Balanced Aspect Ratio Trees: Combining the Advantages of k-d Trees and Octrees

CHRISTIAN A. DUNCAN*
Center for Geometric Computing
The Johns Hopkins University
duncan@jhu.edu

MICHAEL T. GOODRICH†
Center for Geometric Computing
The Johns Hopkins University
goodrich@jhu.edu

STEPHEN KOBOUROV‡
Center for Geometric Computing
The Johns Hopkins University
kobourov@cs.jhu.edu

Abstract

Given a set S of n points in $\mathbb{R}^d$, we show, for fixed d, how to construct in $O(n \log n)$ time a data structure we call the Balanced Aspect Ratio (BAR) tree. A BAR tree is a binary space partition tree on S that has $O(\log n)$ depth and in which every region is convex and "fat" (that is, has a bounded aspect ratio). While previous hierarchical data structures, such as k-d trees, quadtrees, octrees, fair-split trees, and balanced box decompositions can guarantee some of these properties, we know of no previous data structure that combines all of these properties simultaneously. The BAR tree data structure has numerous applications ranging from solving several geometric searching problems in fixed dimensional space to aiding in the visualization of graphs and three-dimensional worlds.

1 Introduction

Geometric searching of multi-dimensional spaces is a fundamental operation in many fields, including astronomy, geographic information systems (GIS), computer graphics, information retrieval, pattern recognition, natural language processing, and statistics. Typical searches include nearest-neighbor searches, farthest-neighbor searches, and range queries (which are intersection queries for geometric shapes). In this paper we study efficient data structures for performing such queries in moderate-dimensional spaces, that is, in spaces where the dimensionality, d, of the space can be viewed as a constant compared to the number, n, of multi-dimensional points in that space[1].

1.1 Previous Work. Data structures for performing multi-dimensional geometric searching in moderate-dimensional spaces is a well-studied problem in computational geometry and spatial databases. Many previous data structures for performing geometric searching for a multi-dimensional point set S are instances of a general class of structures known as binary space partition (BSP) trees [16] (see also [27]). Each node in a BSP tree T represents both a convex region in space and all of the points of S lying inside this region. Each leaf node in T represents a region with a constant number of points of S inside it. Every other node in T has an associated hyperplane cut partitioning the region into two subregions, each corresponding to a child node. The root of T is associated with a bounding hyperbox containing S. One of the main advantages of BSP trees is that they allow for simple multi-dimensional searching, with a typical comparison for a node v in T simply involving a sidedness test against the hyperplane cut associated with v. BSP trees are used often to solve problems in computer graphics [11, 23, 24, 28], such as global illumination, shadow generation, ray casting, and visibility. They are also used in information retrieval for finding nearest neighbors, farthest neighbors, and performing range queries.

The performance bounds of a BSP tree T for answering such queries for a point set S are directly related to the depth of T and the "fatness" [2, 14, 20] (that is, the boundedness of the aspect ratios) of the regions associated with T's vertices. One well-known class of BSP trees, known as k-d trees [6, 7, 17, 22], uses axis-parallel cutting hyperplanes that are placed so as to divide the set of points associated with a node more-or-less in half. Such trees have excellent depth properties, in that their depth is always $O(\log n)$. Unfortunately, since the objects in S are points, which are not themselves fat (as with the sets of objects studied in [1, 12, 21]), the regions associated with vertices in a k-d tree can have arbitrarily large aspect

*This research partially supported by ARO under grant DAAH04-96-1-0013.
†This research partially supported by NSF under Grant CCR-9625289, and ARO under grant DAAH04-96-1-0013.
‡This research partially supported by ARO under grant DAAH04-96-1-0013.
[1]We will view d as a constant relative to n throughout this paper.

ratios. This unbounded aspect ratio property of k-d trees partly accounts for why there are few simple theoretical results better than the $O(n^{1-1/d})$ average running time, even for approximation versions.

Another well-known class of search structures, known as quadtrees [15, 25, 27] and octrees [3, 10, 18, 26], are based on the alternate approach of using axis-parallel hyperplanes to divide region volumes equally. These structures produce space partitioning trees with regions having good aspect ratios, but their depths can be quite large, which again results in poor worst-case search times.

These poor worst-case performances of k-d trees and octrees have motivated some researchers to abandon the BSP tree framework altogether in search of alternate structures with good depth and aspect-ratio bounds. In particular, the balanced box-decomposition (BBD) tree structure of Arya et al. [5, 4], which is based on the fair-split tree of Callahan and Kosaraju [8, 9], provides a space partitioning tree that has $O(\log n)$ depth while also achieving vertex-associated regions with bounded aspect ratios. Arya et al. show that this structure can be used, for example, to perform approximate nearest-neighbor searching and range searching in $O(\log n + k)$ time, where k is the size of the output. The only drawback with this approach is that it partitions space using non-hyperplanar cuts with "holes," which sacrifices the convexity property for vertex-associated regions. This sacrifice makes the BBD tree inappropriate for several applications in computer graphics and graph drawing, where convexity of the partitioned regions is desirable (e.g., see [19]).

1.2 Our Results. Building on earlier work of the authors for a two-dimensional balanced space partition data structure [13], useful in the context of graph drawing, we introduce in this paper a multi-dimensional space partition tree data structure, which we call the *balanced aspect ratio (BAR) tree*, that is defined for any set S of n points in $\mathbb{R}^d$. The BAR tree data structure is conceptually quite simple, for it follows the traditional format of the binary space partition (BSP) tree. Moreover, BAR trees simultaneously achieve the desired properties of having $O(\log n)$ depth and vertex-associated regions that are convex and "fat" (having bounded aspect ratios).

In the following sections, we give a general framework for BAR trees, some geometric searching applications that are realized by BAR trees, and finally give an efficient method for constructing a BAR tree. Our method for constructing a BAR tree for a point set S uses hyperplane cuts that are either parallel to a coordinate axis or form 45° angles with coordinate axes, the latter of which we call "corner cuts." This method can be viewed as an extension of the traditional k-d tree, for our BAR tree is identical to the k-d tree defined on S so long as the k-d tree maintains a balanced aspect ratio; we only use corner cuts when an axis-parallel cut would produce a region that is too "skinny."

2 The General BAR Tree Framework

In this section, we develop a general framework for constructing BAR trees. We begin by defining what we mean by bounded aspect ratio.

DEFINITION 2.1. A convex region R in $\mathbb{R}^d$ has aspect ratio $\mathtt{asp}(R) = O_R/I_R$ with respect to some underlying metric, where O_R is the radius of the smallest circumscribed hypersphere in $\mathbb{R}^d$ and I_R is the radius of the largest inscribed d-hypersphere. R has *balanced aspect ratio* with maximum aspect ratio $\alpha \geq 1$, if $\mathtt{asp}(R) \leq \alpha$. We say, a region R is an *α-balanced region* if it has maximum aspect ratio α. A collection of regions, $\mathcal{R}$, has balanced aspect ratio with balancing factor α if each region $R \in \mathcal{R}$ is an α-balanced region.

Typically, we use one of the standard L_p metrics to define aspect ratios, as these are all within a polynomial factor of d from each other. In keeping with current custom [2, 14, 20], we use the terms *fat* and *skinny* to refer to regions which have respectively balanced and unbalanced aspect ratios. This definition states that a region R has a minimum width associated with its diameter. As in Arya et al. [5], we use this property to show worst-case bounds on some geometric approximation problems.

DEFINITION 2.2. A *canonical cut set*, $\mathcal{C} = \{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_\gamma\}$, is a collection of γ, not necessarily independent, vectors that span $\mathbb{R}^d$ (thus, $\gamma \geq d$). A *canonical cut* is any hyperplane, H, in $\mathbb{R}^d$ with a normal in $\mathcal{C}$. A *canonical region* is a convex polyhedron in $\mathbb{R}^d$ with every facet having a normal in $\mathcal{C}$.

DEFINITION 2.3. Any two canonical cuts H and H' that are normal to the same vector in $\mathcal{C}$, i.e. parallel to each other, are *opposing canonical cuts.* For any bounded region R, define the *canonical bounding cuts with respect to a direction $\vec{v}_i \in \mathcal{C}$* to be the two unique opposing canonical cuts normal to $\vec{v}_i$, and tangent to R. Intuitively, R is "sandwiched" between the two opposing cuts.

The canonical set used to define a partition tree can vary from method to method. For example, the standard k-d tree algorithm uses a canonical set of all axis-parallel directions. For notation, we often refer to a canonical cut by its normal vector, $\vec{v}_i \in \mathcal{C}$. In the k-d tree model, we would represent a cut along the y-axis by $(0, 1, 0, \ldots, 0)$. We refer to such an opposing pair by its direction, $\vec{v}_i \in \mathcal{C}$, in positive or negative form of the normal vector. Also, let $|R|$ represent the number of points from a given data set S contained in the region R, i.e. its size in terms of points rather than volume.

DEFINITION 2.4. An α-balanced canonical region, R, is *one-cuttable* with reduction factor β, where $1/2 \leq \beta < 1$, if there is a cut $s_1 \in \mathcal{C}$, called a *one-cut*, dividing R into two subregions R_1 and R_2 such that

1. R_1 and R_2 are α-balanced canonical regions,
2. $|R_1| \leq \beta|R|$ and $|R_2| \leq \beta|R|$.

DEFINITION 2.5. An α-balanced canonical region, R, is *k-cuttable* with reduction factor β, for $k > 1$, if there is a cut $s_k \in \mathcal{C}$, called a *k-cut*, dividing R into two subregions R_1 and R_2 such that

1. R_1 and R_2 are α-balanced canonical regions,
2. $|R_2| \leq \beta|R|$,
3. Either $|R_1| \leq \beta|R|$ or R_1 is $(k-1)$-cuttable with reduction factor β.

In other words, the sequence of cuts, $s_k, s_{k-1}, \ldots, s_1$, results in $k+1$ balanced canonical regions each containing no more than βn points. If β is understood, we simply say R is k-cuttable.

THEOREM 2.1. For a canonical set, $\mathcal{C}$, if every possible α-balanced canonical region is k-cuttable with reduction factor β, then a BAR tree with maximum aspect ratio α can be constructed with depth $O(k \log_{1/\beta} n)$, for any set S of n points.

Proof. Start with an initial bounding α-balanced canonical region on S. Since this region is k-cuttable, use a sequence of k cuts to divide the region into $k+1$ α-balanced canonical subregions each containing fewer than βn of the points. Repeat this process on each of the resulting subregions until a subregion has less than some constant number of points. The process, down any path of subregions, can be repeated for no more than $O(\log_{1/\beta} n)$ times, resulting in the stated tree depth bound. ■

The main challenge in creating a specific instance of a BAR tree is in defining a canonical set $\mathcal{C}$ such that every possible α-balanced canonical region is k-cuttable with reduction factor β for reasonable choices of α, β, and k. But, before we do this, let us motivate it by describing a few applications for BAR trees.

3 BAR Tree Applications

Suppose we are given a point set S of n points in $\mathbb{R}^d$ using any one of the Minkowski L_p metrics. After constructing a BAR tree T on S, we are able to perform some useful queries. For any query point $q \in \mathbb{R}^d$, we are able to efficiently report both the approximate-nearest and approximate-farthest neighbors of q in S. If we are also given a radius r, we are able to efficiently return all points within a distance r from q plus possibly any points that are approximately near r, which is a form of approximate range searching. We will more formally describe some of these applications shortly.

Arya et al. [4, 5] propose a technique to solve the approximate nearest-neighbor and range query problems by constructing a balanced box-decomposition tree. Similar to our BAR trees, these trees maintain an α-balanced aspect ratio but only by introducing non-convex hole cuts. Their arguments and techniques for solving these query problems, however, are easily transferable to our data structure.

DEFINITION 3.1. For a set S of points in $\mathbb{R}^d$, a query point $q \in \mathbb{R}^d$, and $\epsilon > 0$, a point $p \in S$ is a $(1+\epsilon)$-*nearest neighbor of* q if $\delta(p,q) \leq (1+\epsilon)\delta(p^*,q)$, where p^* is the true nearest neighbor to q.

In other words, such a p is within a constant error factor of the true nearest neighbor. This definition can also be extended to report a sequence of s $(1+\epsilon)$-nearest neighbors. Rather than adapt all theorems presented by Arya et al. in this extended abstract, we instead prove another useful query operation, applicable to both their and our methods, and establish an important packing feature for BAR trees.

DEFINITION 3.2. For a set S of points in $\mathbb{R}^d$, a query point $q \in \mathbb{R}^d$, and $\epsilon > 0$, a point $p \in S$ is a $(1-\epsilon)$-*farthest neighbor of* q if $\delta(p,q) \geq \delta(p^*,q) - \epsilon D$, where p^* is the true farthest neighbor and D is the diameter of the point set.

We are using an absolute error bound rather than the standard relative error, $\delta(p,q) \geq (1-\epsilon)\delta(p^*,q)$, because the bound is tighter in every case. Imagine a point set that is tightly contained in the unit sphere, and a query point that is extremely, $100/\epsilon$ units, far from this sphere. Now, any point returned would be a $(1-\epsilon)$-farthest neighbor of S using a relative error bound. In our definition, a query point is the better of the absolute and relative distances, within a constant factor of ϵ. Since D is the diameter of the set, any query point must be at least half the diameter away from one of the points in the set, $\delta(p^*,q) \geq D/2$. Letting $\epsilon' = 2\epsilon$, we see that

$$\begin{aligned}\delta(p,q) &\geq \delta(p^*,q) - \epsilon D \geq \delta(p^*,q) - \epsilon 2\delta(p^*,q)\\ &= (1-2\epsilon)\delta(p^*,q) = (1-\epsilon')\delta(p^*,q).\end{aligned}$$

Hence, using an absolute error bound in the approximate farthest-neighbor query always gives a point whose distance is at least as far as the distance obtained using a relative error bound. In fact, one can extend this notion and our arguments to compensate for this problem in nearest-neighbor queries as well, i.e., when the query point is relatively far away from the entire data set.

We now, briefly, discuss the farthest neighbor approximation algorithm using a BAR tree. Given our query point q, we begin by finding a leaf node that is the farthest away from q. Here, a region's maximum distance from a point is considered, implying that, in theory, the node containing the point q might still also be the node that is the farthest from q. We next enumerate, via a priority queue, all leaf nodes in decreasing order of distance, i.e. farthest leaf nodes first. For every leaf node, we compute the distance between that node's data point and q and maintain the current farthest visited point. When the distance between q and the current farthest node is less than $\delta(p,q)+\epsilon D$, we can terminate the search, as $\delta(p^*,q) < \delta(p,q)+\epsilon D$.

A priority queue can be maintained in such a way that the running time is $O(\log n)$ times the number of leaf nodes visited. The key to the algorithm's success is that the number of leaf nodes visited can be limited by using a packing argument. Let us, therefore, describe both the priority queue technique and the packing argument needed to limit the number of leaf nodes visited.

3.1 Searching Farthest Nodes. Let us describe, in more detail, the searching technique which uses the priority queue. To avoid confusion between points in space and in the data set, we call a point p, a *data point* if $p \in S$, and a point r, a *real point* if $r \in \mathbb{R}^d$. For any node u and its associated region R, let $\delta(u,q) = \delta(R,q) = \max_{r\in R}\delta(r,q)$, i.e. the distance between q and a node u is the distance between q and the farthest real point from q in the region R.

Initially, a priority queue Q starts with the root node of T. Let p be the current farthest neighbor, initially set to q. At every stage, extract from Q the node, u, that is the farthest from q. If $\delta(u,q) < \delta(p,q)+\epsilon D$, we return p as the $(1+\epsilon)$-approximate farthest neighbor. If u is a leaf node, let $p' \in S$ be the node's associated data point, if any. If $\delta(p',q) > \delta(p,q)$, let $p \leftarrow p'$. Remove u from consideration, and continue with the next node in Q. If u is not a leaf, let u_1 and u_2 be u's children. Since $u = u_1 \bigcup u_2$, one of the two nodes must contain the real point that was farthest from q. Wlog, let this be u_1 and insert u_2 into the queue. Use u_1 as next "extracted" node from Q and continue.

LEMMA 3.1. Given a BAR tree T with depth $O(k\log_{1/\beta} n)$ and γ canonical vectors, for any query point q, a $(1-\epsilon)$-farthest neighbor to q can be found in $O(kl\gamma\log_{1/\beta} n)$ time, where l is the number of leaf nodes visited in our algorithm.

Proof. First we can see the correctness of our algorithm by looking at the leaf node u^* containing p^*. If u^* has been visited, our algorithm would set $p \leftarrow p^*$ and return the correct solution upon termination. If u^* has not yet been visited, implying $p \neq p^*$, we see that $\delta(p^*,q) \leq \delta(u^*,q) \leq \delta(u,q) \leq \delta(p,q)+\epsilon D$.

Now, let u be any node that is extracted from the queue. Since our algorithm does not do another extract operation until it reaches a leaf node which has depth $O(k\log_{1/\beta} n)$, we perform $O(k\log_{1/\beta} n)$ queue inserts per leaf node visited and one extract per leaf node. If we use a Fibonacci heap, although in practice this would not be necessary, insertions take $O(1)$ amortized time and extractions take $O(\log n)$, since the queue has size $O(kn)$. At each split, deciding which node is farther takes $O(\gamma)$ time as the node regions have $O(\gamma)$ complexity. Thus, if l is the number of leaf nodes visited, the algorithm terminates in $O(kl\gamma\log_{1/\beta} n)$ steps. ■

3.2 Packing Constraint. Thus, as in the nearest-neighbor algorithm of Arya et al., if we can limit the number of leaf nodes that we need to visit, the running time would be known. This is where a packing constraint comes in.

LEMMA 3.2. (**Packing Lemma**) Given a BAR tree with maximum aspect ratio α for a set S of data points in $\mathbb{R}^d$ and two size parameters $r, r' > 0$, using any Minkowski metric, L_p, there are $O((\alpha\sqrt{d})^d(r/r')^{d-1})$ leaf nodes which pierce any annulus with radii $r+r'$ and r.

Proof. Let l be a leaf node in the tree with associated region R that pierces the annulus, A. This means that the outer radius $O_R \geq r'$. Since the region is α-balanced, recall that the inner radius of R, $I_R \geq O_R/\alpha$. We then know that the volume of R, in any metric, is greater than the volume of the L_1 hypersphere, a diamond in the plane,

$$V_R \geq (2I_R/\sqrt{d})^d \geq (2O_R/(\alpha\sqrt{d}))^d \geq (2r'/(\alpha\sqrt{d}))^d,$$

and because the objects are convex, the intersection $B = A \bigcap R$ has volume $V_B \geq (2r'/(\alpha\sqrt{d}))^d$. Now, in any metric, the volume of any annulus, A, of radius

$r + r', r$ is less than or equal to the difference between the volume of the outer and inner boxes of length $2(r + r')$ and $2r$ respectively, the L_∞ metric. Thus, $V_A \le (2r + 2r')^d - (2r)^d = (2r)^d((1 + r'/r)^d - 1)$. Since the leaf nodes do not overlap, the number of leafs nodes, L, is larger than the ratio of the two volumes, $L \le V_A/V_B \le (\alpha\sqrt{d})^d((r/r' + 1)^d - (r/r')^d) = O((\alpha\sqrt{d})^d(r/r')^{d-1})$. ■

Our only other concern, then, is that some leaf nodes might not contain any points, as we never made this stipulation in our general framework, although this scenario is highly unlikely. This problem is averted by noticing that since the regions are all k-cuttable, there are at most $k-1$ regions in sequence that can be empty, as after k cuts two regions must both have less than a β fraction of the points. We then defer the costs for an empty leaf node to one of the non-empty leaf nodes in its sibling's region. As there is an actual split after every $k - 1$ cuts, we can ensure that no leaf node gets more than k of the deferred charges, so the running time increases by at most a factor of k.

THEOREM 3.1. Suppose we are given a BAR tree T with depth $O(k \log_{1/\beta} n)$, a balancing factor α, and γ canonical vectors, on a point set S with diameter D and n data points. For any query point q, a $(1 - \epsilon)$-farthest neighbor to q can be found in $O(k^2\gamma(\alpha\sqrt{d})^d(1/\epsilon)^{d-1} \log n)$ time.

Proof. For any leaf node visited, but the last, u, $\delta(u, q) > \delta(p, q) + \epsilon D$, as this was the halting condition. Also, every leaf node visited, u, which contained a point, p', had to have $\delta(p', q) \le \delta(p, q)$ by the fact that p was the farthest point found. Thus, similar to the method of Arya et al. [5], we know that every node containing a point visited by the algorithm must completely pierce the annulus of radii $\delta(p, q)$ and $\delta(p, q) + \epsilon D$, i.e. lie on both sides of this annulus. Since the diameter of the point set is S, every leaf node containing a point must also lie at least partly inside a ball of radius D. Any leaf node piercing the original annulus, therefore, also pierces an annulus with radii D and $D(1 + \epsilon)$. In applying the Packing Lemma (3.2), Lemma 3.1, and our compensation for empty leaf nodes, we get a running time of $O(k\gamma \log_{1/\beta} n * (\alpha\sqrt{d})^d(D/(\epsilon D))^{d-1} * k)$, and the stated bound follows. ■

4 Corner-Cut BAR Trees

We now show a specific instance of a BAR tree. Ideally, in constructing a BAR tree we would prefer to use only axis-parallel cuts, but this is not possible. If, for example, the vast majority of the points are concentrated at a particular corner of a hyperbox region, cutting along any face forces the next cut to either be too close to the opposing face or not divide a significant portion of the points in the region, resulting in an unbalanced depth. Our solution is to introduce a simple corner cut that yields enough freedom of direction to construct a BAR tree.

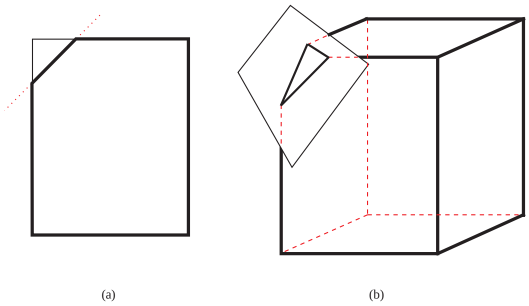

Figure 1: The notion of corner cuts. (a) A corner cut in the plane. (b) A corner cut in 3-d.

DEFINITION 4.1. A *corner cut* in $\mathbb{R}^d$ is a hyperplane whose normal vector is of the form $(I_0, I_1, \ldots, I_d)$ where $I_i \in \{1, -1\}$.

Notice that there are 2^d possible corner cuts corresponding to all combinations of ± 1. In the plane, such a cut forms a 45° angle with both the x- and y-axes (see Figure 1). The advantage to using these corner cuts will soon become apparent. Let us therefore show how to use such cuts to define a canonical cut set that is sufficient for constructing an efficient BAR tree.

Canonical Cut Set: Define a specific *canonical set*, $\mathcal{C}$, to be the set of all cuts which are either axis-parallel or corner cuts, in other words, all cuts whose normal is of the form $(0, 0, \ldots, 1, \ldots, 0)$ or $(I_0, I_1, \ldots, I_d)$ where $I_i \in \{1, -1\}$. Let $\mathcal{C}'$ be the set of axis-parallel cuts and $\mathcal{C}''$ be the set of corner cuts. So, $C = C' \bigcup C''$.

For a region R, canonical with respect to $\mathcal{C}$, consider a direction vector $\vec{v}_i \in \mathcal{C}$. Let $\texttt{width}_i(R)$ be the distance between the two bounding canonical cuts of R normal to $\vec{v}_i$. For simplicity, let us normalize the distance between two opposing planes by using the L_∞ metric. In a region R, this means that for two bounding canonical corner cuts p and q with normal $\vec{v}_i$ whose equations are of the form $p : \vec{v}_i\vec{x} = a$ and $q : \vec{v}_i\vec{x} = b$, the width $\texttt{width}_i(R) = \frac{|a-b|}{d}$ (see Figure 2.a).

Notice, since the normal vector to a corner cut forms a 45° angle with each axis, the regular Euclidean distance between the planes is $\frac{|a-b|}{d}\sqrt{d}$. The distances between two axis-parallel cuts in the L_p metrics are all identical, for $p \in \{1, \ldots, \infty\}$. Thus, using any other L_p

metric will only change our α parameter by $O(d)$.

DEFINITION 4.2. A canonical region R has *canonical aspect ratio*, $\texttt{casp}(R) = \frac{\max_{i \in \mathcal{C}}(\texttt{width}_i(R))}{\min_{j \in \mathcal{C}}(\texttt{width}_j(R))}$. R has *bounding box aspect ratio*, $\texttt{basp}(R) = \frac{\max_{i \in \mathcal{C}'}(\texttt{width}_i(R))}{\min_{j \in \mathcal{C}'}(\texttt{width}_j(R))}$.

In other words, the canonical aspect ratio is the ratio of the longest to smallest widths among the $2^{d-1} + d$ face pairs, and the bounded box aspect ratio is the aspect ratio of the bounding hyperbox of the region, ignoring the corner cuts. Because we are using the L_∞ metric, the maximum distance in the region R must be from a cut direction in $\mathcal{C}'$, implying that $\texttt{basp}(R) \leq \texttt{casp}(R)$. Also, for any region R, $\texttt{asp}(R) \leq \texttt{casp}(R)d$. It is acceptable, then, to use the canonical aspect ratio rather than the general aspect ratio in our construction of a BAR tree, with only a factor of d cost in aspect ratio performance with respect to the general framework. As a result, we call a canonical region R α-balanced with respect to the current canonical set if $\texttt{casp}(R) \leq \alpha$ for some $\alpha \geq 1$.

Even with these new cuts introduced into our algorithm, it is still difficult, in fact impossible, to guarantee a single cut that will divide the region into two proportionally equal regions. We do, however, prove that any balanced canonical region R is $(d+2)$-cuttable for sufficiently large values of α and β and from Lemma 2.1 we have a bound on our BAR tree depth. In the process of proving the existence of a $(d+2)$-cut, we also show a simple $O(n)$ time method for the discovery of such a cut.

4.1 Specific Properties. The use of corner cuts gives us certain very useful and important properties. First, during the process of making a corner cut on a region R, it is possible to create a subregion, R_1, such that some other bounding canonical cut, c, in R is no longer a bounding canonical cut in R_1, i.e. it is no longer tangential to the new subregion R_1. If this is the case, we simply take the bounding canonical cut of R_1 which is parallel to c to represent part of R_1. Therefore, when we refer to canonical cuts of a region, we are always referring to the region's *bounding* canonical cuts.

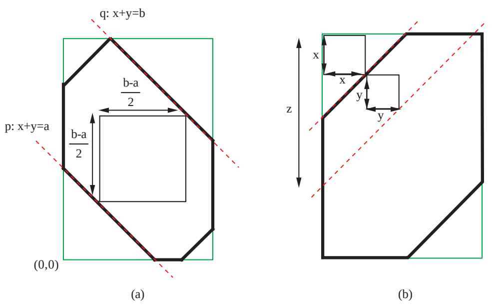

Figure 2: (a) Illustrates the distance between two corner cuts in L_∞ for the plane. (b) Illustrates the notion of a shielding cut. If x is the length of the corner cut, the shielding cut is at distance y from the corner cut, where $y \leq x/(\frac{\alpha}{d} - 1)$.

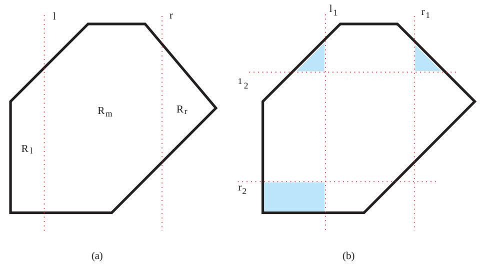

Figure 3: (a) Two cuts l and r which divide a region R into subregions R_l, R_m, and R_r. (b) Pairs of parallel cuts l_1, r_1 and l_2, r_2 divide region R into several subregions. The shaded areas are the intersections of the outer regions.

Every corner cut, besides having an opposing corner, also has d neighboring axis-parallel cuts, and d opposing axis-parallel cuts. For a corner cut, $c = (I_1, I_2, \ldots, I_d)$, let $c^j = (0, \ldots, I_j, \ldots, 0)$ for $j \in \{1, d\}$ be the j-th neighboring cut.

LEMMA 4.1. Suppose we are given a convex polyhedral region R, and two parallel hyperplanes l and r intersecting R (see Figure 3.a). We have three (possibly empty) subregions of R lying to the left, middle, and right of the two hyperplanes, respectively, R_l, R_m, and R_r. For any $1 \geq \beta \geq 1/2$, either there exists a hyperplane m parallel to l and r intersecting R_m which divides the region into two subregions both of which have no more than $\beta|R|$ points or no less than $\beta|R|$ points lies in either R_l or R_r.

Proof. Assume the contrary. Then, we know that $|R_l| < \beta|R|$ which implies that $|R_m| + |R_r| \geq \beta|R|$, or else the line l would be a suitable choice for m. Similarly, we know that $|R_l| + |R_m| \geq \beta|R|$. We now continually sweep a hyperplane m from l towards r, let R_1 and R_2 be the two subregions of R to the left and right of m. $|R_1| = |R_l| + x$ for $0 \leq x \leq |R_m|$. This implies that there exists $x \in \{0, \ldots, |R_m|\}$ such that $|R_1| = \beta|R|$ and $|R_2| = |R| - \beta|R| \leq \beta|R|$ for $\beta \geq 1/2$. Thus, there exists a hyperplane parallel to l and r which intersects R_m and divides R into two regions of size no more than $\beta|R|$, contradicting our original assumption. ■

In other words, there either is a cut that divides a region into two subregions each with less than a fraction of the original number of points or one of the two outside regions has more than this fraction of points. If this latter case holds, we call this region the *large outer region*, $\mathtt{lor}_{l,r}(R)$, where l and r are the two parallel hyperplanes. If there exists a dividing cut, m, then $\mathtt{lor}_{l,r}(R) = \emptyset$.

THEOREM 4.1. Suppose we are given a convex region R, and k pairs of parallel hyperplanes $(l_i, r_i), i \in \{1, \ldots, k\}$ (see Figure 2.b). For any $\beta \geq (k-1)/k$ (and $\beta \geq 1/2$), either there exists a hyperplane m parallel to one of the pairs which divides the region into two subregions of size less than $\beta|R|$ or $|P| \geq (1-(1-\beta)k)|R|$, where $P = \bigcap_{i=1}^{k} \mathtt{lor}_{l_i,r_i}(R)$.

Proof. Assume that there does not exist a hyperplane m dividing the region into two small subregions. By Lemma 4.1, every pair must have a maximal outer region for R or we would have a dividing hyperplane m. For each pair (l_i, r_i), let $P_i = \mathtt{lor}_{l_i,r_i}(R)$ be the large outer region of R for each pair of hyperplanes.

If $k = 1$, we know, from Lemma 4.1, $|P_1| \geq \beta|R| \geq (1-(1-\beta)k)|R|$. By induction, assume the theorem holds for all values less than k. Recall from set theory that for two sets A and B, $|A \bigcap B| = |A| + |B| - |A \bigcup B|$. Let $P' = \bigcap_{i=1}^{k-1} P_i$. We know that $|P' \bigcup P_k| \leq |R|$. Then, $P = P' \bigcap P_k \Rightarrow |P| \geq (1-(1-\beta)(k-1))|R| + \beta|R| - |R| = (1-(1-\beta)k)|R|$. Thus, it is true for k as well. ■

There are many corollaries that can be derived from this simple theorem. For example, if the intersection of all the large outer regions is empty and $\beta > (k-1)/k$, a hyperplane cut must exist that divides the region into two subregions of size less than $\beta|R|$ each as there is no other alternative location for the points to lie.

DEFINITION 4.3. Given an α-balanced canonical region R and a canonical cut c with normal $\vec{v}_i$, sweep a cut c' from the opposing cut towards c, calling the region of R between these two cuts P, either until P is empty or just before $\mathtt{casp}(P) > \alpha$. Notice, if P, is not empty, then P has maximum aspect ratio. Call the region P, the *shield region of c in R*, $\mathtt{shield}_c(R)$.

THEOREM 4.2. Given R, c, and c' as defined above, if $\mathtt{shield}_c(R) \neq \emptyset$ and we continue sweeping c' towards c, then the region between c and c' will have an unbalanced aspect ratio until it becomes empty.

Proof. Intuitively, if a convex region R starts out fat and in the process of cutting in a constant direction becomes skinny, continuing in that direction will never make it fat again. ■

COROLLARY 4.1. Given a canonical region R, a vector $\vec{v}_i \in \mathcal{C}$ and a reduction factor $1 > \beta \geq 1/2$, let b and c be the two opposing canonical cuts in R with normal $\vec{v}_i$. If $\mathtt{shield}_b(R) \bigcap \mathtt{shield}_c(R) = \emptyset$, then either a cut exists which divides R into two α-balanced region R_1 and R_2 each with less than $\beta|R|$ points or one of the shield regions has more than $\beta|R|$ points.

Proof. Since the two regions do not intersect, we have the two cuts defining each shield divide R into three regions, as in Lemma 4.1, the size constraints hold. More importantly, by the definition of a shield region (4.3) and the fact that the two shield regions do not intersect, any cut lying between the two regions will produce two α-balanced regions. ■

We refer to the shield region from the above corollary as the *large outer shield*, $\mathtt{los}_i(R)$, where $\mathtt{los}_i(R) = \emptyset$ if a dividing cut exists. Notice, the corollary corresponds to Lemma 4.1, with the added guarantee that the two regions produced from the cut chosen are α-balanced. Furthermore, this corollary may be extended to multiple hyperplane directions, i.e., extends Theorem 4.1.

DEFINITION 4.4. For a given canonical region R and a canonical corner cut c, the *length*, $\mathtt{len}_c(R)$, is defined as the distance from the cutting hyperplane c to the associated corner of the bounding hyperbox containing R, i.e., how far into the region does c cut, measured in the L_∞ metric.

Imagine any axis-parallel hyperbox. If we were to remove any corner and continue cutting inward, after traveling a certain length we would begin to shrink the bounding box of the remaining region, in at least one of the axis dimensions. After going even a little farther, we would begin to shrink the bounding box in every dimension simultaneously. The instant at which this happens is the largest the corner cut can possibly be while guaranteeing that the bounding hyperbox of the remaining region is now actually a hypercube. Because we exploit this property in proving certain regions are one-cuttable, we more formally state and prove this corollary next.

COROLLARY 4.2. For a given canonical region R and a canonical corner cut c, $\mathtt{len}_c(R) \leq \max_{i \in \mathcal{C}}(\mathtt{width}_i(R))(1 - 1/d)$. If equality holds, then $\mathtt{basp}(R) = 1$.

Proof. Wlog, assume that $c = (1, 1, \ldots, 1)$, i.e., the upper left hand corner, and the corresponding corner of the bounding hypercube for R be placed at the origin. This makes the equation of the hyperplane for c, $(1,1,\ldots,1)\vec{x} = \texttt{len}_c(R)d$.

Let $x = \max_{i\in\mathcal{C}}(\texttt{width}_i(R))$ be the maximum width between all of the bounding canonical cuts. This must then correspond to a particular pair of axis-parallel cuts. Wlog, assume that they are c^1, $(1,0,\ldots,0)\vec{x} = 0$, and its opposing cut b^1, $(1,0,\ldots,0)\vec{x} = x$, i.e. the left and right sides.

Now, assume that $\texttt{len}_c(R) > x(1-1/d) = \frac{x}{d}(d-1)$. Let the intersection of hyperplanes c and c^1 be the hyperplane c', $(0,1,1,\ldots,1)\vec{x} = \texttt{len}_c(R)d$. Notice that since $c \bigcap R$ and $c^1 \bigcap R$, $c' \bigcap R \neq \emptyset$. However, no point $p \in R$ can lie on c', as $(0,1,1,\ldots,1)p \leq x*(d-1) = \frac{x}{d}(d-1)(d) < \texttt{len}_c(R)d$. Thus, $c' \bigcap R = \emptyset$. $\Rightarrow\Leftarrow$

For the second statement, if $\texttt{len}_c(R) = x(1-1/d)$ and following a similar argument from above, we know that the distance from every axis-parallel side must also be at length x, or again no point in R can lie on the intersection of the two hyperplanes. ■

LEMMA 4.2. For an α-balanced canonical region R and a corner cut c,

$$\max_{i\in\mathcal{C}}(\texttt{width}_i(R)) \leq d(\texttt{len}_c(R) + \texttt{width}_c(R))$$

Proof. For simplicity, assume that the corner cut has normal vector, $\vec{v}_i = (1,1,\ldots,1)$ and the associated corner of the bounding hyperbox is at the origin. The hyperplane equation of the corner cut c is then $\vec{v}_i\vec{x} = d\texttt{len}_c(R)$ and the opposing bounding corner cut has equation $\vec{v}_i\vec{x} = d(\texttt{len}_c(R) + \texttt{width}_c(R))$. Thus, the distance between any two axis-parallel sides of the region and its associated bounding hyperbox must be no more than $d(\texttt{len}_c(R) + \texttt{width}_c(R))$. Since we are using the L_∞ metric, the maximum distance between any two bounding hyperplanes in R must correspond to one of the axis-parallel sides, i.e., a hyperplane on the bounding box, and the result follows (see Figure 4.a). ■

LEMMA 4.3. **(Corner-Cut Shield Lemma)** For an α-balanced canonical region R and a corner cut $c \in \mathcal{C}''$, $\texttt{width}_c(\texttt{shield}_c(R)) \leq \texttt{len}_c(R)/(\frac{\alpha}{d} - 1)$, for $\alpha > d$.

Proof. Let $x = \texttt{len}_c(R)$, $P = \texttt{shield}_c(R)$, $y = \texttt{width}_c(P)$, and $z = max_{i\in C}(\texttt{width}_i(P))$. We know from Lemma 4.2 that $z \leq (x+y)d$. Since P is the shield region for c, $\texttt{casp}(P) = \alpha$. Because we are using the L_∞ metric, y is the minimum width in P, implying that $z/y = \alpha \Rightarrow y = z/\alpha \leq (x+y)d/\alpha$. Solving for y, we get $y \leq xd/(\alpha - d) = x/(\frac{\alpha}{d} - 1)$ (see Figure 2.b). ■

This lemma is where the main advantage to using corner cuts comes in. If the corner cut is small, i.e., cuts very little of the region, we can make a cut in that direction very close to this corner cut, in proportion to its size and disregarding the dimensions of the rest of the region entirely.

4.2 k-Cut Existence Theorem. Before we prove that every balanced canonical region is $(d+2)$-*cuttable*, we first describe a few regions that are k-cuttable for $k \leq 3$.

THEOREM 4.3. For $\beta > (d-1)/d$ and $\alpha > 2d$, an α-balanced canonical region R is one-cuttable if there exists a corner cut $c \in \mathcal{C}''$ with opposing cut b such that

1. $\texttt{len}_b(R) = \max_{i\in\mathcal{C}}(\texttt{width}_i(R))(1 - 1/d)$,
2. $\texttt{len}_c(R) > \max_{i\in\mathcal{C}}(\texttt{width}_i(R))(1/\alpha)$.

Proof. Wlog, assume that $c = (1,1,\ldots,1)$. Let $x = \max_{i\in\mathcal{C}}(\texttt{width}_i(R))$. Recall from corollary 4.2, $\texttt{basp}(R) = 1$. Now, let us look at the various shield regions for each of the axis-parallel directions, $\vec{v}_i \in \mathcal{C}'$, of the neighboring face, b^i, of b. If we look at any hyperspace cut along this direction, the region formed towards this neighboring face will always have $\texttt{basp}(R) = 1$. This is because of the corner cut b which cuts every neighboring face simultaneously. Therefore, $P_i = \texttt{shield}_{b^i}(R) = \emptyset$. In the other direction, $Q_i = \texttt{shield}_{c^i}(R)$, has balanced aspect ratio α, this implies that $\texttt{width}_c(R) \leq x/\alpha$. We can do this for every one of the axis-parallel directions. We have d pairs of shield regions $(P_i, Q_i), \forall i \in (1,\ldots,d)$. From corollary 4.1, this means that for any $\beta > (d-1)/d$ either a one-cut exists or at least $(1-(1-\beta)d)|R| > 0$ points lies in one of the intersections of the regions. Let us determine which intersection of regions this could be. If it contained any of the P_i regions, the intersection would be empty as P_i is empty, notice the strictly greater than 0 condition.

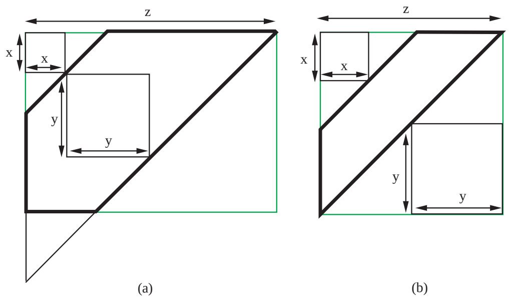

Figure 4: (a) The longest side of the bounding box of a given cannonical region has to be shorter than the sum of the length and distance of the corner cuts, i.e. $z \leq (x+y)d$. (b) An example of a basic one-cuttable region in the plane. Notice that $y = z/2$ and $x \geq z/\alpha$

Therefore, at least one point must lie in $I = \bigcap_{i=1}^{d} Q_i$. Now, let us look at $I \bigcap c$. Notice that $\texttt{len}_c(R) > x/\alpha$, yet in I, $\texttt{width}_i(I) <= x/\alpha, \forall i \in (1,\ldots,d)$. The intersection is empty and no points in R can lie in this intersection either. Consequently, there must be a one-cut on R. ■

THEOREM 4.4. For $\alpha \geq 3d$ and $\beta \geq d/(d+1)$, an α-balanced canonical region R is three-cuttable if $\texttt{basp}(R) \leq 2$.

Proof. Let $z = \max_{i \in \mathcal{C}'}(\texttt{width}_i(R))$, $y = \min_{i \in \mathcal{C}'}(\texttt{width}_i(R))$, and $n = |R|$. Since $\texttt{basp}(R) = z/y \leq 2$, $z/2 \leq y$. For each axis-parallel direction, $\vec{v}_i$, let us find the two shield regions, P_i and Q_i, associated with the opposing cuts. Wlog, let us look at P_i. Notice, that the $\texttt{width}_i(P_i) \leq z/\alpha \leq z/4 \leq y/2$ and the same for Q_i which implies that $P_i \bigcap Q_i = \emptyset$. Since $\beta \geq d/(d+1) > (d-1)/d$, from Corollary 4.1, either a dividing cut exists or $P = \bigcap_{i \in \mathcal{C}'} \texttt{los}_i(R) \neq \emptyset$. If a dividing cut exists, we are done, as the region is by definition three-cuttable. Otherwise, we know from our corollary that $|P| \geq (1-(1-\beta)d)n \geq (1-(1-d/(d+1))d)n = n/(d+1)$.

It is quite clear from our construction of P that P corresponds to one of the 2^d corners of the bounding hyperbox of R. Wlog, let this be the upper left corner, i.e. the corner whose associated cut is $\vec{v}_i = (1,1,\ldots,1)$. Now, since P is not empty, $\texttt{len}_c(R) \leq \texttt{width}_c(P) \leq x/\alpha$. Let $P_c = \texttt{shield}_c(R)$ and $P_b = \texttt{shield}_b(R)$, where b is the opposing cut of c. Because b can be at most the diagonal, $\texttt{len}_c(R) + \texttt{width}_c(R) \geq z/d$. Then, since $\texttt{len}_c(R) \leq z/\alpha$, $\texttt{width}_c(R) \geq z(1/d - 1/\alpha$. For $\alpha \geq 3d$, this means $\texttt{width}_c(R) \geq 2z/\alpha$ and a cut exists between b and c which creates two α-balanced regions. This means $P_b \bigcap P_c = \emptyset$. Also, from this, we know that $P_b \bigcap P = \emptyset$. Now again, either a dividing cut exists in R in which case we are again finished or at least βn of the points lies in either P_c or P_b. If $|P_b| \geq \beta n$, then $|R - P_b| \leq (1-\beta)n$. However, $|R - P_b| \geq |P| > n/(d+1) > (1-\beta)n$ which implies $|P_c| \geq \beta n$, but $|P_b| \geq \beta n$, a contradiction.

We claim that P_c is three-cuttable. First, observe that $|R - P_c| \leq (1-\beta)n \leq \beta n$; thus, if P_c is two-cuttable then by definition R is three-cuttable. By the Corner-Cut Shield Lemma 4.3, $\texttt{width}_c(P_c) \leq \texttt{len}_c(R)/(\alpha/d - 1) \leq \texttt{len}_c(R)/2 \Rightarrow z' = \max_{i \in \mathcal{C}}(\texttt{width}_i(P_c)) \leq d(\texttt{len}_c(P_c) + \texttt{width}_c(P_c)) \leq 3d\texttt{len}_c(P_c)/2$. Recall that $\texttt{len}_c(P_c) = \texttt{len}_c(R) \leq z/\alpha$ and $\alpha \geq 3d$. This means that $z' \leq 3dz/(2\alpha) \leq z/2$. Since $y \geq z/2$, one of two types of regions may be formed. We know that either the bounding box of P_c has minimum width z', in which case P_c is one-cuttable, or P_c is nearly identical to the region in Theorem 4.3 except that the corners may have additional corner cuts. However, with the simple addition of one extra shield cut in one of the d axis-parallel directions, it is not too difficult to see that we can produce a region which is one-cuttable, although we leave the details for the journal version. ■

THEOREM 4.5. For $\alpha \geq 3d$ and $\beta \geq d/(d+1)$, any α-balanced canonical region R is $(d+2)$-cuttable.

Proof. Let $z = \max_{i \in \mathcal{C}'}(\texttt{width}_i(R))$ and $y = \min_{i \in \mathcal{C}'}(\texttt{width}_i(R))$. If $\texttt{basp}(R) \leq 2$, we are done as the region is three-cuttable. If not, then there exists at least one axis-parallel direction with width less than $z/2$ which means that $y < z/2 \Rightarrow z > 2y$. Let $i \in \mathcal{C}'$ be the direction corresponding to z, i.e., corresponding to the longest side, and b and c be the two opposing canonical cuts with normal $\vec{v}_i$. Since z is large, it is easy to see that $\texttt{shield}_b(R) \bigcap \texttt{shield}_c(R) \neq \emptyset$. Now look at the $\texttt{los}_i(R)$. If $\texttt{los}_i(R) = \emptyset$, then a dividing cut exists in R in direction $\vec{v}_i$ and R is one-cuttable. Note that in many cases, especially those applications that find k-d trees practical, $\texttt{los}_i(R)$ will be empty as there is usually a dividing cut along the longest direction that produces α-balanced subregions. Otherwise, wlog, assume $shield_c(R)$ is the large outer region. Now, rather than cut at this shield region, we use a cut, c', in the direction of $\vec{v}_i$ to create a region R' where $\texttt{width}_c(R) = y$. Notice that $R \supset R' \supset \texttt{shield}_c(R)$, i.e., R' is larger than the shield region, and if R' is k-cuttable, then R is $(k+1)$-cuttable. Also, since there are at most $d-1$ sides with width greater than y and after every cut we reduce this count by one, after no more than $d-1$ cuts we will have an α-balanced canonical region with $\texttt{basp}(R) \leq 2$, which is three-cuttable. Therefore, any α-balanced canonical region is $(d+2)$-cuttable. ■

Notice that our scheme generally cuts along the longest axis-parallel side. We can alter this to include searching for one-cuts in the other axis-parallel directions and in so doing mimic the performance of k-d trees. In practice, as k-d trees often appear to perform quite well, the special two-cuttable regions may never have to be used. However, we have the added safety net of using these corner cuts, and in fact even finding appropriate one-cuts and two-cuts are quite easy although their existence is difficult to prove.

THEOREM 4.6. For the given canonical set, $\mathcal{C}$, a BAR tree with depth $O(d^2 \log n)$ and balancing factor α can be constructed in $O(d^2 \gamma n \log n)$ time, where γ is the size of the canonical set, here $O(2^d)$. This is $O(n \log n)$ for fixed dimensions.

Proof. Notice that at any stage, using even the most naive search, the large outer regions of a region R can be found in $O(|R|)$ time, and the shielding regions at each stage can easily be found in $O(\gamma)$ time. Thus, we can find any k-cut of a region R in $O(|R|\gamma)$ time. Since the depth is bounded, we have the running time as $O(k\gamma n \log_{1/\beta} n)$ time. Noticing that $k = O(d)$ and $\beta = O(d/(d+1)) \Rightarrow \log_{1/\beta} n = O(d \log n)$, we get the above stated running time. ■

5 Conclusion and Open Problems

In this paper, we introduced the general framework of the BAR tree and described some important applications that may be solved by using this type of tree. We also showed that when the dimensionality is fixed, an (α, β)-BAR tree can be constructed in $O(n \log n)$ time, where n is the number of points in the data set.

These results, however, are only preliminary. There are still many open problems for this new type of data structure. For example, the complexity of the canonical regions ($O(2^d)$), the number of cuts needed to ensure β-balance ($O(d)$), and the maximum α-balance factor ($O(d)$) are all far from optimal and could with more careful analysis be significantly dropped, perhaps by choosing a different canonical set entirely.

References

[1] P. K. Agarwal, E. F. Grove, T. M. Murali, and J. S. Vitter, *Binary space partitions for fat rectangles*, in Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci., Oct. 1996, pp. 482–491.

[2] P. K. Agarwal, M. Katz, and M. Sharir, *Computing depth orders for fat objects and related problems*, Comput. Geom. Theory Appl., 5 (1995), pp. 187–206.

[3] W. G. Aref and H. Samet, *Perspective viewing of objects represented by octrees*, Comput. Graph. Forum, 14 (1995), pp. 59–66. also University of Maryland Computer Science TR-2757.

[4] S. Arya and D. M. Mount, *Approximate range searching*, in Proc. 11th Annu. ACM Sympos. Comput. Geom., 1995, pp. 172–181.

[5] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu, *An optimal algorithm for approximate nearest neighbor searching*, in Proc. 5th ACM-SIAM Sympos. Discrete Algorithms, 1994, pp. 573–582.

[6] J. L. Bentley, *Multidimensional binary search trees used for associative searching*, Commun. ACM, 18 (1975), pp. 509–517.

[7] ———, *K-d trees for semidynamic point sets*, in Proc. 6th Annu. ACM Sympos. Comput. Geom., 1990, pp. 187–197.

[8] P. B. Callahan and S. R. Kosaraju, *Algorithms for dynamic closest-pair and n-body potential fields*, in Proc. 6th ACM-SIAM Sympos. Discrete Algorithms, 1995, pp. 263–272.

[9] P. B. Callahan and S. R. Kosaraju, *A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields*, J. ACM, 42 (1995), pp. 67–90.

[10] H. H. Chen and T. S. Huang, *A survey of construction and manipulation of octrees*, Comput. Vision Graph. Image Process., 43 (1988), pp. 409–431.

[11] N. Chin and S. Feiner, *Near real-time shadow generation using BSP trees*, in Proc. SIGGRAPH '89, New York, Aug. 1989, ACM SIGGRAPH, pp. 99–106.

[12] M. de Berg, *Linear size binary space partitions for fat objects*, in Proc. 3rd Annu. European Sympos. Algorithms, vol. 979 of Lecture Notes Comput. Sci., Springer-Verlag, 1995, pp. 252–263.

[13] C. A. Duncan, M. T. Goodrich, and S. G. Kobourov, *Balanced aspect ratio trees and their use for drawing very large graphs*, in Sixth Symposium on Graph Drawing, to be published in 1998.

[14] A. Efrat and M. Sharir, *On the complexity of the union of fat objects in the plane*, in Proc. 13th Annu. ACM Sympos. Comput. Geom., 1997, pp. 104–112.

[15] R. A. Finkel and J. L. Bentley, *Quad trees: a data structure for retrieval on composite keys*, Acta Inform., 4 (1974), pp. 1–9.

[16] H. Fuchs, Z. M. Kedem, and B. Naylor, *On visible surface generation by a priori tree structures*, Comput. Graph., 14 (1980), pp. 124–133. Proc. SIGGRAPH '80.

[17] A. Henrich, *Improving the performance of multi-dimensional access structures based on kd-trees*, in Proc. 12th IEEE Intl. Conf. on Data Engineering, 1996, pp. 68–74.

[18] C. Jackins and S. L. Tanimoto, *Oct-trees and their use in representing 3-d objects*, Comput. Graph. Image Process., 14 (1980), pp. 249–270.

[19] M. J. Katz, *3-D vertical ray shooting and 2-D point enclosure, range searching, and arc shooting amidst convex fat objects*, Comput. Geom. Theory Appl. to appear.

[20] J. Matoušek, J. Pach, M. Sharir, S. Sifrony, and E. Welzl, *Fat triangles determine linearly many holes*, SIAM J. Comput., 23 (1994), pp. 154–169.

[21] M. H. Overmars and A. F. van der Stappen, *Range searching and point location among fat objects*, J. Algorithms, 21 (1996), pp. 629–656.

[22] M. H. Overmars and J. van Leeuwen, *Dynamic multi-dimensional data structures based on quad- and k-d trees*, Acta Inform., 17 (1982), pp. 267–285.

[23] 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.

[24] ———, *Optimal binary space partitions for orthogonal objects*, J. Algorithms, 13 (1992), pp. 99–113.

[25] H. Samet, *The quadtree and related hierarchical data structures*, ACM Comput. Surv., 16 (1984).

[26] ———, *An overview of quadtrees, octrees, and related hierarchical data structures*, in Theoretical Foundations of Computer Graphics and CAD, R. A. Earnshaw, ed., vol. 40 of NATO ASI Series F, Springer-Verlag, Berlin, West Germany, 1988, pp. 51–68.

[27] ———, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, Reading, MA, 1990.

[28] E. Torres, *Optimization of the binary space partition algorithm (BSP) for the visualization of dynamic scenes*, in Eurographics '90, North-Holland, 1990, pp. 507–518.