbb{R}^3$, for which they maintain a randomized BSP of expected $O(n^2)$ size and $O(\log n)$ height in $O(n^2 \log n)$ total time.

De Berg, Comba, and Guibas [de Berg et al. 2001] give a *deterministic* kinetic BSP for disjoint segments in the plane. Their algorithm is based on a *static* BSP algorithm due to Paterson and Yao [1990] using segment trees. They maintain a BSP of size $O(n \log n)$ and height $O(\log^2 n)$. The segment tree changes every time when the x-coordinates of two segment endpoints swap. So if every endpoint moves along a constant-degree polynomial trajectory, then the number of events is $O(n^2)$, each requiring $O(\log^2 n)$ update time. The average update time is $O(\log n)$, though, similarly to the randomized algorithm of [Agarwal et al. 2000c].

Open Problem. *Is there a kinetic BSP of size $O(n \operatorname{polylog} n)$ and height $O(\operatorname{polylog} n)$ with $o(n^2)$ events for n disjoint line segments whose endpoints are moving along straight line trajectories*?

Acknowledgments

I thank the volume editors, Eli Goodman, János Pach, and Emo Welzl, for their encouragement to compile this survey paper. I am also indebted to the anonymous referees whose critical comments helped tremendously to improve the presentation of this paper.

Work on this survey paper was done while at the Department of Computer Science, University of California at Santa Barbara.

References

[Agarwal and Matoušek 1992] P. K. Agarwal and J. Matoušek, "On range searching with semialgebraic sets", pp. 1–13 in *Mathematical foundations of computer science* (Prague, 1992), Lecture Notes in Comput. Sci. **629**, Springer, Berlin, 1992.

[Agarwal and Suri 1998] P. K. Agarwal and S. Suri, "Surface approximation and geometric partitions", *SIAM J. Comput.* **27**:4 (1998), 1016–1035.

[Agarwal et al. 1998] P. K. Agarwal, J. Erickson, and L. J. Guibas, "Kinetic binary space partitions for intersecting segments and disjoint triangles (extended abstract)", pp. 107–116 in *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, 1998), ACM, New York, 1998.

[Agarwal et al. 2000a] P. K. Agarwal, J. Basch, M. de Berg, L. J. Guibas, and J. Hershberger, "Lower bounds for kinetic planar subdivisions", *Discrete Comput. Geom.* **24**:4 (2000), 721–733.

[Agarwal et al. 2000b] P. K. Agarwal, E. F. Grove, T. M. Murali, and J. S. Vitter, "Binary space partitions for fat rectangles", *SIAM J. Comput.* **29**:5 (2000), 1422–1448.

[Agarwal et al. 2000c] P. K. Agarwal, L. J. Guibas, T. M. Murali, and J. S. Vitter, "Cylindrical static and kinetic binary space partitions", *Comput. Geom.* **16**:2 (2000), 103–127.

[Ar et al. 2000] S. Ar, B. Chazelle, and A. Tal, "Self-customized BSP trees for collision detection", *Comput. Geom. Theory Appl.* **15**:1-3 (2000), 91–102.

[Ar et al. 2002] S. Ar, G. Montag, and A. Tal, "Deferred, self-organizing BSP trees", *Comput. Graph. Forum* **21**:3 (2002), 269–278.

[Arya 2002] S. Arya, "Binary space partitions for axis-parallel line segments: size-height tradeoffs", *Inform. Process. Lett.* **84**:4 (2002), 201–206.

[Arya et al. 2000] S. Arya, T. Malamatos, and D. M. Mount, "Nearly optimal expected-case planar point location", pp. 208–218 in *41st Annual Symposium on Foundations of Computer Science* (Redondo Beach, CA, 2000), IEEE Comput. Soc. Press, Los Alamitos, CA, 2000.

[Asano et al. 2003] T. Asano, M. de Berg, O. Cheong, L. J. Guibas, J. Snoeyink, and H. Tamaki, "Spanning trees crossing few barriers", *Discrete Comput. Geom.* **30**:4 (2003), 591–606.

[Ballieux 1993] C. Ballieux, "Motion planning using binary space partitions", Tech. Rep. Inf/src/93-25, Utrecht University, 1993.

[Basch et al. 1999] J. Basch, L. J. Guibas, and J. Hershberger, "Data structures for mobile data", *J. Algorithms* **31**:1 (1999), 1–28.

[Batagelo and Jr. 1999] H. C. Batagelo and I. C. Jr., "Real-time shadow generation using BSP trees and stencil buffers", pp. 93–102 in *Proc. 12th Brazilian Sympos. Comp. Graph. and Image Processing* (Campinas, SP), edited by J. Stolfi and C. L. Tozzi, IEEE, Los Alamitos, CA, 1999.

[de Berg 1993] M. de Berg, *Ray shooting, depth orders and hidden surface removal*, Lecture notes in computer science **703**, Springer, Berlin, 1993.

[de Berg 1995] M. de Berg, "Linear size binary space partitions for fat objects", pp. 252–263 in *Algorithms—ESA '95* (Corfu, 1995), Lecture Notes in Comput. Sci. **979**, Springer, Berlin, 1995.

[de Berg 2000] M. de Berg, "Linear size binary space partitions for uncluttered scenes", *Algorithmica* **28**:3 (2000), 353–366.

[de Berg et al. 1997a] M. de Berg, M. M. de Groot, and M. H. Overmars, "Perfect binary space partitions", *Comput. Geom.* **7**:1-2 (1997), 81–91.

[de Berg et al. 1997b] M. de Berg, M. de Groot, and M. Overmars, "New results on binary space partitions in the plane", *Comput. Geom.* **8**:6 (1997), 317–333.

[de Berg et al. 2001] M. de Berg, J. Comba, and L. J. Guibas, "A segment-tree based kinetic BSP", pp. 134–140 in Proc. 17th ACM Sympos. on Comput. Geom. (Somerville, MA, 2001), ACM Press, 2001.

[de Berg et al. 2002] M. de Berg, M. J. Katz, A. F. van der Stappen, and J. Vleugels, "Realistic input models for geometric algorithms", *Algorithmica* **34**:1 (2002), 81–97.

[de Berg et al. 2003] M. de Berg, H. David, M. J. Katz, M. Overmars, A. F. van der Stappen, and J. Vleugels, "Guarding scenes against invasive hypercubes", *Comput. Geom.* **26**:2 (2003), 99–117.

[Berman et al. 2002] P. Berman, B. Dasgupta, and S. Muthukrishnan, "Exact size of binary space partitionings and improved rectangle tiling algorithms", *SIAM J. Discrete Math.* **15**:2 (2002), 252–267.

[Buchele 1999] S. F. Buchele, *Three-dimensional binary space partitioning tree and constructive solid geometry tree construction from algebraic boundary representations*, Ph.D. thesis, University of Texas, Austin, 1999.

[Chazelle 1984] B. Chazelle, "Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm", *SIAM J. Comput.* **13**:3 (1984), 488–507.

[Chin and Feiner 1989] N. Chin and S. Feiner, "Near real-time shadow generation using BSP trees", *Computer Graphics* **23**:3 (1989), 99–106.

[d'Amore and Franciosa 1992] F. d'Amore and P. G. Franciosa, "On the optimal binary plane partition for sets of isothetic rectangles", *Inform. Process. Lett.* **44**:5 (1992), 255–259.

[Dumitrescu et al. 2004] A. Dumitrescu, J. S. B. Mitchell, and M. Sharir, "Binary space partitions for axis-parallel segments, rectangles, and hyperrectangles", *Discrete Comput. Geom.* **31**:2 (2004), 207–227.

[Duncan et al. 2001] C. A. Duncan, M. T. Goodrich, and S. Kobourov, "Balanced aspect ratio trees: combining the advantages of k-d trees and octrees", *J. Algorithms* **38**:1 (2001), 303–333.

[Efrat and Sharir 2000] A. Efrat and M. Sharir, "On the complexity of the union of fat convex objects in the plane", *Discrete Comput. Geom.* **23**:2 (2000), 171–189.

[Fuchs et al. 1980] H. Fuchs, Z. M. Kedem, and B. Naylor, "On visible surface generation by a priori tree structures", *Comput. Graph.* **14**:3 (1980), 124–133.

[Hershberger and Suri 2003] J. Hershberger and S. Suri, "Binary space partitions for 3D subdivisions", pp. 100–108 in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Baltimore, MD, 2003), ACM, New York, 2003.

[Hershberger et al. 2004] J. Hershberger, S. Suri, and C. D. Tóth, "Binary space partitions of orthogonal subdivisions", pp. 230–238 in *Proc. 20th Sympos. Comput. Geom.* (Brooklyn, NY, 2004), ACM Press, 2004.

[Mata and Mitchell 1995] C. S. Mata and J. S. B. Mitchell, "Approximation algorithms for geometric tour and network design problems", pp. 360–369 in *Proc. 11th Sympos. Comput. Geom.* (Vancouver, 1995), ACM Press, 1995.

[Mitchell et al. 1997] J. S. B. Mitchell, D. M. Mount, and S. Suri, "Query-sensitive ray shooting", *Internat. J. Comput. Geom. Appl.* **7**:4 (1997), 317–347.

[Mulmuley 1990] K. Mulmuley, "A fast planar partition algorithm, I", *J. Symbolic Comput.* **10**:3-4 (1990), 253–280.

[Murali 1998] T. M. Murali, *Efficient hidden-surface removal in theory and in practice*, Ph.D. thesis, D. Thesis, Department of Computer Science, Brown University, Providence, RI, 1998.

[Naylor and Thibault 1986] B. Naylor and W. Thibault, "Application of BSP trees to ray-tracing and CSG evaluation", Tech. Rep. GIT-ICS 86/03, Georgia Institute of Tech., 1986.

[Naylor et al. 1990] B. Naylor, J. A. Amanatides, and W. Thibault, "Merging BSP trees yields polyhedral set operations", *Comput. Graph.* **24**:4 (1990), 115–124.

[Nguyen 1996] V. H. Nguyen, *Optimal binary space partitions for orthogonal objects*, Diss. 11818, ETH Zürich, 1996.

[Nguyen and Widmayer 1995] V. H. Nguyen and P. Widmayer, "Binary space partitions for sets of hyperrectangles", pp. 59–72 in *Algorithms, concurrency and knowledge (Pathumthani, 1995)*, Lecture Notes in Comput. Sci. **1023**, Springer, Berlin, 1995.

[Pach and Tardos 2002] J. Pach and G. Tardos, "On the boundary complexity of the union of fat triangles", *SIAM J. Comput.* **31**:6 (2002), 1745–1760.

[Pach et al. 2003] J. Pach, I. Safruti, and M. Sharir, "The union of congruent cubes in three dimensions", *Discrete Comput. Geom.* **30**:1 (2003), 133–160.

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

[Paterson and Yao 1992] M. S. Paterson and F. F. Yao, "Optimal binary space partitions for orthogonal objects", *J. Algorithms* **13**:1 (1992), 99–113.

[Pauly et al. 2002] M. Pauly, M. H. Gross, and L. Kobbelt, "Efficient simplification of point-sampled surfaces", pp. 163–170 in *Proc. 13th IEEE Visualization Conference*, 2002.

[Preparata 1981] F. P. Preparata, "A new approach to planar point location", *SIAM J. Comput.* **10**:3 (1981), 473–482.

[Preparata and Shamos 1985] F. P. Preparata and M. I. Shamos, *Computational geometry*, Springer, New York, 1985.

[Schumacker et al. 1969] R. A. Schumacker, R. Brand, M. Gilliland, and W. Sharp, "Study for applying computer-generated images to visual simulation", Tech. Rep. AFHRL–TR–69–14, U.S. Air Force Human Resources Laboratory, 1969.

[Seidel 1991] R. Seidel, "A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons", *Comput. Geom.* **1**:1 (1991), 51–64.

[Shaffer and Garland 2001] E. Shaffer and M. Garland, "Efficient adaptive simplification of massive meshes", pp. 127–134 in *Proc. 12th IEEE Visualization Conference*, 2001.

[Thibault and Naylor 1987] W. C. Thibault and B. F. Naylor, "Set operations on polyhedra using binary space partitioning trees", *Comput. Graphics* **21**:4 (1987), 153–162.

[Tobola and Nechvile 2003] P. Tobola and K. Nechvile, "Linear binary space partitions and hierarchy of object classes", pp. 64–67 in *Proc. 15th Canadian Conf. Comput. Geom.* (Halifax, NS, 2003), 2003.

[Tóth 2003a] C. D. Tóth, "Binary space partition for orthogonal fat rectangles", pp. 494–505 in *Algorithms—ESA 2003*, Lecture Notes in Comput. Sci. **2832**, Springer, Berlin, 2003.

[Tóth 2003b] C. D. Tóth, "Binary space partitions for line segments with a limited number of directions", *SIAM J. Comput.* **32**:2 (2003), 307–325.

[Tóth 2003c] C. D. Tóth, "A note on binary plane partitions", *Discrete Comput. Geom.* **30**:1 (2003), 3–16.

Csaba D. Tóth
Department of Mathematics
Massachusetts Institute of Technology
77 Massachusetts Ave., Room 2-336
Cambridge, MA 02139
United States of America
toth@math.mit.edu