BOXTREE: A Hierarchical Representation for Surfaces in 3D

Gill Barequet	Dept. of Computer Science, Tel Aviv University, 69978 Tel Aviv, Israel
Bernard Chazelle	Dept. of Computer Science, Princeton University, Princeton, NJ 08544
Leonidas J. Guibas	Dept. of Computer Science, Stanford University, CA
Joseph S.B. Mitchell	Dept. of Applied Mathematics, SUNY Stony Brook, NY 11794
Ayellet Tal	Dept. of Applied Mathematics, The Weizmann Institute of Science, Israel

Abstract

We introduce the boxtree*, a versatile data structure for representing triangulated or meshed surfaces in 3D. A boxtree is a hierarchical structure of nested boxes that supports efficient ray tracing and collision detection. It is simple and robust, and requires minimal space. In situations where storage is at a premium, boxtrees are effective alternatives to octrees and BSP trees. They are also more flexible and efficient than R-trees, and nearly as simple to implement.*

Keywords: collision detection, hierarchical data structures, ray shooting.

1. Introduction

In 1981 Ballard[1] presented a simple data structure for representing digitized curves by means of nested strips. This work is an attempt to generalize his *strip tree* structure to the case of surfaces in 3D. As is well known, curves can seem quite tame when compared to surfaces. For example, collision detection in 3D is orders of magnitude more difficult than in 2D. Expectedly, generalizing a strip tree into a *boxtree* raises a number of thorny issues. For one thing, there are many ways to go about it, few of which can be dismissed out of hand as uncompetitive. Thus, to guide our experimental search through a plethora of candidate heuristics, we kept an additional motivation in mind. We explain: The Dobkin-Kirkpatrick hierarchy[2] is a compact representation of a convex polytope that supports logarithmic-time collision detection. The main limitation of the method is that it requires convexity. Commonly, 3D objects (say, in medical imaging) appear convex from a distance, but close examination reveals a locally crumpled structure which invalidates the use of the DK hierarchy.

The initial motivation of this work was to remedy this shortcoming: was there a way of redesigning the DK hierarchy so that it would not break in the absence of convexity but, instead, degrade gracefully as objects become "less and less" convex? Another problem with the DK hierarchy our work tries to overcome is its lack of robustness: for a convex polyhedron with a thousand vertices the hierarchy will inevitably generate extremely thin, and hence trouble-prone, tetrahedra,

The boxtree is our answer to all these problems. It is a binary tree whose leaves are in bijection with the facets of a triangulated or meshed surface. (In this paper we consider only piecewise-linear surfaces, but modulo the cost of representing surface patches within boxes, what we say applies to the general case as well.) With each leaf we associate a box that encloses the corresponding facet. In a similar manner, each internal node is associated with a box that encloses the boxes of its two children.

With a boxtree, ray shooting and collision detection become straightforward: Starting from the root of the tree, determine whether the box collides with the ray or the obstacle in question. If yes, explore both children and recurse. This is the same search procedure found in octrees,[3] BSP trees,[4] and R-trees.[5] Of course, to make such a hierarchical boxing strategy work for us, various design factors must be taken into account:

- What shape of boxes should be chosen? We consider three kinds: (i) axis-parallel boxes, (ii) arbitrarily oriented rectangular boxes, and (iii) triangular cylinders (ie, pie slices).
- What merge rule should be followed, ie, how should sibling pairs be selected? We consider various alternatives

depending on several geometric criteria. Intuitively, sibling preference should be given to pairs of boxes that are small, incident to each other, and not too skewed.
- Given a pair of sibling boxes how do we form the enclosing parent box? We consider several optimization criteria involving volumes, areas, angles, and aspect ratios. We provide empirical data to shed light on our design choices.

Let us take a closer look at ray-shooting. Given an internal node v with children (u_1, u_2), if it is determined that the ray hits box(v), then both child boxes, box(u_1) and box(u_2), must be examined. However, only those actually hit by the ray trigger a recursive call. It follows immediately that the number of nodes traversed coincides precisely with the number of boxes hit by the ray. The maximum such number, over all rays, is called the *stabbing number* of the boxtree. It is asymptotically the worst-case cost of ray-shooting. Obviously, to keep this number low must be the chief goal of any design strategy.

A finely triangulated low-curvature surface looks locally like a planar subdivision, and so it is important to understand the case of terrains or "flat" surfaces. The latter are the subject of Section 2. The case of a rectangular meshed surface is particularly interesting and lends itself to nice mathematical analysis. As a byproduct, it also leads to a strikingly simple optimal point location structure for rectilinear subdivisions. We will also see in Section 3 that it bears direct relevance to the design of boxtrees with axis-parallel boxes.

Section 3 describes in detail the various heuristics we tried out to converge towards the best possible boxtree design. In Section 4 we report on experimental results and provide evidence to back our choice of the most promising design. Section 5 provides theoretical lower bounds on the performance of boxtrees.

Related Work. We are motivated by the need to do efficient ray shooting and collision detection among polyhedral objects in three dimensions.[6, 7, 8] The Dobkin-Kirkpatrick hierarchy[2] supports logarithmic intersection queries for a point, line, or plane against a single preprocessed convex polytope. For more general polyhedral surfaces and collections of triangles, recent theoretical results allow one to obtain $O(\log n)$ query time, with roughly $O(n^4)$ space, or sublinear query time with $O(n)$ space.[9, 10, 11]

The boxtree is perhaps most closely related to spatial access data structures that are popularly used in practice, but do not necessarily have good theoretical performance bounds. The strip tree data structure[1] is commonly used for representing polygonal curves hierarchically, by arranging subcurves into a tree, with each one approximated by a smallest enclosing rectangle. Another closely related data structure is the *R-tree* and its variants (which include the *packed* R-tree,[12] R$^+$-tree,[13] the R*-tree,[14] *optimized* R-trees,[15] *prism* trees,[16] and *cone* trees[17, 3]). An R-tree is an organization of geometric objects into a tree, with each node associated with a subset of the objects, for which we keep a minimal amount of geometric information stored at the node — we simply store the axis-aligned bounding box of the subset of objects. A leaf node contains some small constant number of primitive objects (e.g., triangular facets). The root node corresponds to the full set of objects. The children of an internal node v represent a partitioning of the objects associated with v. There are various heuristics that are used in determining the goodness of a partition, such as minimizing the total volume ("coverage"), total amount of overlap, or total surface areas of the resulting bounding boxes of the children. R-trees are constructed and maintained dynamically through simple insertion and deletion routines.

A big advantage of both the boxtree and the R-tree (as opposed to *spatial* partitioning methods, such as BSP-trees and octrees) is that the storage space is only linear, since its hierarchy is based on partitioning the set of *objects*. In fact, Held et al.[18] have compared four methods for doing collision detection for a flying object among a set of obstacles, and have found that a method based on a variant of R-trees uses significantly less space than methods based on tetrahedral decomposition, simple grid hashing, and k-d trees, and was often also the method having the fastest query processing. With boxtrees we allow more general classes of "bounding boxes" than with R-trees. In related work, prism trees[16] use truncated tetrahedra to bound objects, and bounding convex polyhedra based on a small fixed set of normal directions has been utilized in generalized bounding box methods.[19]

Our construction of boxtrees can be contrasted with insertion methods[14, 5, 15] and "top-down" methods.[18] We proceed in a "bottom-up" fashion, starting at the leaves and combining pairs of "neighboring" boxes. (Packed R-trees[12] are also built bottom-up, by grouping boxes according to a nearest neighbor criterion.) Zachmann and Felger [20] also use the term 'BoxTree". However, their data structure is substantially different from ours and is based on a partitioning of an object's bounding box, in a manner similar to k-d trees.

Finally, we should mention some other recent related work on interference detection. Hubbard[21] uses *sphere*

trees to arrange bounding spheres in a hierarchical fashion. Lin et al.[22, 23] have a system (I-COLLIDE) based on incrementally maintaining closest distances between pairs of convex polyhedra while pruning the number of pairs that need to be checked.

2. Hierarchical Boxing in the Plane

Suppose we are given a collection $\mathcal{R}$ of n axis-parallel rectangles that partition the unit square. Can one design a 2D boxtree of linear size and logarithmic stabbing number? In this case, boxes are to be understood as axis-parallel rectangles. We answer this question in the affirmative. In the process we also provide a very simple linear-preprocessing data structure for logarithmic-time point location in a rectilinear planar subdivision. The advantage of this method over Kirkpatrick's point location method[24] is that it only utilizes horizontal and vertical lines, which greatly facilitates the computations.

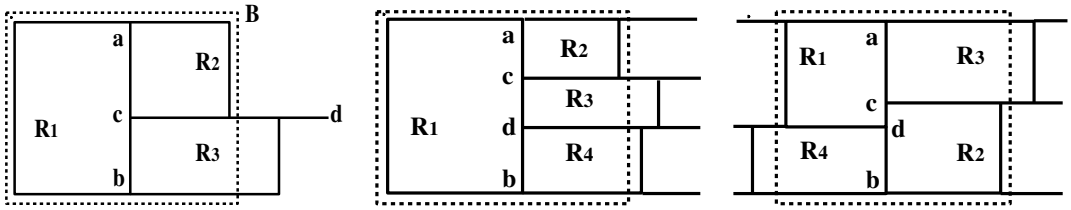
Let S denote the union of all rectangle boundaries. Let $G = (V, E)$ be the graph whose nodes are the maximal (horizontal/vertical) line segments contained in S, with an edge joining node u with node v if the endpoint of one segment (say, corresponding to u) lies on the other segment (the one corresponding to node v); ie, $(u, v) \in E$ if and only if the corresponding edges intersect. (We can assume that each endpoint of a maximal segment lies at a "T" junction: We can either disallow "cross" junctions by perturbing the input or slightly modifying the definition of "maximal" to consider segments of maximal length that do not contain a cross junction.) It is easy to check that $|V| = n + 3$. The graph G is planar, with every node of degree at least two. It is also *bipartite*, since nodes corresponding to horizontal segments can be colored "red", while the remaining nodes are colored "blue". Thus, $|E| \leq 2|V| - 4$, which shows that the average node degree is less than 4. A slightly more careful analysis gives the following result:

Lemma 2.1 Consider a bipartite planar graph in which every node has degree at least two. Then, there exists an independent set of at least $N/6$ (e.g., "blue") nodes, each having degree 4 or less, where N is the total number of nodes in the graph.

Proof Assume that there are N_b "blue" nodes and N_r "red" nodes. Without loss of generality, assume that $N_b \geq N_r$, so that $N_b \geq N/2$. Let D_b denote the sum of the degrees of all blue nodes. Since the graph is planar and bipartite, it has at most $2N - 4$ edges. Since each edge is incident exactly once on a blue node, we know that D_b is equal to the number of edges, and hence that $D_b \leq 2N - 4$.
Now, we claim that at least $1/3$ of the blue nodes have degree 4 or less. Assume to the contrary that fewer than $N_b/3$ blue nodes have degree 4 or less; then, more than $(2/3)N_b$ blue nodes have degree 5 or more. Consequently, since every node has degree at least two, the total blue degree, D_b, is greater than $5(2/3)N_b + 2(1/3)N_b = 4N_b$, thus contradicting the fact that $D_b \leq 2N - 4$. It follows that at least $1/3$ of the blue nodes have degree 4 or less. Thus, at least $N/6$ nodes have degree 4 or less. □

Theorem 2.2 A rectangular partition $\mathcal{R}$ must contain at least $(n+3)/6$ pairs of adjacent rectangles, R_1 and R_2, such that the region $B \setminus (R_1 \cup R_2)$, where B denotes the bounding box of R_1 and R_2, is covered by at most two other rectangles of $\mathcal{R}$. Furthermore, each of these rectangles has at least one side collinear with a side of B.

Proof The bipartite planar graph G has $n + 3$ nodes, each with degree at least 2. By the previous lemma, G must contain at least $(n + 3)/6$ nodes of degree at most 4, all of the same color (say, corresponding to vertical sides). We claim that each such node gives rise to a desired "good" pair of rectangles. Clearly, a node of degree 2 is good — we get two rectangles that share a common edge. Similarly, a node of degree 3 is good since it gives rise to the situation depicted in Figure 1(a): R_1 can be paired with R_2 or R_3, whichever has the least horizontal extent.
Finally, for a node of degree 4, corresponding to maximal segment ab, we get the two cases depicted in Figure 1(b), depending on whether the two (horizontal) segments incident on ab at the internal points c and d both extend to the same side of ab (as in case (1) of the figure), or extend on opposite sides of ab (as in case (2) of the figure). In both cases, it is easy to check that there is a pair, R_i and R_j, with the claimed property. For example, in case (1), one could pair R_1 with whichever R_k $(k > 1)$ has the least horizontal extent. In case (2), the rectangle to the right of ab with the least horizontal extent can be paired with the one to the left of ab with the least horizontal extent. □
Of course, the $(n+3)/6$ pairs of rectangles guaranteed in the theorem need not be "independent" in the sense that two pairs may involve the same rectangle. But, since a single rectangle can be involved in at most 2 pairs



(a) Node of degree 3 (b) Node of degree 4

Figure 1: *The configuration corresponding to a node of degree 3 or 4*

(corresponding to opposite sides of it), we know that we can pick a subset of at least half of the pairs, so that no two pairs share a common rectangle. Thus, we are assured of at least $(n+3)/12$ such pairs. Note that it is elementary to find all these pairs in linear time, assuming any standard plane graph representation.
We build a 2D boxtree bottom up as follows: Initially, we associate each rectangle with a distinct leaf of the tree. Next, we pick a maximal set of disjoint rectangle pairs among those provided by the lemma. For each pair we create a parent node for the two corresponding leaves: the box associated with the new node is the smallest enclosing axis-parallel rectangle enclosing the pair in question.
We now replace the chosen pairs by their enclosing boxes and iterate on this process recursively so as to finish off the construction of the boxtree. The one problem with this scheme is that boxes will now overlap, and technically the lemma no longer applies. This is a minor difficulty, however, because the overlapping occurs in a nicely controlled fashion: indeed, each overlapped box only needs to be "shortened" in one direction (horizontal or vertical) in order to make it nonoverlapping. So, we apply this shortening wherever needed in order to produce a new rectangular partition, at which point we are able to apply the pairing process again. (The pairing is done with respect to the shortened boxes; however, whenever we replace a pair by its enclosing box, we use the original (not shortened) boxes.) This forms the inductive invariant of the recursion.
Because at least a fixed fraction of the root nodes are paired at each iteration, this produces a boxtree of logarithmic height. Since at each level (prior to shortening) at most two rectangles can overlap at any given point, the stabbing number is at most twice the height. In this case, note that the stabbing number is defined as the maximum number of nodes whose corresponding box encloses a given point. The entire construction takes linear time, so this leads to a simple optimal point location structure for rectilinear subdivisions. □
It is possible to prove a stronger statement regarding the existence of mergeable rectangles (proof omitted).

Theorem 2.3 The collection of rectangles $\mathcal{R}$ always contains a pair of adjacent rectangles, R_1 and R_2, such that the set $B \setminus (R_1 \cup R_2)$ is either empty (meaning that an edge of R_1 exactly coincides with an edge of R_2) or is covered by a single rectangle, $R_3 \in \mathcal{R}$, with R_3 sharing part of one of its sides with a side of B, the bounding box of R_1 and R_2.

It is interesting to mention at this point that if we allow arbitrarily oriented subdivisions, all these nice schemes collapse. Actually, we will prove the following lower bound in Section 5.

Theorem 2.4 There exists a planar subdivision that admits of no boxtree with sublinear stabbing number.

3. Boxtrees in 3D

As we mentioned earlier, three main considerations must guide the choice of a heuristic: (i) box shape, (ii) merge rule, (iii) design of parent box. We describe each of them below and discuss implementation issues. Recall the basic principle underlying the construction of a boxtree. Given a triangulated (or meshed) surface, associate each facet with a leaf of a (non-necessarily balanced) binary tree T. For each internal node v with children u_1, u_2, we build a node-box such that $\text{box}(v) \supseteq \text{box}(u_1) \cup \text{box}(u_2)$. Next we discuss various box shapes and their design implications.

3.1. Axis-Parallel Boxes

The choice of axis-parallel parallepipeds for boxes has the advantage of simplicity. (It is a bottom-up version of an R-tree.) To test for collision detection is particularly simple. Similarly, merging two boxes is straightforward. There is one obvious parent box rule: make $\text{box}(v)$ the smallest box that encloses $\text{box}(u_1) \cup \text{box}(u_2)$.

The computation requires only coordinate comparisons. Initially, each facet of the triangulated surface can be covered by a minimum-area rectangle. This might be fine in the case of meshed surfaces but it might produce excessive overlap in the triangulated case. An octree decomposition[3] of the surface might be applied in preprocessing to produce well-behaved boxes for the leaves of the boxtree. This involves refining the surface into small patches with disjoint rectangular projections. Note that this might significantly increase the complexity of the surface, and even violate the linear-storage feature of boxtrees. Often, though, the increase in size might be tolerable. As shown in Section 5, however, some arbitrarily bad examples can always be hand-crafted.

Finding the proper merge rule is less obvious. Clearly, only pairs of adjacent or overlapping boxes should be selected, but which ones? The previous section suggests an answer. If the surface is a terrain (ie, the graph of a piecewise-linear bivariate function) then its vertical projection is a plane graph. Furthermore, the boxes enclosing the facets project into rectangles which form a configuration similar to that of Section 2. We can then apply the pairing rule devised for that case. The choice of pairings and their ordering leads naturally to a boxtree in 3D: we lift each 2D node-box along the z-axis by fitting it within the smallest horizontal strip that encloses both child-boxes.

The main result of the previous section implies that if we consider only vertical rays the stabbing number of the boxtree is logarithmic. This might be good enough in practice, if the surface is flat enough and the rays are not too tangential to it. But in the worst case, it is conceivable that a ray hits up to a linear number of boxes. This is not necessarily so bad. Indeed, we prove a linear lower bound in Section 5. We exhibit a polyhedral surface for which *any* boxtree has a linear stabbing number. Of course, this is a worst-case lower bound. So, there is hope that by exploiting more flexible box shapes one can reduce the stabbing number in practice.

3.2. Arbitrarily Oriented Boxes

We keep parallepipeds as our box shapes, but we are now free to rotate them as we please. Smallest enclosing rectangles (turned into boxes of zero thickness) can be used to cover each facet of the triangulated surface. The rectangles are easily computed (in constant time). To choose the best merged pairs of boxes, we considered both surface-areas (or volumes) and slopes. In other words, we checked combinations of the minimum-area (volume) enclosing box and minimum-slope difference of the children. Our conclusion is that area (or volume) criteria are the most important factors. We will attempt some theoretical justification of this below.

Given a pair of boxes, their 16 vertices are treated as a 3D population whose covariance matrix is computed and diagonalized. Recall that if the points are $\{(a_i, b_i, c_i)\}$ (translated so that the center of mass is at the origin), then the covariance matrix is the canonical matrix derived from the quadratic form: $\sum_i (a_i x + b_i y + c_i z)^2$. The statistical principal components that describe the population are the eigendirections of this matrix. For example, the direction along which the variance of the population is the largest is the eigendirection that corresponds to the largest eigenvalue. We investigated three methods for determining the enclosing box, given the 16 vertices of a pair of child boxes:

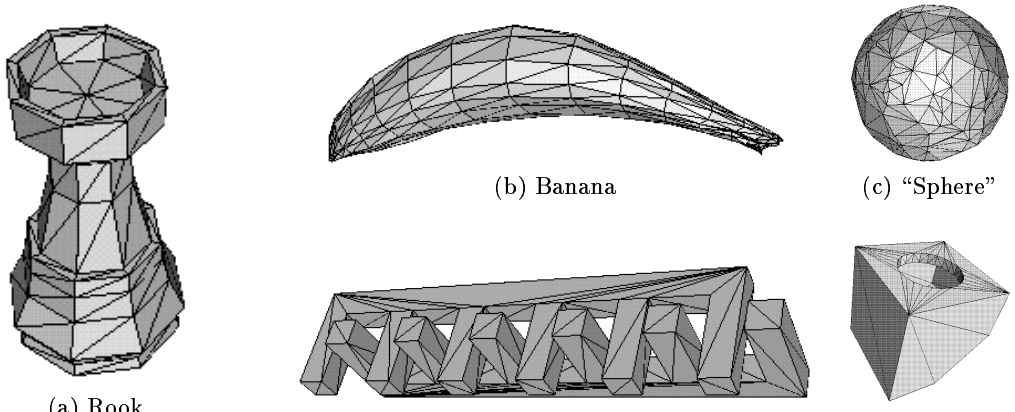
1. **All-Principal-Component Box:** We use the three principal components (ie, eigenvectors) as the three directions of the newly-built box. The box is centered at the center of mass of the 16 points.
2. **Max-Principal-Component Box:** We use the first principal component (the eigendirection that corresponds to the largest eigenvalue) as one direction. We then project the "population" (vertices of the input boxes) to a plane normal to that eigendirection. We find the minimum-area enclosing box in two dimensions. This box determines the other two directions of the resulting three-dimensional box.
3. **Min-Principal-Component Box:** We use the shortest principal component (corresponding to the smallest eigenvalue) as one direction of the newly-built box. We then proceed as in the previous method.

Plate 1 illustrates the construction of the min-principal-component box hierarchy. It shows the merge of two pairs of boxes into their parent boxes. It also depicts the lowest levels of the hierarchy.

3.3. Pie Slices

The "box" associated with a node is now a triangular pyramid (ie, an affinely translated Minkowski sum of a triangle and a segment normal to it). We call this a *pie slice*. Given the 12 vertices of a pair of pie slices (child boxes), we investigated two methods for determining the enclosing pie slice:

1. **Max-Principal-Component Pie Slice:** We use the first principal component as the direction of the pie slice. Then, we project the population of 12 vertices to a plane perpendicular to that direction. We find the enclosing isosceles triangle of minimum area. The Minkowski sum of this triangle, together with the principal component, determines the newly-built pie slice.



(a) Rook (b) Banana (c) "Sphere" (d) Two "combs" (e) CAD model

Figure 2: *Models that correspond to the data in Figures (3,4)*

2. **Min-Principal-Component Pie Slice:** We use the shortest principal component as the direction of the pie slice. We then proceed as before.

Plate 2 illustrates the construction of the min-principle-component pie slice hierarchy. It shows the merge of two pairs of slices into their parent slices. Also shown are the lowest levels of the hierarchy.

4. Experimental Results

We investigated six hierarchy methods and compared their performance: axis-aligned box, all-principal-component box, max-principal-component box, min-principal-component box, max-principal-component pie slice, and min-principal-component pie slice. Theoretically, area considerations[25] are the relevant measure in correlating the query processing time with the choice of a heuristic. This is due to the observation that the thickness of the enclosing boxes (or pie slices) increase only very slowly from level to level. This means that the overwhelming majority of node-boxes have roughly the same thickness. Empirically, volume considerations seem relevant as well.

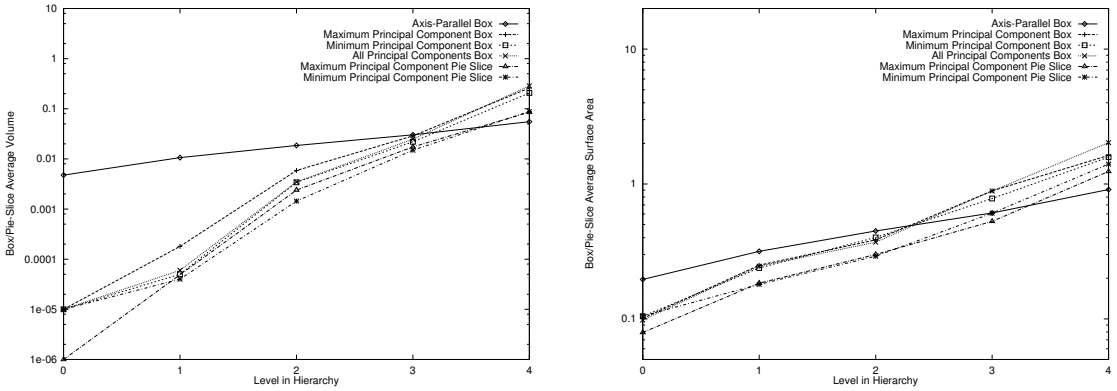
Now, in collision-detection applications where one is given a plane π (not a ray) and all boxes that π cuts must be found, it happens that surface area plays a major role in assessing the effectiveness of a heuristic. Indeed, suppose we define a *random* plane π as one that cuts—randomly and uniformly—a large box Ω enclosing the entire input surface. The probability measure used here is the unique plane density $d\omega$ invariant under motions. By Cauchy's integral-geometry theorem, the probability that a random plane cuts a given convex region is proportional to its surface area.

Let S_v be the surface area of the box (or pie slice) B_v at node v of the boxtree T. For simplicity, choose Ω to be B_r, where r is the root of the boxtree. Then, the expected number of nodes visited in a query determined by a random plane is equal to $\int \sum_{v \in T} \text{Prob}[\,\pi \text{ cuts } B_v\,]\, d\omega(\pi) = \frac{1}{S_r} \sum_{v \in T} S_v$. Thus, the average complexity of a query is minimized by simply choosing boxes that minimize the total surface area they occupy.

Figures 3 and 4 illustrate the average volume/surface-area per level of the enclosing objects (boxes or pie slices) for the six hierarchy methods that we investigated. The graphs correspond to the five input models shown in Figure 2. The rook contains 131 vertices and 256 facets. The banana contains 273 vertices and 512 facets. The sphere contains 556 facets connecting 280 vertices randomly distributed on a unit sphere. The comb contains 48 vertices and 92 facets. Finally, the CAD model contains 396 vertices and 132 facets.

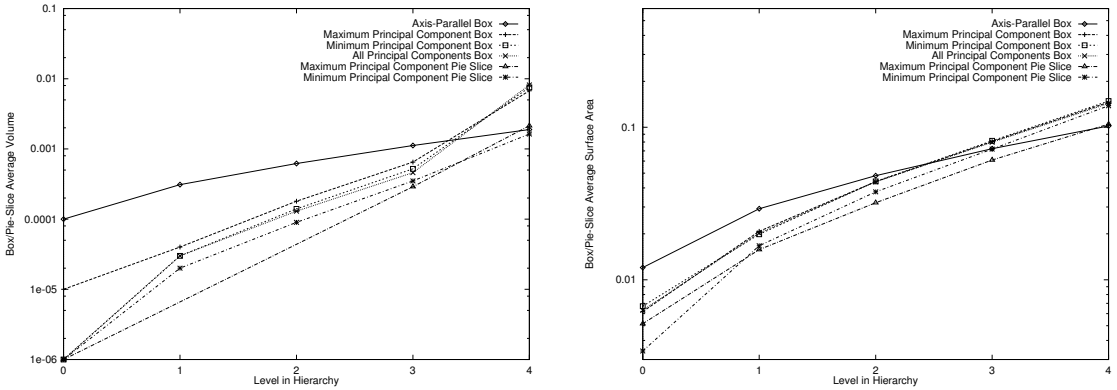
Pie slices have consistently smaller areas and volumes than boxes. The axis-parallel box method seems to be the worst in the lower levels of the hierarchy although in the upper levels it is the best. Since axis-parallel boxes tend to be smaller in the higher levels of the hierarchy, we chose to implement hybrid methods. In the lower levels we implemented each of the methods mentioned above while in the higher levels of the hierarchy we implemented the axis-aligned strategy in all cases. The graphs plot the lowest levels of the hierarchies.

Conclusion. Our goal is to minimize the number of hierarchy-objects whose collisions we attempt to detect. Area and volume are good measures since they indicate this number. When counting the number of collision tests between hierarchy elements, our experimentations show that the minimum-principal-component pie slice



Volume Surface area

(a) Rook



Volume Surface area

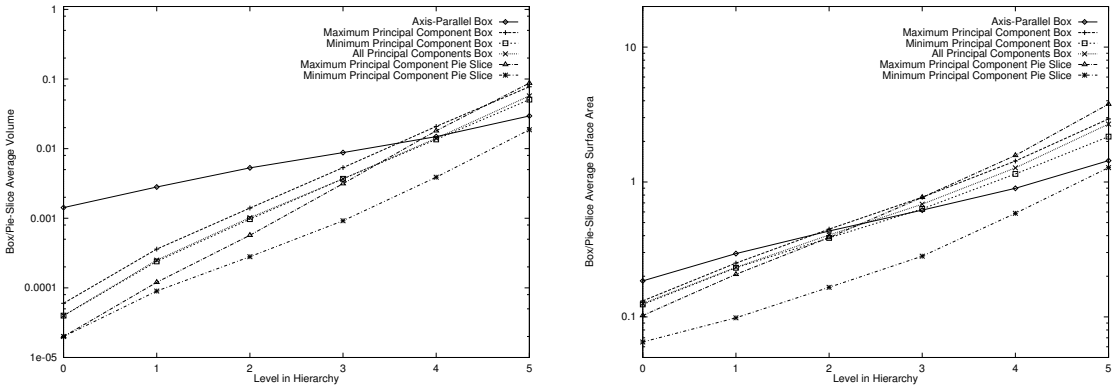
(b) Banana

Figure 3: *Average volume or surface area of enclosing objects*

heuristic is the method of choice. It is by far better than all other methods. The second-best method is the minimum-principal-component box heuristic. All other methods behave quite similarly. At first glance it might seem surprising that the three methods that rely on maximum-principal-component do not have an advantage over the axis-aligned boxing strategy. A close examination reveals the reason: the maximum principal component is the direction with the maximum variance of distribution of the vertices. Thus, this direction contributes the longest possible edge for the enclosing object.

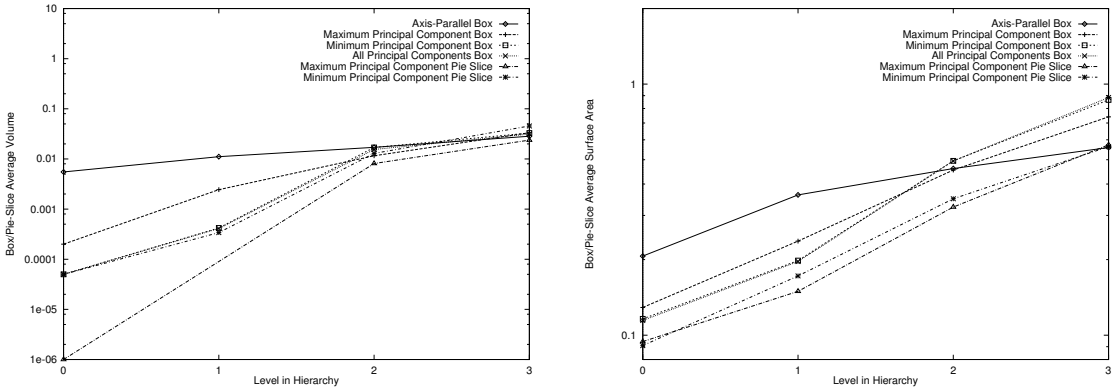
For example, we consider two convex objects, each with 1000 vertices and 1996 facets. With volume-directed hierarchies, the axis-aligned box strategy performed 35749 collision tests, the all-principal-component box strategy – 30039 tests, the maximum-principal-component box strategy – 33679, the minimum-principal-component box strategy – 22627, the maximum-principal-component pie slice strategy – 43757, and the minimum-principal-component pie slice – only 6179 tests.

Table 1 compares the number of collision tests performed by three methods: the best two methods (minimum-principal-component pie slice and box) and axis-aligned box. Table 1(a) shows the collision checks between two interleaved "combs" shown in Figure 2(d). The results of collision tests for three pairs of concentric sphere-models are given in Table 1(b,c,d). Each pair consists of two spheres whose surfaces are very close to each other. This construction was picked in order to enforce many collision tests between hierarchy objects. Each model in the first (resp. second, third) pair contains 280 (resp. 1000, 5000) vertices and 556 (resp. 1996, 9996) facets. As expected, as the input models grow in size, the minimum-principal-component strategies exhibit their superiority over the axis-aligned box strategy.



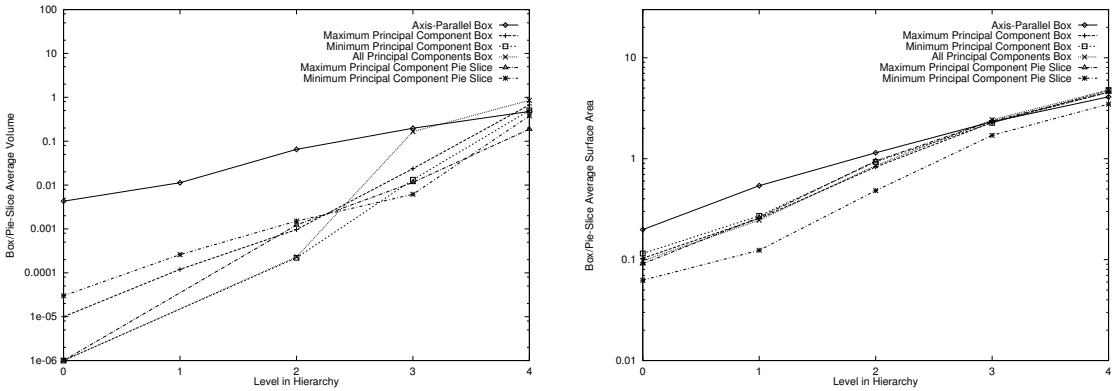
Volume Surface area

(c) "Sphere"



Volume Surface area

(d) "Comb"



Volume Surface area

(e) CAD model

Figure 4: *Average volume or surface area of enclosing objects (cont.)*

	Objects	# facets	Axis-aligned box	Min. prin. comp. box	Min. prin. comp. pie slice
(a)	combs	92	1,917	2,191	1,107
(b)	556-spheres	556	9,479	7,721	3,849
(c)	1996-spheres	1,996	35,749	22,627	6,179
(d)	9996-spheres	9,996	77,841	37,937	16,107

Table 1: *Number of collision tests*

5. Lower Bounds

We exhibit some surfaces that seem to doom the whole idea of a boxtree. In particular, we describe a convex polytope for whose boundary any boxtree has a linear stabbing number. This might seem disappointing in view of the fact that the DK hierarchy works wonders on convex polytopes. As we shall see, however, the lower bound construction is highly pathological. Indeed, empirical evidence seems to confirm that such bad cases are unlikely to turn up in practice. Our lower bound is to be compared to the linear lower bound[26] proven for the stabbing number of a triangulation of a polytope: our argument is completely different, however, which is to be expected since we use a convex polytope (for which logarithmic triangulation stabbing numbers are achievable).

We begin with a trivial linear lower bound on the stabbing number of axis-parallel boxtrees. Take the boundary of a cone whose apex is joined to the vertices of a regular $(n-1)$-polygon. It is a simple exercise to show that some point must belong to $\Omega(n)$ leaf-level boxes, hence the linear lower bound on the stabbing number. Note that the apex seems to always qualify as such a point: this is not the case, however, because just as in the DK hierarchy, the stabbing number must be defined as the maximum number of ray hits with respect to the *interior* of node-boxes. A more interesting scenario is to allow any convex box shape in defining a boxtree. The previous lower bound collapses immediately, and a more involved argument is needed. Our construction begins with a collection S of n segments in the plane. In a second stage, we lift the segments along the z-axis and take the convex hull of the resulting set.

Here are the details of the construction: $S = \{(p_k, q_k) : 0 \leq k < n\}$, where p_k is the point $(3^k, 9^k)$ and q_k is $(-3^n, -2 \times 3^{k+n} - 9^k)$. Note that the segment $p_k q_k$ is tangent to the parabola $y = x^2$. We easily verify that the convex hull of any two segments $p_j q_j$ and $p_k q_k$, $j < k$, contains the origin. To begin with, observe that the convex hull in question is the triangle $p_k q_j q_k$. The line passing through the segment $p_k q_k$ has for equation, $y = 2 \times 3^k x - 9^k$, so the origin lies right above the segment $p_k q_k$. It now suffices to show that it lies below the line through $p_k q_j$. This is equivalent to showing that $x(p_k)y(q_j) - y(p_k)x(q_j) > 0$. Replacing by the appropriate x and y coordinates, we find $-3^{k+j}(2 \times 3^n + 3^j) + 3^{2k+n} > 0$, which is always true. By completing the construction into a full-fledged triangulation (whichever way), we complete the proof of Theorem 2.4. □

Now, lift the segments onto the boundary of a convex 3D solid so that each segment is an edge of the convex hull. We omit the details. Obviously, there are at least $n-1$ node boxes that contain distinct pairs of (distinct) segments. Each such box is hit by the vertical line through the origin, which proves the desired lower bound. □

Theorem 5.1 There exists a convex polytope whose boundary admits no boxtree of sublinear stabbing number.

Acknowledgments

Work by G. Barequet has been supported by the Israeli Ministry of Science, Eshkol Grant 0562-1-94. Work by B. Chazelle has been supported in part by NSF Grant CCR-93-01254. Work by J. Mitchell has been supported in part by Hughes Research Laboratories, Boeing Computer Services, and NSF Grants CCR-9204585 and CCR-9504192. Work by A. Tal has been supported by the Israeli Ministry of Science, Eshkol Grant 0554-1-94.

References

1. D.H. Ballard, "Strip trees: A hierarchical representation for curves", Comm. of the ACM, **24** pp. 310–321 (1981).

2. D.P. Dobkin and D.G. Kirkpatrick, "Fast detection of polyhedral intersection", Theoret. Comput. Sci., **27** pp. 241–253 (1983).

3. H. Samet, *Spatial Data Structures: Quadtrees, Octrees, and Other Hierarchical Methods*, Addison-Wesley, Redding, Mass., 1989.

4. H. Fuchs, Z.M. Kedem, and B. Naylor, "On visible surface generation by a priori tree structures", Proc. SIGGRAPH '80, Computer Graphics, **14** pp. 124–133 (1980).

5. A. Guttman, "R-trees: A dynamic index structure for spatial searching", Proc. ACM SIGACT-SIGMOD Conf. Principles Database Systems, pp. 569–592 (1984).

6. T.L. Kay and J.T. Kajiya, "Ray tracing complex scenes", Proc. SIGGRAPH '86, Computer Graphics, **20** (4) pp. 269–278 (1986).

7. J. Goldsmith and J. Salmon, "Automatic creation of object hierarchies for ray tracing", IEEE Computer Graphics and Applications, **7** (5) pp. 14–20 (1987).

8. A.S. Glassner, *An Introduction to Ray Tracing*, Academic Press, London, 1989.

9. M. de Berg, *Efficient Algorithms for Ray Shooting and Hidden Surface Removal*, Ph.D. thesis, Dept. of Computer Science, Utrecht Univ., Utrecht, The Netherlands (1992).

10. M. de Berg, D. Halperin, M. Overmars, J. Snoeyink, and M. van Kreveld, "Efficient ray shooting and hidden surface removal", Proc. 7th Ann. ACM Symp. on Computational Geometry, pp. 21–30 (1991).

11. M. Pellegrini, "Stabbing and ray shooting in 3-dimensional space", Proc. 6th Ann. ACM Symp. on Computational Geometry, pp. 177–186 (1990).

12. N. Roussopoulos and D. Leifker, "Direct spatial search on pictorial databases using packed R-trees", Proc. ACM SIGACT-SIGMOD Conf. Principles Database Systems, pp. 17–31 (1985).

13. T. Sellis, N. Roussopoulos, and C. Faloutsos, "The R^+-tree: A dynamic index for multidimensional objects", Proc. 13th VLDB Conf., pp. 507–518 (1987).

14. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R^*-tree: An efficient and robust access method for points and rectangles", ACM SIGMOD Int. Conf. on Manag. of Data, pp. 322–331 (1990).

15. Y. Theodoridis and T. Sellis, "Optimization issues in R-tree construction", Technical Report KDBSLAB-TR-93-08, National Technical University of Athens, Athens, Greece (1993).

16. J. Ponce and O. Faugeras, "An object centered hierarchical representation for 3D objects: The prism tree", Computer Vision, Graphics, and Image Processing, **38** pp. 1–28 (1987).

17. Y.C. Chen, "An introduction to hierarchical probe model", Technical Report, Dept. of Mathematical Sciences, Purdue Univ. (1985).

18. M. Held, J.T. Klosowski, and J.S.B. Mitchell, "Evaluation of collision detection methods for virtual reality fly-throughs", Proc. 7th Canadian Conf. Computational Geometry, pp. 205–210 (1995).

19. M. Held, J.T. Klosowski, and J.S.B. Mitchell, "Speed comparison of generalized bounding box hierarchies", Technical report, Applied Math, SUNY Stony Brook (1995).

20. G. Zachmann and W. Felger, "The BoxTree: Enabling real-time and exact collision detection of arbitrary polyhedra", Proc. SIVE'95, pp. 104–113 (1995).

21. P.M. Hubbard, "Collision detection for interactive graphics applications", IEEE Trans. Visualization and Computer Graphics, **1** pp. 218–230 (1995).

22. J. Cohen, M. Lin, D. Manocha, and K. Ponamgi, "I-COLLIDE: An interactive and exact collision detection system for large-scaled environments", Proc. ACM Int. 3D Graphics Conference, pp. 189–196 (1995).

23. M.C. Lin, *Efficient Collision Detection for Animation and Robotics*, Ph.D. thesis, University of California, Berkeley, 1994.

24. D.G. Kirkpatrick, "Optimal search in planar subdivisions", SIAM J. Computing, **12** pp. 28–35 (1983).

25. H.C. van de Hulst, *Light Scattering by Small Particles*, Dover Publications, NY, 1981.

26. P.K. Agarwal, B. Aronov, and S. Suri, "Stabbing triangulations by lines in 3D", Proc. 11th Ann. ACM Symp. on Computational Geometry, pp. 267–276 (1995).