Eurographics Symposium on Virtual Environments (2006)
Roger Hubbold and Ming Lin (Editors)

A Model for the Expected Running Time of Collision Detection using AABBs Trees

René Weller[2] and Jan Klein[1] and Gabriel Zachmann[2]

[1] MeVis, Center for Medical Diagnostic Systems and Visualization, Bremen, Germany
[2] Department of Computer Science, Clausthal University, Germany

Abstract

In this paper, we propose a model to estimate the expected running time of hierarchical collision detection that utilizes AABB trees, which are a frequently used type of bounding volume (BV).
We show that the average running time for the simultaneous traversal of two binary AABB trees depends on two characteristic parameters: the overlap of the root BVs and the BV diminishing factor within the hierarchies. With this model, we show that the average running time is in $O(n)$ or even in $O(\log n)$ for realistic cases. Finally, we present some experiments that confirm our theoretical considerations.
We believe that our results are interesting not only from a theoretical point of view, but also for practical applications, e. g., in time-critical collision detection scenarios where our running time prediction could help to make the best use of CPU time available.

1. Introduction

Bounding volume hierarchies (BVHs) have proven to be a very efficient data structure for collision detection (CD), even for (reduced) deformable models [JP04].

The idea of BVHs is to partition the set of object primitives (e. g. polygons or points) recursively until some leaf criterion is met. In most cases, each leaf contains a single primitive, but the partitioning can also be stopped when a node contains less than a fixed number of primitives. Each node in the hierarchy is associated with a subset of the primitives and a BV that encloses this subset.

Given two BVHs, one for each object, virtually all CD approaches traverse the hierarchies simultaneously by an algorithm similar to Algorithm 1. It conceptually traverses a bounding volume test tree (BVTT; see Figure 2) until all overlapping pairs of BVs have been visited. It allows to quickly "zoom in" to areas of close proximity and stops if an intersection is found or if the traversal has visited all relevant sub-trees. Most differences between hierarchical CD algorithms lie in the type of BV, the overlap test, and the algorithm for constructing the BVH.

Figure 1: *Some models of our test suite: Infinity Triant (www.3dbarrel.com), lock (courtesy by BMW) and pipes.*

There are two conflicting constraints for choosing an appropriate BV. On the one hand, a BV-BV overlap test during the traversal should be done as fast as possible. On the other hand, BVs should enclose their subset of primitives as tight as possible so as to minimize the number of false positives with the BV-BV overlap tests. As a consequence, a wealth of BV types has been explored in the past, such as spheres [Hub96, PG95], OBBs [GLM96], DOPs [KHM*98, Zac98], Boxtrees [Zac02, AdBG*01], AABBs [vdB97, LAM01], spherical shells [KGL*98] and convex hulls [EL01].

In order to capture the characteristics of different approaches and to estimate the time required for a collision query, the cost function $T = N_v C_v + N_p C_p + N_u C_u + C_i$ was proposed [GLM96, KHM*98, He99], where

$N_v, C_v =$ num. and avg. costs of BV overlap tests, resp.
$N_p, C_p =$ num./avg. costs of primitive intersection tests
$N_u, C_u =$ num. and avg. costs of BV updates, resp.
$C_i =$ initialization costs

An example of a BV update is the transformation of the BV into a different coordinate system. During a simultaneous traversal of two BVHs, the same

traverse(A,B)
if A and B do not overlap **then**
  return
if A and B are leaves **then**
  return intersection of primitives enclosed by A and B
else
  for all children A_i and B_j **do**
    traverse(A_i,B_j)

Algorithm 1: Most hierarchical collision detection methods implement this algorithm to traverse two given BVHs.

BVs might be visited multiple times. However, if the BV updates are not saved, then $N_v = N_u$. This cost function was introduced by [WHG84] to analyze hierarchical methods for ray tracing and later adapted to hierarchical collision detection methods by [GLM96, KHM*98, He99].

In practice, N_v, the number of overlap tests, usually dominates the running time, i. e., $T(n) \sim N_v(n)$, because $N_p = \frac{1}{2}N_v$ in a binary tree and $N_u \leq N_v$. While it is obvious that $N_v = n^2$ in the worst case, it has long been noticed that, in practice, this number seems to be linear or even sub-linear.

However, until now there is no rigorous average-case analysis for the running time of simultaneous BVH traversals.

Therefore, the goal of this paper is to present a model, with which one can estimate the average number N_v, the number of overlap tests in the average case. Since this is, to our knowledge, the first attempt, we restrict ourselves to AABB trees. This allows to estimate the probability of an overlap of a pair of bounding boxes by simple geometric reasoning.

2. Related Work

In the last few years, some very interesting theoretical results on the collision detection problem have been shown. One of the first results was presented by Dobkin and Kirkpatrick [DK85]. They have shown that the distance of two convex polytopes can be determined in time $O(\log^2 n)$, where $n = \max\{|A|,|B|\}$, and $|A|$ and $|B|$ are the number of faces of object A and B, respectively.

For two general polytopes whose motion is restricted to fixed algebraic trajectories, [ST95] have shown that there is an $O(n^{\frac{5}{3}+\varepsilon})$ algorithm for rotational movements, and an $o(n^2)$ algorithm for a more flexible motion that still has to be along fixed, known trajectories [ST96].

[SHH98] proved that for n convex, well-shaped polytopes (with respect to aspect ratio and scale factor), all intersections can be computed in time $O((n+k)\log^2 n)$, where k is the number of intersecting object pairs. They have generalized their approach to first *average-shape* results in computational geometry [ZS99].

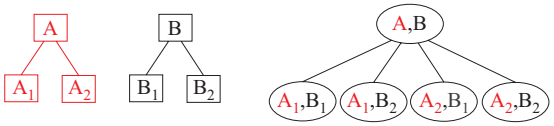

Figure 2: *The BV test tree (BVTT) shows all possible pairs of BVs that might need to be tested for overlap. All hierarchical CD algorithms, such as the one in Algorithm 1, basically perform a traversal through this (conceptual) tree.*

Under mild coherence assumptions, [VCC98] showed linear expected time complexity for the CD between n convex objects. They use well-known data structures, namely octrees and heaps, along with the concept of spatial coherence.

The Lin-Canny algorithm [LC91] is based on a closest-feature criterion and makes use of Voronoi regions. Let n be the total number of features, the expected run time is between $O(\sqrt{n})$ and $O(n)$ depending on the shape, if no special initialization is done.

In [KZ03], an average-case approach for CD was proposed. However, no analysis of the running time was given.

3. Analyzing Simultaneous Hierarchy Traversals

In this section, we will derive a model that allows to estimate the number N_v, the number of BV overlap tests. This is equivalent to the number of nodes in the BVTT (see Fig. 2) that are visited during the traversal. The order and, thus, the exact traversal algorithm are irrelevant.

For the most part of this section, we will deal with 2-dimensional BVHs, for sake of illustration. At the end, we extend these considerations to 3D, which is fairly trivial.

The general approach of our analysis is as follows. For a given level l of the BVTT, we estimate the probability of an overlap by recursively resolving it to similar probabilities on higher levels. This yields a producct of conditional probabilities. Then, we estimate the conditional probabilities by geometric reasoning.

Let $\tilde{N}_v^{(l)}$ be the expected number of nodes in the BVTT that are visited on level l. Clearly,

$$\tilde{N}_v^{(l)} = 4^l \cdot P[A^{(l)} \cap B^{(l)} \neq \varnothing] \qquad (1)$$

where $P[A^{(l)} \cap B^{(l)} \neq \varnothing]$ denotes the probability that any pair of boxes on level l overlaps. In order to render the text more readable, we will omit the "$\neq \varnothing$" part and just write $P[A^{(l)} \cap B^{(l)}]$ henceforth.

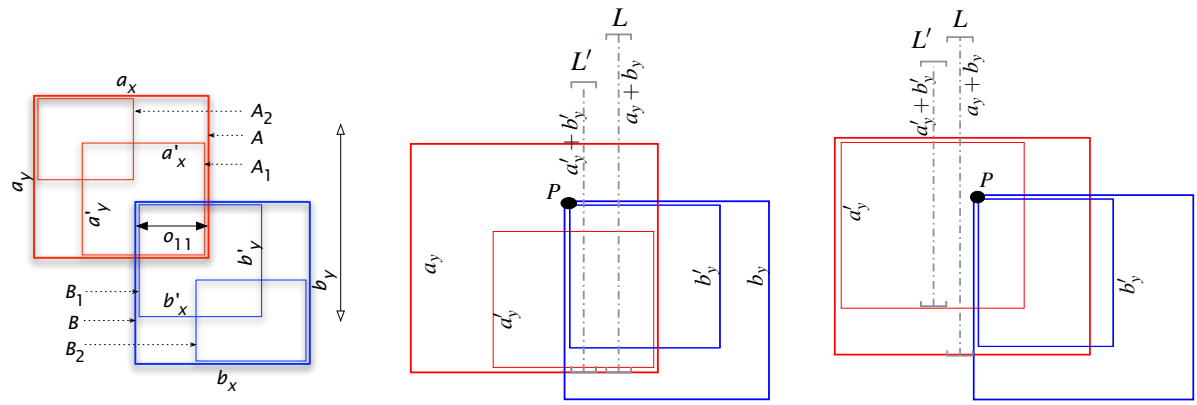

Figure 3: *Left: general configuration of the boxes, assumed throughout our probability derivations. For sake of clarity, boxes are* not *placed flush with each other. Middle: The ratio of the length of segments L and L' equals the probability of A_1 overlapping B_1. Right: ditto for p_{21}.*

Overall, the expected total number of nodes we visit in the BVTT is

$$\tilde{N}_v(n) = \sum_{l=1}^{d} \tilde{N}_v^{(l)} = \sum_{l=1}^{d} 4^l P[A^{(l)} \cap B^{(l)}] \tag{2}$$

where $d = \log_4(n^2) = \lg(n)$ is the depth of the BVTT (equaling the depth of the BVHs).

In order to derive a closed-form solution for $P[A^{(l)} \cap B^{(l)}]$, we recall the general equations for conditional probabilities:

$$P[X \wedge Y] = P[Y] \cdot P[X \,|\, Y] \tag{3}$$

and, in particular, if $X \subseteq Y$

$$P[X] = P[Y] \cdot P[X \,|\, Y] \tag{4}$$

where X and Y are arbitrary events (i. e., subsets) in the probability space.

Let $o_x^{(l)}$ denote the overlap of a given pair of bounding boxes when projected on the x-axis, which we call the *x-overlap*. Then,

$$P[A^{(l)} \cap B^{(l)}] = P[A^{(l)} \cap B^{(l)} \,|\, A^{(l-1)} \cap B^{(l-1)} \wedge o_x^{(l)} > 0] \cdot P[A^{(l-1)} \cap B^{(l-1)} \wedge o_x^{(l)} > 0]$$

by Eq. 4, and then, by Eq. 3,

$$P[A^{(l)} \cap B^{(l)}] = P[A^{(l)} \cap B^{(l)} \,|\, A^{(l-1)} \cap B^{(l-1)} \wedge o_x^{(l)} > 0] \cdot P[A^{(l-1)} \cap B^{(l-1)}] \cdot P[o_x^{(l)} > 0 \,|\, A^{(l-1)} \cap B^{(l-1)}]$$

Now we can recursively resolve $P[A^{(l-1)} \cap B^{(l-1)}]$, which yields

$$P[A^{(l)} \cap B^{(l)}] = \prod_{i=1}^{l} P[A^{(i)} \cap B^{(i)} \,|\, A^{(i-1)} \cap B^{(i-1)} \wedge o_x^{(i)} > 0] \cdot \prod_{i=1}^{l} P[o_x^{(i)} > 0 \,|\, A^{(i-1)} \cap B^{(i-1)}] \tag{5}$$

3.1. Preliminaries

Before proceeding with the derivation of our estimation, we will set forth some denotations and assumptions.

Let $A := A^{(l)}$ and $B := B^{(l)}$. In the following, we will, at least temporarily, need to distinguish several cases when computing the probabilities from Eq. 5, so we will denote the two child boxes of A and B by A_1,A_2 and B_1,B_2, resp.

For sake of simplification, we assume that the child boxes of each BV sit in opposite corners within their respective parent boxes.[†] Furthermore, without loss of generalization, we assume an arrangment of A, B, and their children according to Figure 3, so that A_1 and B_1 overlap before A_2 and B_1 do (if at all).

Finally, we assume that there is a constant *BV diminishing factor* throughout the hierarchy, i.e.,

$$a'_x = \alpha_x a_x, \qquad a'_y = \alpha_y a_y, \qquad \text{etc.}$$

Only for sake of clarity, we assume that the scale of the boxes is about the same, i. e.,

$$b_x = a_x, \qquad b'_x = a'_x, \qquad \text{etc.}$$

[†] According to our experience, this is a very mild assumption [Zac02].

This assumption allows us some nice simplifications in Equations 6 and 10, but it is not necessary at all.

3.2. Probability of Box Overlap

In this section, we will derive the probability that a given pair of child boxes overlaps under the condition that their parent boxes overlap.

Since we need to distinguish, for the moment, between 4 different cases, we define a shorthand for the four associated probabilities:

$$p_{ij} := P[A_i \cap B_j \mid A \cap B \wedge o_x > 0]$$

One of the parameters of our probability function is the distance $o_x^{(0)} := \delta$, by which the root box $B^{(0)}$ penetrates $A^{(0)}$ along the x axis from the right. Our analysis considers all arrangments as depicted in Figure 3, where δ is fixed but B is free to move vertically, under the condition that A and B overlap.

First, let us consider p_{11} (see Figure 3). By precondition, A overlaps B, so the point P (defined as the upper left (common) corner of B and B_1) must be on a certain vertical segment L that has the same x coordinate as the point P. Its length is $a_y + b_y$.[‡] Note that for sake of illustration, segment L has been shifted slightly to the right from its true position in Figure 3 (center). If, in addition, A_1 and B_1 overlap, then P must be on segment L'.

Thus,

$$p_{11} = \frac{\text{Length}(L')}{\text{Length}(L)} = \frac{a'_y + b'_y}{a_y + b_y} = \alpha_y. \tag{6}$$

Next, let us consider p_{21} (see Figure 3; for sake of clarity, we re-use some symbols, such as a'_x). For the moment, let us assume $o_{21,x} > 0$; in Section 3.3 we estimate the likelihood of that condition.

Analogously as above, P must be anywhere on segment L', so

$$p_{21} = \alpha_y = p_{11}$$

and, by symmetry, $p_{12} = p_{21}$. Very similarly, we get $p_{22} = \alpha_y$.

At this point, we have shown that $p_{ij} \equiv \alpha_y$ in our model.

3.3. Probability of X-Overlap

We can trivially bound

$$P[o_x^{(i)} > 0 \mid A^{(i-1)} \cap B^{(i-1)}] \leq 1$$

[‡] Actually, P can be chosen arbitrarily, under the condition that stays fixed on B as B assumes all possible positions. L would be shifted accordingly, but its length would be the same.

$\alpha_x \cdot \alpha_y$	$T(n)$
$< 1/4$	$O(1)$
$1/4$	$O(\lg n)$
$\sqrt{1/8} \approx 0.35$	$O(\sqrt{n})$
$3/4$	$O(n^{1.58})$
1	$O(n^2)$

Table 1: *Effect of the BV diminishing factor α_y on the running time of a simultaneous hierarchy traversal.*

Plugging this into Equation 2, and substituting that in Equation 5 yields

$$\tilde{N}_v(n) \leq \sum_{l=1}^{d} 4^l \cdot \alpha_y^l = \frac{(4\alpha_y)^{d+1} - 1}{4\alpha_y - 1} \qquad (4\alpha_y \neq 1)$$
$$\in O\big((4\alpha_y)^d\big) = O\big(n^{\lg(4\alpha_y)}\big). \tag{7}$$

The corresponding running time for different α_y can be found in Table 1. For $\alpha_y > 1/4$, the running time is in $O(n^c)$, $0 < c \leq 2$.

In order to derive a better estimate for $P[o_x^{(l)} > 0 \mid A^{(l-1)} \cap B^{(l-1)}]$, we observe that the geometric reasoning is exactly the same as in the previous section, except that we now consider all possible loci of point P when A and B are moved only along the x-axis. Therefore, we estimate

$$P[o_x^{(l)} > 0 \mid A^{(l-1)} \cap B^{(l-1)}] \approx \alpha_x. \tag{8}$$

Plugging this into Equations 2 and 5 yields an overall estimate

$$\tilde{N}_v(n) \leq \sum_{l=1}^{d} 4^l \cdot \alpha_x^l \cdot \alpha_y^l \in O\big(n^{\lg(4\alpha_x\alpha_y)}\big). \tag{9}$$

This results in a table very similar to Table 1.

3.4. The 3D Case

As mentioned above, our considerations can be extended to 3D straight-forwardly. In 3D, L and L' in Equation 6 are not line segments any longer, but 2D rectangles in 3D lying in the y/z plane. The area of L' can be determined by $(a'_y + b'_y)(a'_z + b'_z)$ and the area of L by $(a_y + b_y)(a_z + b_z)$. Thus,

$$p_{11} = \frac{\text{area}(L')}{\text{area}(L)} = \frac{(a'_y + b'_y)(a'_z + b'_z)}{(a_y + b_y)(a_z + b_z)} = \frac{4a'_y a'_z}{4a_y a_z} = \alpha_y \alpha_z. \tag{10}$$

The other probabilities p_{ij} can be determined analogously as above, so that $p_{11} = p_{12} = p_{21} = p_{22} = \alpha_y \alpha_z$.

Overall, we can estimate the number of BV overlap tests by

$$\tilde{N}_v(n) \leq \sum_{l=1}^{d} 4^l \cdot \alpha_x^l \cdot \alpha_y^l \cdot \alpha_z^l \in O\big(n^{\lg(4\alpha_x\alpha_y\alpha_z)}\big). \tag{11}$$

where $d = \log_4(n^2) = \lg(n)$.

Note that Table 1 is still valid in the 3D case.

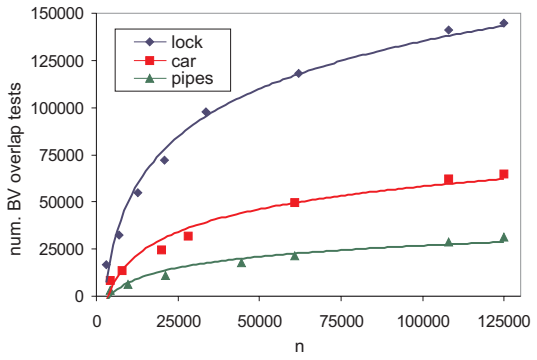

Figure 4: *The number of visited BVTT nodes for models shown in Fig. 1 at distance* $\delta = 0.4$.

4. Experimental Support

Intuitively, not only α should be a parameter of the model of the probabilities (Eqs. 6 and 8), but also the amount of penetration of the root boxes. This is not captured by our model, so in this section we present some experiments that provide a better feeling of how these two parameters affect the expected number of BV overlap tests.

We have implemented a version of Algorithm 1 using AABBs as BVs (in 3D, of course). As we are only interested in the number of visited nodes in the BVTT, we switched off the intersection tests at the leaf nodes.

For the first experiment, we used a set of CAD objects, each of them with varying numbers of polygons (Fig. 1). Fig. 4 shows the number of BV overlap tests for our models depending on their complexities for a fixed distance $\delta = 0.4$. Clearly, the average number of BV overlap tests behaves logarithmically for all our models.

For our second experiment, we used artificial BVHs where we can adjust the BV diminishing factors $\alpha_{x,y,z}$. As above, the child BVs of each BV are placed in opposite corners. In addition, we varyied the root BV penetration depth δ.

We plotted the results for different choices of α and n, averaged over the range 0.0–0.9 for δ (see Fig. 5). For larger α's, this seems to match our theoretical results. For smaller α, our model seems to underestimate the number of overlapping BVs. However, it seems, that the asymptotical running-time does not depend very much on the amount of overlap of the root BVs, δ (see Fig. 7).

5. Application to Time-Critical Collision Detection

As observed in [KZ03], almost all CD approaches use some variant of Algorithm 1, but often, there is no special order defined for the traversal of the hierarchy, which can be exploited to implement time-critical computing.

Our probability model suggests one way how to prioritize the traversal. for a given BVH, we can measure the average BV diminishing factor for each subtree and store this with the nodes. Then, during run-time, a good heuristic could be to traverse the subtrees with lower α-values first, because in these subtrees the expected number of BV pairs we have to check is asymptotically smaller than in the other subtrees.

In addition, we could tabulate the plots in Figure 7 (or fit a function), and thus compute a better expected number of BV overlaps during run-time of time-critical collision detection.

6. Conclusion and Future Work

We have presented an average-case analysis for simultaneous AABB tree traversals, under some assumptions about the AABB tree, that provides a better understanding of the performance of hierarchical collision detections, which has been observed in the past. Our analysis is independent of the order of the traversal.

In addition, we have performed several experiments to support the correctness of our model. Moreover, we have shown that the running time behaves logarithmically for real world models, even for a large overlap between the root BVs.

Several existing methods for hierarchical collision detection may benefit from our analysis and our model. Especially in time-critical environments or real-time applications it could be very helpful to predict the running-time of the collision detection process only with the help of two parameters that can be determined on-the-fly. We will try to speed up probabilistic collision detection by the heuristics mentioned in this paper.

We have already tried to derive a theoretical model of the probabilities that depends on the BV diminishing factor as well as the penetration distance of the two root BVs. This would, hopefully, lead to a probability density function describing the x-overlaps, thus yielding a better estimat of $\tilde{N}_v^{(l)}$. However, this challenge seems to be difficult.

Furthermore, a particular challenge will be a similar average-case analysis for BVHs utilizing other types of BVs, such as DOPs or OBBs. The geometric reasoning would probably have to be quite different from the one presented in this paper.

Finally, it would be very interesting to apply our technique to other areas, such as ray tracing. And, finally, we believe one could exploit these ideas to obtain better bounding volume hierarchies.

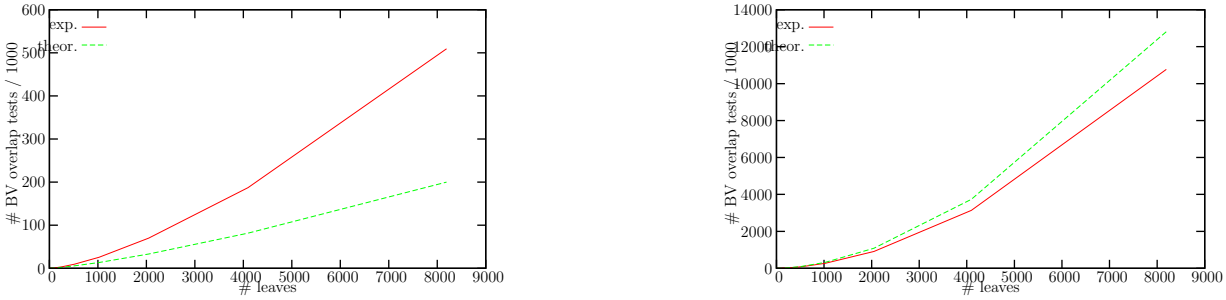

Figure 5: *For larger values of* α*, our theoretical model seems to match the experimental findings fairly well (left:* $\alpha = 0.7$*, right:* $\alpha = 0.9$*.*

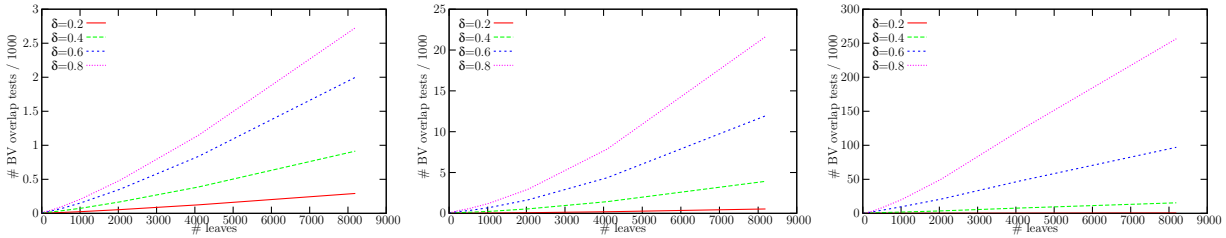

Figure 6: *The asymptotic number of overlapping BVs depends mainly on* α*, the BV diminishing factor, and only to a minor extent on* δ*, the penetration depth of the root BVs. The plots from left to right show* $\alpha = 0.6, 0.7, 0.8$*.*

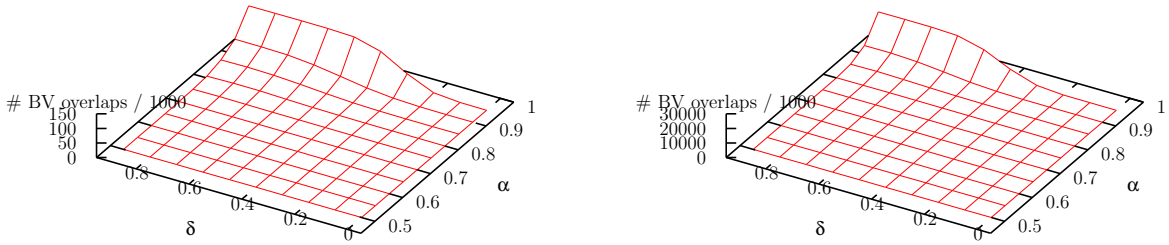

Figure 7: *For each number of leaves in the BVH, the distribution of overlapping BVs seems to be nearly the same. Left: Each BVH has* $n = 512$ *leaves, right: each has* $n = 8192$ *leaves.*

Acknowledgement

We would like to extend our thanks to one anonymous reviewer (you know who you are), who has asked us a friendly and seemingly artless question and thus, we believe, spurred a significant improvement of this work.

This work was partially supported by DFG grant ZA292/1-1.

References

[AdBG*01] Agarwal P. K., de Berg M., Gudmundsson J., Hammar M., Haverkort H. J.: Box-trees and r-trees with near-optimal query time. In *Proc. Seventeenth Annual Symposium on Computational Geometry (SCG 2001)* (2001), pp. 124–133.

[DK85] Dobkin D. P., Kirkpatrick D. G.: A linear algorithm for determining the separation of convex polyhedra. *J. Algorithms 6*, 3 (1985), 381–392.

[EL01] Ehmann S. A., Lin M. C.: Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum (Proc. of EUROGRAPHICS 2001) 20*, 3 (2001), 500–510.

[GLM96] Gottschalk S., Lin M., Manocha D.: OBBTree: A hierarchical structure for rapid interference detection. *ACM Transactions on Graphics (SIGGRAPH 1996) 15*, 3 (1996), 171–180.

[He99] He T.: Fast collision detection using quospo trees. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (1999), pp. 55–62.

[Hub96] Hubbard P. M.: Approximating polyhedra with

spheres for time-critical collision detection. *ACM Transactions on Graphics 15*, 3 (July 1996), 179–210.

[JP04] James D. L., Pai D. K.: BD-Tree: Output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3 (Aug. 2004), 393–398.

[KGL*98] Krishnan S., Gopi M., Lin M. C., Manocha D., Pattekar A.: Rapid and accurate contact determination between spline models using ShellTrees. *Computer Graphics Forum (Proc. of EUROGRAPHICS 1998) 17*, 3 (Sept. 1998), 315–326.

[KHM*98] Klosowski J. T., Held M., Mitchell J. S. B., Sowrizal H., Zikan K.: Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (Jan. 1998), 21–36.

[KZ03] Klein J., Zachmann G.: Time-critical collision detection using an average-case approach. In *Proc. ACM Symposium on Virtual Reality Software and Technology (VRST 2003)* (Osaka, Japan, 2003), pp. 22–31.

[LAM01] Larsson T., Akenine-Möller T.: Collision detection for continuously deforming bodies. In *Eurographics* (2001), pp. 325–333. short presentation.

[LC91] Lin M. C., Canny J. F.: A fast algorithm for incremental distance calculation. In *Proceedings of the IEEE International Conference on Robotics and Automation* (1991), pp. 1008–1014.

[PG95] Palmer I. J., Grimsdale R. L.: Collision detection for animation using sphere-trees. *Computer Graphics Forum 14*, 2 (June 1995), 105–116.

[SHH98] Suri S., Hubbard P. M., Hughes J. F.: Collision detection in aspect and scale bounded polyhedra. In *SODA* (1998), pp. 127–136.

[ST95] Schömer E., Thiel C.: Efficient collision detection for moving polyhedra. In *11th Annual Symposium on Computational Geometry* (June 1995), pp. 51–60.

[ST96] Schömer E., Thiel C.: Subquadratic algorithms for the general collision detection problem. In *12th European Workshop on Computational Geometry* (March 1996), pp. 95–101.

[VCC98] Vemuri B. C., Cao Y., Chen L.: Fast collision detection algorithms with applications to particle flow. *Computer Graphics Forum 17*, 2 (1998), 121–134.

[vdB97] van den Bergen G.: Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools 2*, 4 (1997), 1–14.

[WHG84] Weghorst H., Hooper G., Greenberg D. P.: Improved computational methods for ray tracing. *ACM Trans. Graph. 3*, 1 (1984), 52–69.

[Zac98] Zachmann G.: Rapid collision detection by dynamically aligned DOP-trees. In *Proc. of IEEE Virtual Reality Annual International Symposium (VRAIS 1998)* (Atlanta, Georgia, Mar. 1998), pp. 90–97.

[Zac02] Zachmann G.: Minimal hierarchical collision detection. In *Proc. ACM Symposium on Virtual Reality Software and Technology (VRST 2002)* (Hong Kong, China, Nov. 2002), pp. 121–128.

[ZS99] Zhou Y., Suri S.: Analysis of a bounding box heuristic for object intersection. *Journal of the ACM 46*, 6 (1999), 833–857.