

The Expected Running Time of Hierarchical Collision Detection

Technical Report

Jan Klein¹ and Gabriel Zachmann²

¹ Heinz Nixdorf Institute and Institute of Computer Science, University of Paderborn, Germany

² Institute of Computer Sciences, Clausthal University of Technology, Germany

Abstract

In this paper, we propose a model to estimate the expected running time of hierarchical collision detection, which is commonly based on bounding volume hierarchies (BVHs).

We show that the average running time for the simultaneous traversal of two binary BVHs 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.

Not only from a theoretical point of view our results are very useful, but they can particularly be used in practical applications, e.g., in time-critical collision detection scenarios where our running time prediction would help to determine an optimal distribution of available CPU time.

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

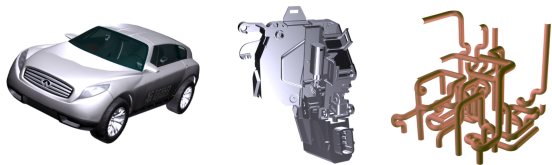


Figure 1: Some models of our test suite: Infinity Triant (www.3dbarrel.com), lock (courtesy by BMW) and pipes.

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.

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

```

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.

proposed [GLM96, KHM*98, He99], where

N_v, C_v = num. and avg. costs of BV overlap tests, resp.
 N_p, C_p = num. and 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 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 an asymptotic analysis, N_v , the number of overlap tests, defines 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. In this paper, we present a model based on a geometrically motivated estimation of the probability of an overlap of BV pairs, which are the nodes in the BVTT.

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 $O(n^{\frac{5}{3}+\epsilon})$ 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

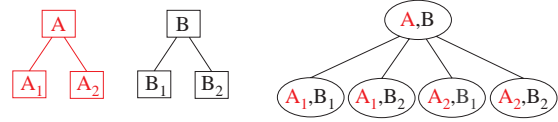


Figure 2: The BV test tree (BVTT) shows all possible pairs of BVs that might need to be tested for overlap. All hierarchical CD algorithms, such as the one in Figure 1, basically perform a traversal through this (conceptual) tree.

the number of intersecting object pairs. They have generalized their approach to first average-shape results in computational geometry [ZS99].

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 each level l of the BVTT, we estimate the number $\tilde{N}_v^{(l)}$ of nodes whose x-overlap is > 0 . That means, given a node and its corresponding BVs $A_i^{(l)}, B_j^{(l)}$, we test whether the overlap $o_{ij}^{(l)}$ of the BVs $A_i^{(l)}$ and $B_j^{(l)}$, when projected onto the x axis, is larger than zero.

Then, we determine the conditional probability that the BV pair $(A_i^{(l)}, B_j^{(l)})$ on level l overlaps

$$p^{(l)} := Pr[A_i^{(l)} \cap B_j^{(l)} \neq \emptyset \mid A \cap B \neq \emptyset \wedge o_{ij}^{(l)} > 0]$$

where (A, B) is the parent node of $(A_i^{(l)}, B_j^{(l)})$ in the BVTT. In contrast to what one might think, it turns out that we do not need individual probabilities per BV pair.

Let X_l denote the number of nodes we visit on level l in the BVTT. Then, its expected number $E(X_l)$ can be determined

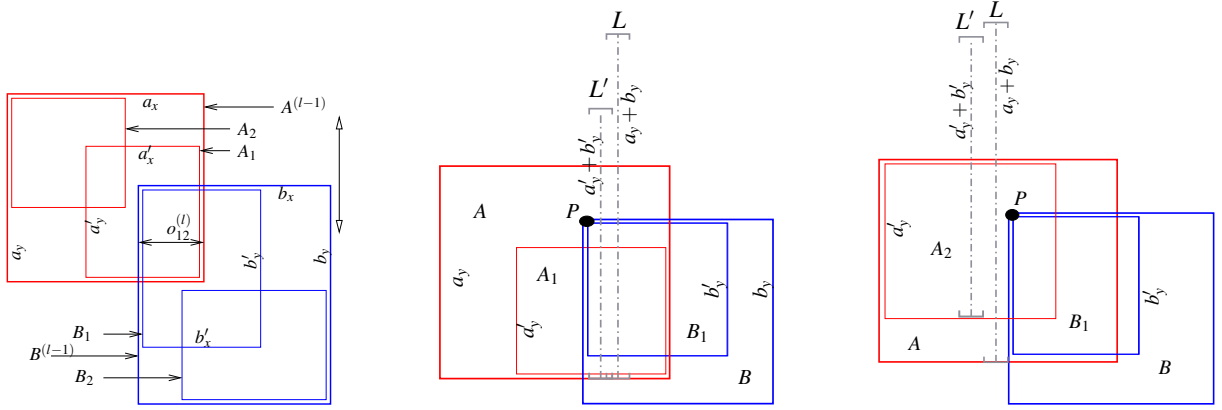


Figure 3: Left: general configuration of the boxes, assumed throughout our probability derivations. For sake of clarity, boxes are not placed flush with each other. Middle: The ratio of the length of segments L and L' equals the probability of A_1 overlapping B_1 . Right: ditto for p_{21} .

by

$$E(X_I) = \tilde{N}_v^{(l)} \cdot \prod_{i=1}^l p^{(l)}$$

where we only assume that the two root BVs $A^{(0)}$ and $B^{(0)}$ overlap (i.e., $p^{(0)} = 1$). Overall, the expected total number of nodes we visit in the BVTT is

$$\tilde{N}_v(n) = E\left(\sum_{I=1}^d X_I\right) = \sum_{I=1}^d \tilde{N}_v^{(l)} \cdot \prod_{i=1}^l p^{(l)} \quad (1)$$

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

In the following, we will first derive an estimate for $p^{(l)}$, then for $\tilde{N}_v^{(l)}$.

3.1. Preliminaries

For sake of simplification, we assume that the child boxes of each BV sit in opposite corners within their respective parent boxes, so that A_1 and B_1 overlap before A_2 and B_1 do (if at all), see Figure 3. In addition, we assume that there is a constant *BV diminishing factor* throughout the hierarchy, i.e.,

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

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

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

This assumption allows us some nice simplifications in Equation 2 and 5 as well as the use of a single ω in Equation 4, but it is not necessary at all.

3.2. Probability of Overlap

Similarly to [Zac02], we could estimate $p^{(l)}$ by the volume of Minkowski sums. However, in that model all positions of B with respect to A would be equally likely, which is not true in practice.

Instead we assume, without loss of generality, that the root box $B^{(0)}$ penetrates $A^{(0)}$ by some known distance $o_{ij}^{(0)} := \delta$ along the x axis from the right. Our analysis considers all configurations 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} (in the following, we will drop the superscript (l)) (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 segment L , which has the same x component as the point P and which length is defined by $a_y + b_y$. (The BV B and, thus, P can be chosen arbitrarily, under the condition that A and B still overlap. L would be shifted accordingly, but its length would be the same.) Note that for sake of illustration, segment L has been shifted slightly to the right from its true position in Figure 3 (center). If we restrict the position of B such that A_1 and B_1 overlap, too, then P remains on segment L' . Thus,

$$p_{11} = \frac{\text{Length}(L')}{\text{Length}(L)} = \frac{a'_y + b'_y}{a_y + b_y} = \alpha_y. \quad (2)$$

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 $o_{ij}^{(l)} > 0$; 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 $p^{(l)} \equiv \alpha_y$ in our model.

3.3. Positive x-Overlap

We can trivially bound $\tilde{N}_v^{(l)}$, the number of nodes whose x-overlap is > 0 , by 4^l . Plugging this in Equation 1 yields

$$\begin{aligned} \tilde{N}_v(n) &\leq \sum_{l=1}^d 4^l \cdot \alpha_y^l = \frac{(4\alpha_y)^{d+1} - 1}{4\alpha_y - 1} \quad (4\alpha_y \neq 1) \\ &\in O((4\alpha_y)^d) = O(n^{\lg 4\alpha_y}). \end{aligned} \quad (3)$$

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

Clearly, $\tilde{N}_v^{(l)} \leq 4^l$ is not a tight bound. So, in the following, we derive a better one.

Each pair $(A_i^{(l)}, B_j^{(l)})$, when projected onto the x axis, has a specific amount of overlap, $o_{ij}^{(l)}$. As mentioned before, only pairs with $o_{ij}^{(l)} > 0$ can possibly intersect (remember that $\tilde{N}_v^{(l)}$ denotes the number of these pairs).

Given $o_{ij}^{(l-1)}$, the x-overlap of $(A^{(l-1)}, B^{(l-1)})$, we can easily compute $o_{ij}^{(l)}$ for each case (see Figure 4):

$$\begin{aligned} \text{case } ij = 11: & \quad o_{11}^{(l)} = o_{11}^{(l-1)}, \\ \text{case } ij = 21: & \quad o_{21}^{(l)} = o_{21}^{(l-1)} - \omega, \\ \text{case } ij = 12: & \quad o_{12}^{(l)} = o_{12}^{(l-1)} - \omega, \\ \text{case } ij = 22: & \quad o_{22}^{(l)} = o_{22}^{(l-1)} - 2\omega. \end{aligned} \quad (4)$$

with $\omega = a_x^{(0)} \alpha_x^{l-1} (1 - \alpha_x)$ and $a_x^{(0)}$ = the extent of the root BV.

So, in order to determine whether or not a pair of boxes $A_i^{(l)}, B_j^{(l)}$ on level l could possibly overlap, we need to determine whether its $o_{ij}^{(l)} > 0$. We can do this by starting at the root of the BVTT, initializing $o_{ij}^{(0)} := \delta$, the overlap of the root boxes when projected onto the x axis. Then we proceed down through the BVTT along the path to $A_i^{(l)}, B_j^{(l)}$, decrementing $o_{ij}^{(0)}$ with each step by 0, ω , or 2ω according to Equation 4, depending on which child we go into.

Clearly, $\tilde{N}_v^{(l)}$ depends on the parameters α_x and δ . So, for a range of parameter values, we could tabulate $\tilde{N}_v^{(l)}$, and then compute a much better estimate for the number of BV overlap tests using Equation 1.

3.4. 3D Collision Detection

As mentioned, our considerations can simply be extended to 3D. Then, L and L' of Equation 2 are not line segments any longer, but 2D rectangles in 3D which are perpendicular to the x axis 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,

$$p_{11} = \frac{\text{area}(L')}{\text{area}(L)} = \frac{(a'_y + b'_y)(a'_z + b'_z)}{(a_y + b_y)(a_z + b_z)} = \frac{4a'_y a'_z}{4a_y a_z} = \alpha_y \alpha_z. \quad (5)$$

probability $p^{(l)}$	$T(n)$
$< 1/4$	$O(1)$
$1/4$	$O(\lg n)$
$\sqrt{1/8} \approx 0.35$	$O(\sqrt{n})$
$1/2$	$O(n)$
$3/4$	$O(n^{1.58})$
1	$O(n^2)$

Table 1: Effect of the probability $p^{(l)}$ on the running time of a simultaneous hierarchy traversal.

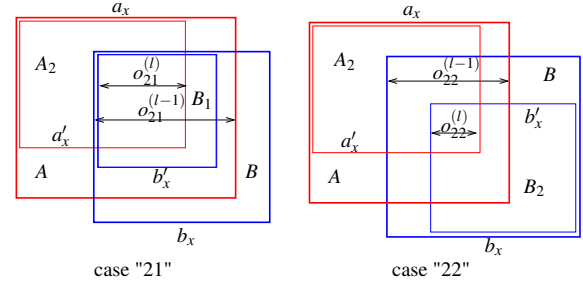


Figure 4: Denotations for computing $o_{ij}^{(l)}$ for a child pair. The case "12" is symmetric to "21", and the case "11" is trivial. Here, $\omega = a_x^{(0)} \alpha_x^0 (1 - \alpha_x) = a_x^{(0)} (1 - \alpha_x)$.

The other probabilities p_{ij} can be determined analogously as above, so that $p_{11} = p_{12} = p_{21} = p_{22} = \alpha_y \alpha_z$. Then, we can estimate the number of BV overlap tests by

$$\tilde{N}_v(n) = \sum_{l=1}^d \tilde{N}_v^{(l)} \cdot \alpha_y^l \cdot \alpha_z^l, \quad (6)$$

where $d = \log_4(n^2) = \lg(n)$. Figure 6 shows this number for different parameters $\alpha_{x,y,z}$ (that means $\alpha_x, \alpha_y, \alpha_z$) and δ .

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

4. Experimental Support

We have implemented (using C++) the algorithm shown in Figure 1 using AABBs as BVs. As we are only interested in the number of visited nodes in the BVTT, we switched off the intersection tests at the leaf nodes.

We used a set of CAD objects, each of them with varying complexities with respect to the number of polygons (Fig. 1). Benchmarking is performed by the procedure of [Zac02], which can compute the average number of overlapping BVs for a distance dc between two identical objects.

Fig. 5 (right) shows the number of BV overlap tests for our models depending on their complexities for a fixed distance $dc = 0.4$. Clearly, the average number of BV overlap tests behaves logarithmically for all our models.

We also examined, whether $p^{(l)}$ can be estimated by the

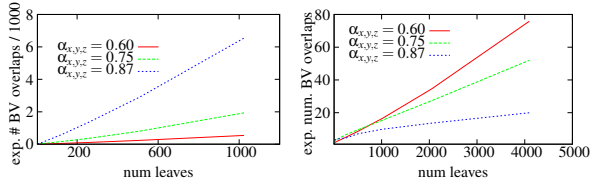


Figure 6: Overall expected number of overlapping BV pairs $\tilde{N}_v(n)$ ($=$ nodes in the BVTT), which is proportional to the expected running time. Left: $\delta = 0.4$; right: $\delta = 0.15$.

volume of Minkowski sums where all positions of B with respect to A are equally likely [Zac02]. In this case,

$$p^{(l)} = \alpha_x \alpha_y \alpha_z.$$

To measure the the number of BV overlap tests, we used artificial BVHs where we can adjust $\alpha_{x,y,z}$, and thus $p^{(l)}$. There, the child pairs of each node are placed in its opposite corners. Fig. 5 (left) shows the number of BV overlap tests depending on n for different choices of $\alpha_{x,y,z}$. If $\alpha_{x,y,z} = 0.5$, then $p^{(l)} = 0.125$ and, as also shown by Table 1, the running time seems to be constant. Moreover, the plot shows that, if $\alpha_{x,y,z} = 0.63$ so that $p^{(l)} = 0.25$, the number behaves logarithmically.

If $\alpha_{x,y,z}$ is chosen so that $p^{(l)} = 0.5$, then the number of overlaps behaves linear as one can see in Fig. 5 (center). Overall, $p^{(l)}$ can also be estimated by the volume of Minkowski sums.

5. Conclusion and Future Work

We have presented an average-case analysis for simultaneous hierarchical BV traversals that provides a better understanding of the performance of hierarchical collision detection that has been observed in the past. Our analysis is independent of the order of the traversal.

Several existing methods for hierarchical collision detection may derive 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.

Currently, we are working towards a closed-form description of the distribution of the o 's on each level of the BVTT, which seems to resemble a binomial distribution. This would allow us to analytically compute the number of BV pairs that could possibly overlap.

Furthermore, 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.

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.: OBB-Tree: 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.

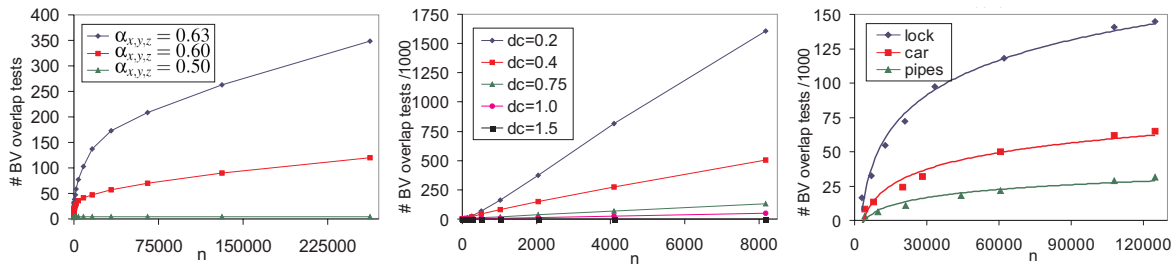


Figure 5: Number of BV overlap tests depending on the number n of leaves (left, center: artificial BVHs, right: BVHs for models shown in Fig. 1). Left: the distance dc between centers of the root BVs is set to 1. Center: if $\alpha_{x,y,z}$ is chosen so that $p^{(l)} = 0.5$, the number of BV overlap tests is linear in n , independent of the distance dc . Right: plots for our models at distance $dc = 0.4$.

[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.