Technical Report TUBS-CG-2003-05

Easy Realignment of k-DOP Bounding Volumes

Christoph Fünfzig, Dieter W. Fellner
{c.fuenfzig,d.fellner}@tu-bs.de

Institute of Computer Graphics
University of Technology
Mühlenpfordtstr. 23, D-38106 Braunschweig
http://graphics.tu-bs.de

Easy Realignment of k-DOP Bounding Volumes

Christoph Fünfzig Dieter W. Fellner

Institute of ComputerGraphics
Technical University of Braunschweig, Germany

Abstract

In this paper we reconsider pairwise collision detection for rigid motions using a *k*-DOP bounding-volume hierarchy. This data structure is particularly attractive because it is equally efficient for rigid motions as for arbitrary point motions (deformations).

We propose a new efficient realignment algorithm, which produces tighter results compared to all known algorithms. It can be implemented easily in software and in hardware. Using this approach we try to show, that *k*-DOP bounding-volumes can keep up with the theoretically more efficient oriented-bounding-boxes (OBBs) in parallel-close-proximity situations.

Key words: virtual reality, collision detection, distance computation

1 Introduction

Collision detection is an important component in any VR application with user interaction like virtual prototyping and virtual simulation systems. In general the collision detection component can be structured into a pipeline much like the rendering pipeline [18].

In Figure 1 the collision detection pipeline is shown with its pipeline stages. The frontend consists of the object handler, which allows to define and identify the objects and to state the application's collision interest. Then comes a first neighbor-finding stage, which reduces the set of all objects to smaller neighbor sets of the interesting objects. Within each neighbor set a pairwise collision detection is performed. For this pairwise collision detection problem several algorithms have been proposed. The first class uses a hierarchy of simpler bounding volumes on the face set to stop the search for colliding faces in sublinear time. Here the performance depends on the tightness of the bounding volume and on the complexity of the collision test for the bounding volume. A wealth of different bounding volumes have been proposed: spheres [6], oriented-bounding-boxes [11], axis aligned bounding boxes [16], k-DOPs (generalization of axis aligned bounding boxes) [7], swept sphere volumes [9], up to arbitrary convex polytopes [3]. Roughly speaking, the bounding volumes like oriented bounding boxes and arbitrary convex polytopes have very good enclosing properties, moderately complex collision tests and complex constructions of the bounding volume and the bounding volume hierarchy. Spheres, axis aligned bounding boxes and its generalization k-DOPs have very simple collision tests and constructions of the bounding volume and the bounding volume hierarchy, but have only moderately good enclosing properties. Because of the simple constructions of the bounding volume itself and the bounding volume hierarchy they are particularly suited for arbitrary point motions (deformations) like in cloth animation [12].

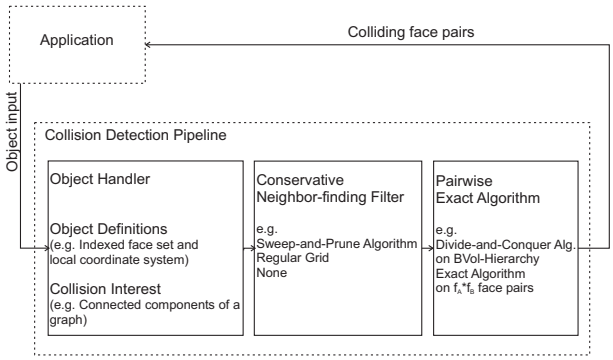

Figure 1: Collision detection can be considered as a pipeline of successive filters. The frontend handles complete objects (of a scenegraph system), intermediate stages use some simpler entities like bounding volumes and the backend handles face pairs.

In this paper we are concerned with pairwise collision detection for the special case of euclidean motions using k-DOP bounding volumes. A *discrete orientation polytope (k-DOP)* is defined by a *fixed* small set of k directions $(D_1,\ldots,D_k)$ and a tuple $(d_1,\ldots,d_k) \in \mathbb{R}^k$ of scalars by

$$\{p \mid D_i \cdot p \le d_i,\ i = 1,\ldots,k\} = \bigcap_{i=1}^{k} H_i \qquad (1)$$

with halfspaces

$$H_i := \{p \in \mathbb{R}^3 \mid D_i \cdot p \le d_i\}$$

Usually the set of directions is restricted, so that for each direction D_i there is an antiparallel one $D_{i+k/2} = -D_i$

$$\{p \mid D_i \cdot p \le d_i,\ i = 1,\dots,k\} \quad = \quad \bigcap_{i=1}^{k/2} S_i \qquad (2)$$

both together defining so-called *slabs*

$$\begin{aligned} S_i &= \{p \in \mathbb{R}^3 \mid -d_{i+k/2} \le -D_{i+k/2} \cdot p, \\ &\qquad\qquad D_i \cdot p \le d_i\} \\ &= \{p \in \mathbb{R}^3 \mid -d_{i+k/2} \le D_i \cdot p \le d_i\} \end{aligned}$$

With this restriction the collision test for k-DOPs reduces to $k/2$ interval overlap-tests. In order to use this simple collision test also with euclidean motions, at least one of the tested k-DOPs is defined by rotated directions (D_i') and has to be realigned to the original directions (D_i). This paper proposes a new algorithm for realigning k-DOPs which is more efficient than all known algorithms, and also produces tighter results. The calculation of the bounding volume and the bounding volume hierarchy is only slightly more complex.

In the next section we briefly summarize previous work on the realignment problem for k-DOP bounding volumes. In Section 3 we first present our realignment approach for bounded rotations. Afterwards we describe how to extend it to the full range of rotations by applying a remapping of directions. With this the full k-DOP intersection test can be presented. Section 5 then extends to proximity computations: Minimum Distance and Approximate Minimum Distance. In Section 6 we show the performance statistics for the whole spectrum of collision scenarios and compare with the other realignment algorithms and with the performance of RAPID [10], using OBBs as bounding volumes. Finally, we conclude and identify directions for future work.

2 Previous work

In a scenegraph system two objects A and B are usually defined by their geometry in a local coordinate system, which is given by a transformation matrix M_A and M_B, respectively. In case of the k-DOP bounding volumes given in the local coordinate systems, we can do the collision tests in A's (or B's) coordinate system.

The *realignment problem* then is to calculate a k-DOP bounding volume for $M \cdot B$, where $M = M_A^{-1} \cdot M_B$ is the matrix for the change of coordinate system from object B to object A.

Three approaches for the realignment problem can be found in the literature. In the original paper Klosowski et al [7] propose two methods: *hill-climbing method* and *approximation method*. The hill-climbing method relies on the convex hull, stored with each inner node of the bounding volume hierarchy of object B. During traversal the convex hull is lazily transformed by matrix M, to find new bounds $d'_{i+k/2}$ and d'_i for the k-DOP. This method is quite expensive in time and space and as stated in [7] it is only justified for the root node. The approximation method does not try to compute a k-DOP from scratch. Instead, it transforms the vertices of the original k-DOP by matrix M and computes a k-DOP for the transformed vertex set (see Figure 2). For non-degenerate situations this requires $\Omega(k)$ vertex transformations and $\frac{k}{2}\Omega(k)$ scalar product computations.

All other work tries to make the approximation method more computationally efficient. In [16, SATlite] only the major axis directions $\{D_0 = (1,0,0), D_1 = (0,1,0), D_2 = (0,0,1)\}$ are considered for the realignment problem (see Figure 3). The resulting approximation algorithm requires only k scalar product evaluations.

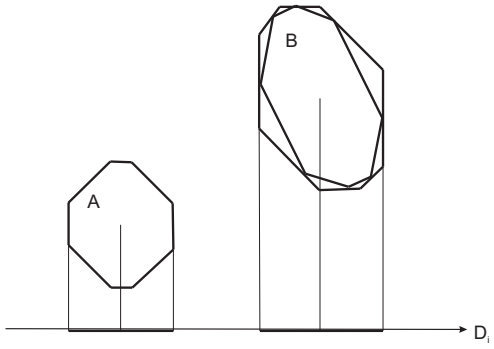

Figure 2: Approximation method using all k-DOP directions.

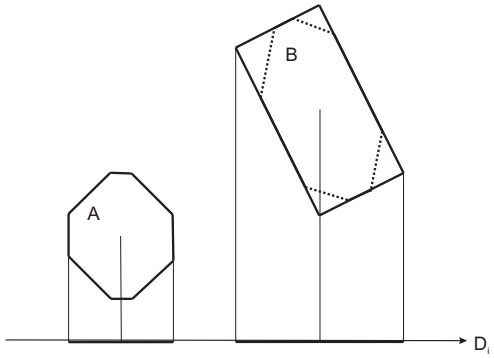

Figure 3: Approximation method using only the major axis directions.

The approximation method in its original form requires to compute the boundary representation (BRep) of the k-DOP. Possible algorithms for the special case of 18-DOPs

can be found in [1] and for the general case in [14, Convex polyhedrons as halfspace intersections]. Zachmann [17] proposes a different implementation of the approximation method (figure 2), which does only require the BRep for the unit DOP (all $d_i = 1$). The algorithm computes the new bounds d'_i by a scalar product with a vector D''_i

$$\begin{aligned} d'_i &= D'_i \cdot \begin{pmatrix} D_{j_{i,1}} \\ D_{j_{i,2}} \\ D_{j_{i,3}} \end{pmatrix}^{-1} \cdot \begin{pmatrix} d_{j_{i,1}} \\ d_{j_{i,2}} \\ d_{j_{i,3}} \end{pmatrix} \\ &= D''_i \cdot \begin{pmatrix} d_{j_{i,1}} \\ d_{j_{i,2}} \\ d_{j_{i,3}} \end{pmatrix} \end{aligned} \tag{3}$$

where $j_{i,h}$, $1 \le h \le 3$ are the indices of k-DOP halfspaces $H_{j_{i,h}}$, supporting an extremal vertex p_i of the original k-DOP. The correspondence $j_{i,h}$, $1 \le h \le 3$ and therefore D''_i is the same for all DOPs in the tree and can be computed once at the beginning of the tree traversal from the unit DOP. Altogether this algorithm requires k matrix inversions and k matrix-vector products once and k scalar product computations per realignment.

3 Realignment as Table Lookup

This work proposes a new realignment method, falling in between the approximation method and the hill-climbing method. First, a point c used as the rotation center is stored with each node of the bounding volume hierarchy. Its coordinates are set as the center-of-mass of the contained points.

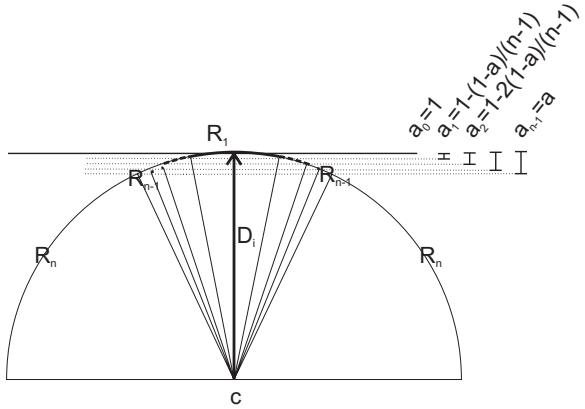

Figure 4: Angle cone, subdivided into $n-1$ rings R_j plus the remaining directions R_n (shown in two dimensions)

Then the k-DOP bounding volume is calculated as usual. In a second pass for each k-DOP direction D_i information is added about the point locations in a conical region around D_i (see Section 3.1). The conical region is subdivided into a number of rings R_j, each with a single bit j recording if there is a point in the voidage region G_j. During the realignment algorithm this information is used to scale the value d_i according to the transformed directions D'_i. For arbitrary rotations we need an additional remapping σ of directions D_i, which is described in Section 3.2.

3.1 Realignment for Bounded Rotations

In order to keep the information about the point locations in the neighborhood of direction D_i small, we have to keep the angle between directions D_i and D'_i small.

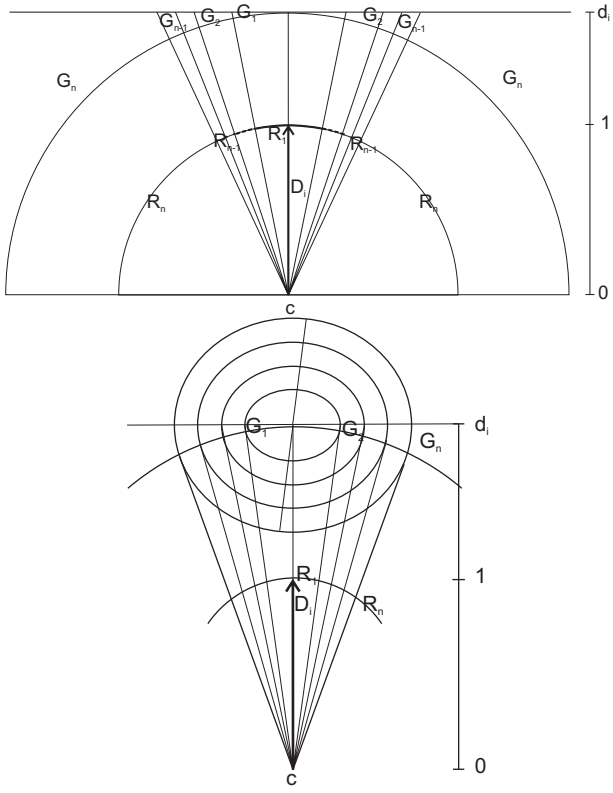

Figure 5: Voidage regions G_j in halfspace H_i

Suppose $D_i \cdot D'_i \le a$ with $0 \ll a \le 1$. Then we subdivide the angle cone with aperture a into $n-1$ rings

$$a_j := \quad 1 - j\tfrac{1-a}{n-1}$$

$$R_j := \{D \in S_2 \mid \quad a_{j-1} > D \cdot D_i \ge a_j\}$$

with $j = 1, \ldots, n-1$ and the remaining directions

$$R_n := \{D \in S_2 \mid \quad a > D \cdot D_i\}$$

as shown in Figure 4.

In the second phase of the k-DOP bounding volume construction we store the information, if there is a point

p in the voidage region G_j with

$$G_j := \{p \mid \quad \|p-c\| > d_i,\ \frac{p-c}{\|p-c\|} \in R_j\}$$

For efficiency we choose $n = 32$ and encode it as bit j in an `unsigned` data member. Figure 5 shows the geometric meaning of this *occupancy table*, which contains the occupancy status of all regions G_j.

For the points p in the remaining region G_n we calculate the projected length l_p of $(p-c)$ onto the shell of the outermost ring R_n. With these projected lengths we set

$$r_i := \min_{p \in G_n} \left\{ \frac{d_i}{l_p} \right\} \tag{4}$$

as the minimum ratio between the scalar d_i and the projected lengths (Figure 6).

Using the occupancy table the realignment problem can be solved as follows. Let direction D'_i be in R_m for index $m \in \{1,\ldots,n-1\}$.

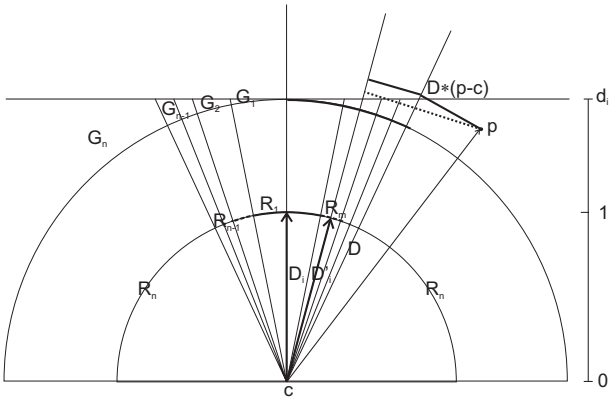

Figure 6: Geometric consideration for points lying in the remaining region G_n (shown in two dimensions)

1. First consider the remaining region G_n and set

$$f_i := \frac{a_{(n-2)-m}}{r_i} \tag{5}$$

if bit n is set. Otherwise we initialize $f_i := a$. The resulting halfspace $H'_i = \{p \mid D'_i p \le f_i d_i\}$ contains the points, lying in the region G_n. Figure 6 shows why this statement holds.

2. Consider all inner regions G_{m-h}, $h = 0,\ldots,m$ and evaluate

$$f_i := \max\left(f_i, \frac{a_{h-1}}{a_{m-h}}\right) \tag{6}$$

Because the sequence $\frac{a_{h-1}}{a_{m-h}}$ is monotonously decreasing the maximum search can be stopped for the first bit $m-h$ found to be set. In the term $\frac{a_{h-1}}{a_{m-h}}$ the denominator gives the length of the shell of ring R_{m-h} and therefore is a lower bound on the projection of a direction in ring R_{m-h}. The numerator is just an upper bound on the cosine between the direction of the point and D'_i (see Figure 7).

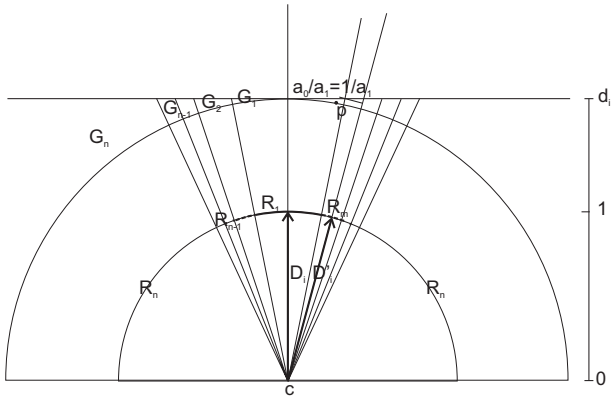

Figure 7: Geometric consideration for points lying in inner regions G_{m-h} relative to direction D'_i (shown in two dimensions)

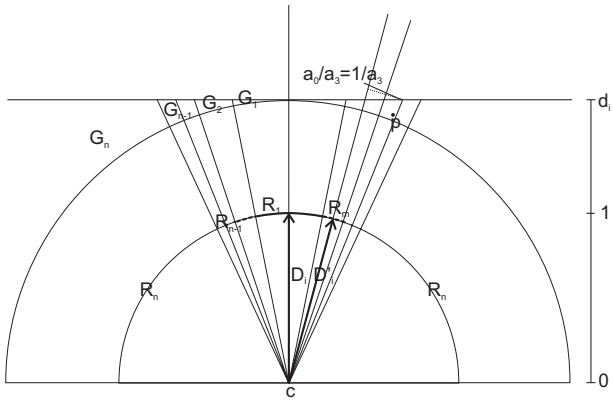

Figure 8: Geometric consideration for points, lying in outer regions G_{m+h} relative to direction D'_i (shown in two dimensions)

3. Consider all outer regions G_{m+h}, $h = 1,\ldots,(n-2)-m$ and evaluate

$$f_i := \max\left(f_i, \frac{a_{h-1}}{a_{m+h}}\right) \tag{7}$$

This sequence $\frac{a_{h-1}}{a_{m+h}}$ is monotonously increasing and again the maximum search can be stopped for the first bit $m+h$ found to be set, if searching inversely. The explanation for the term $\frac{a_{h-1}}{a_{m+h}}$ is similar to the previous one (see Figure 8).

We then use the factor f_i calculated to scale the value d_i

$$d_i' := f_i d_i \tag{8}$$

3.2 Remapping of Directions

In the previous section we have assumed that direction D_i' satisfies $D_i \cdot D_i' \leq a$. Now we show how to ensure that for arbitrarily rotated directions D_i'.

First note, that for a k-DOP the aperture a has to be chosen large enough to cover the directions of $\mathbb{S}_2$ completely (see Figure 9).

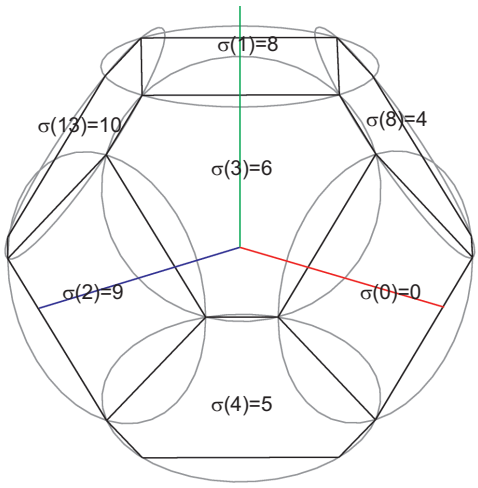

Figure 9: Remapping of directions (here with the 14-DOP). Note, that the conical regions R_j (with aperture a) cover the sphere $\mathbb{S}_2$ completely!

The value a depends on k and the regularity of directions D_i. In Table 1 the values of a are given for 14-DOP, 18-DOP, 26-DOP and for the platonic solids 6-DOP (Cube), 12-DOP (Dodecahedron). For performance reasons larger values of a are preferable.

	a
6-DOP	0.57735 (54.7 deg)
12-DOP	0.787539 (38.04 deg)
14-DOP	0.806898 (36.2 deg)
18-DOP	0.816497 (35.26 deg)
26-DOP	0.886452 (27.57 deg)

Table 1: Aperture a of direction cone for various k-DOP bounding volumes.

At the beginning of the bounding-volume-tree traversal we once calculate *a remapping* σ of directions

$$\begin{aligned}&\sigma(i) := h, \text{ so that}\\ &D_i' \text{ lies in region } R_m \text{ relative to direction } D_h\end{aligned} \tag{9}$$

However, note that the remapping σ is not a permutation in general. Besides finding the remapping, the index m of the region (relative to direction $D_{\sigma(i)}$) can be calculated easily.

4 Intersection Test

For the full k-DOP intersection test we have to add all components together. If the k-DOP of object B is realigned, we need the occupancy table `dop2.occ` and the values `dop2.`r_i to calculate the scaling factors f_i. Additionally the rotation center `dop2.`c is required to calculate the movement `correct` due to the rotated center. This is given by the matrix-vector product $D_i \cdot (M$ `dop2.`$c)$, projected onto direction D_i. In the actual implementation it is done in one scalar product using a table of the scalar products $(D_i \cdot M_{\cdot 1}, D_i \cdot M_{\cdot 2}, D_i \cdot M_{\cdot 3})$ with the columns of the matrix M.

```
for (i=0; i<k; ++i) {
 unsigned mini = sigma(i+k);
 unsigned maxi = sigma(i);
 real correct = dop2.c1(Di · M.1)
                +dop2.c2(Di · M.2)
                +dop2.c3(Di · M.3);
 real min2 = (dop2.dmini -dop2.cmini)·fmini
             + correct;
 real max2 = (dop2.dmaxi -dop2.cmaxi)·fmaxi
             + correct;
 if (max(dop1.di+k,min2)>min(dop1.di,max2))
    return false;
}
return true;
```

Altogether the intersection test requires k multiplications in Equation 8, k multiplications/divisions in Equation 5 and k scalar products (for the rotation of the center point). The quotients in Equations 6 and 7 can be evaluated by a two dimensional table lookup. With $5k$ multiplications the method is nearly as efficient as the approximation method restricted to the major axis directions.

5 Extension to Proximity Computation

In this section we show how to use the realignment algorithm for proximity computation problems.

Minimum Distance Find a face pair with minimum distance.

Approximate Minimum Distance Given an absolute error or relative error, find all face pairs with distance values lying within the error bounds above the minimum distance.

Computation of the approximate minimum distance can be done by solving a weaker problem first:

Weak Approximate Minimum Distance Given an absolute error or relative error, compute a sequence of face pairs $(f_{A,l}, f_{B,l})_{l=1,\ldots,n}$, such that

$$\mathrm{dist}(f_{A,i}, f_{B,i}) \stackrel{\cdot}{\leq} \min_{j=1,\ldots,i-1} \{\mathrm{dist}(f_{A,j}, f_{B,j})\}$$

is within the error bounds for all i and $\min_{j=1,\ldots,n}\{\mathrm{dist}(f_{A,j}, f_{B,j})\}$ is the minimum distance.

In a second pass over the resulting sequence of the Weak Approximate Minimum Distance problem we can then solve the Approximate Minimum Distance problem.

The algorithm for the Weak Approximate Minimum Distance problem using k-DOP bounding volumes is of branch-and-bound type [8]. The pruning with lower bounds on the minimum distance of the faces, contained in two k-DOPs `dop1` and `dop2`, is done as follows

```
for (i=0; i<k; ++i) {
 unsigned mini = sigma(i+k);
 unsigned maxi = sigma(i);
 real correct = dop2.c1(Di · M.1)
                +dop2.c2(Di · M.2)
                +dop2.c3(Di · M.3);
 real min2 = (dop2.dmini −dop2.cmini)·fmini
             + correct;
 real max2 = (dop2.dmaxi −dop2.cmaxi)·fmaxi
             + correct;

 real diff1 = dop1.di −min2;
 real diff2 = dop1.di+k −max2;
 if (diff1>=0 && diff2>=0)
  if (diff2>getDistance()+getAbsError())
   return false;
 else if (diff1<0 && diff2<0)
  if (-diff1>getDistance()+getAbsError())
   return false;
}
return true;
```

6 Results

We have implemented a system based on k-DOP bounding volumes for collision detection and proximity computations on top of the OpenSG scenegraph. This C++ implementation has a `DoubleTraverser` template-class, which traverses two bounding volume hierarchies in parallel and does the dispatching of the four possible cases: InnerNode–InnerNode, InnerNode–LeafNode, LeafNode–InnerNode and LeafNode–LeafNode. The different versions of this template-class incorporate different caching strategies, like the generalized front cache [3] or the simple caching of a single face pair. By a `Traits` class in the template parameters it can be configured to use any intersection test or proximity computation with one of the realignment algorithms presented.

In order to show the efficiency of the new realignment method, we have run a series of benchmarks on a Windows-PC with 1GHz processor clock-rate. Benchmarking collision detection algorithms realistically is a difficult task. This is so, because there is a wealth of models with different characteristics and motions relative to each other [15]. In [17] a benchmarking scheme for two models each contained in a unit-box is proposed, where the second object performs full-z rotations at decreasing distances relative to the first one.

Figure 10 shows the intersection test times for the approximation method restricted to the major-axis directions, the approximation method using all directions and the new realignment algorithm. For the benchmarks the k-DOP intersection test is used also in the mixed situations InnerNode–LeafNode and LeafNode–InnerNode, instead of the primitive test using the face in the leaf node. In practical applications it is reasonable to use only the primitive tests in mixed situations and safe for the k-DOP bounding volumes in leaf nodes. This safes one half of the total memory consumption. In each test the realignment algorithm is performed on the fly and the realignment result is never cached, which would reduce the number of realignments to one half. Concerning the construction of the hierarchy we always used a binary hierarchy with subdivision of the longest k-DOP direction at the midpoint median [10], because it is well known and the node depths are balanced. Of course, there are other possibilities [17]. For the triangle-triangle intersection test we use the approach proposed by Möller [13] with some simple modifications (i.e. without divisions).

The new realignment algorithm performs good on average, but is sometimes outpaced by the approximation algorithm using all k-DOP directions.

Figure 11 shows the performance in a classic scenario: parallel-close-proximity. Here we have modified the scenario of a sphere containing another sphere of slightly smaller radius by randomly perturbing the inner sphere coordinates. As already stated in [10] and [15], RAPID performs especially good in this scenario. With the new realignment algorithm the number of bounding volume and triangle tests reduce with increasing k. Unfortunately in the current implementation this decrease does not compensate the cost increase per bounding volume test.

Figure 12 shows the performance in another classic scenario: point-close-proximity. In this scenario with an increasing number of collisions RAPID has a very homogeneous behavior and falls in between all the k-DOP algorithms.

7 Conclusion and Future Work

We have presented a system for collision detection and proximity computations with *k*-DOP bounding volumes (on top of the OpenSG scenegraph). Integrated into this system is a new realignment algorithm for the *k*-DOPs, which is slightly more complex to implement than the simple approximation algorithm using only the major directions. Because it never requires the boundary representation of a k-DOP, it is much easier to implement than the approximation algorithm using all *k*-DOP directions, presented in [17]. In its current implementation it is only sometimes more efficient than RAPID.

Our next step will be to do some low-level optimizations (like optimizing memory layout) guided by profiling. We want to further remove the direction insensitivity of the approach (see Figure 5). One idea for this might be to subdivide the rings further into ring sections. Currently the *k*-DOP bounding volumes are defined in the local coordinate system of the models. It might be reasonable to choose a different coordinate system for better performance.

An efficiency analysis of this algorithm, as done in [19] for axis-aligned bounding boxes, also requires some further research.

Acknowledgements

This work has been supported by the German Research Foundation (**DFG**) under grant Fe-431/4-3 and by the German Federal Ministry of Education and Science (**BMB+F**) in the OpenSG PLUS project under grant 01 IRA 02G.

References

[1] A. Crosnier and J.R. Rossignac. Tribox bounds for three-dimensional objects. *Computers & Graphics*, 23(3):429–437, 1999.

[2] Olivier Devillers and Philippe Guigue. Faster triangle-triangle intersection tests. Rapport de recherche 4488, INRIA, 2002.

[3] S. Ehmann and M.C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In *Computer Graphics Forum (Proc. of Eurographics 2001)*, 2001.

[4] M. Held. Erit — a collection of efficient and reliable intersection tests. Technical report, University at Stony Brook, 1996.

[5] M. Held, J.T. Klosowski, and J.S.B. Mitchell. Evaluation of collision detection methods for virtual reality fly-throughs. In *Proc. 7th Canadian Conference on Computational Geometry*, pages 205–210. Canadian Conference on Computational Geometry, August 1995.

[6] Philip M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

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

[8] P. Konecny and K. Zikan. Lower bound of distance in 3d. Technical Report 1, Masaryk University, 1997.

[9] E. Larsen, S. Gottschalk, M.C. Lin, and D. Manocha. Distance queries with rectangular swept sphere volumes. In *Proc. of IEEE Int. Conference on Robotics and Automation*, 2000.

[10] M.C. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *Proc. of IMA Conference on Mathematics of Surfaces*, 1998.

[11] M.C. Lin, S. Gottschalk, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Computer Graphics (SIGGRAPH Conference Proceedings)*, 1996.

[12] J. Mezger, S. Kimmerle, and O. Etzmuß. Improved collision detection and response techniques for cloth animation. Technical Report 5, WSI/GRIS, Universität Tübingen, Aug 2002.

[13] T. Möller. A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 2(2):25–30, 1997.

[14] F.P. Preparata and M.I. Shamos. *Computational Geometry — An Introduction*. Springer Verlag, New York, 1985.

[15] P. Terdiman. Opcode benchmark, 2002. http://www.codecorner.com/OpcodeDemo.htm.

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

[17] G. Zachmann. Rapid collision detection by dynamically aligned dop-trees. In *Proc. of IEEE Virtual Reality Annual International Symposium; VRAIS 98*, 1998.

[18] G. Zachmann. Optimizing the collision detection pipeline. In *Proc. of The First International Game Technology Conference GTEC*, Jan 2001.

[19] Yunhong Zhou and Subhash Suri. Analysis of a bounding box heuristic for object intersection. *Journal of the ACM (JACM)*, 46(6):833–857, 1999.

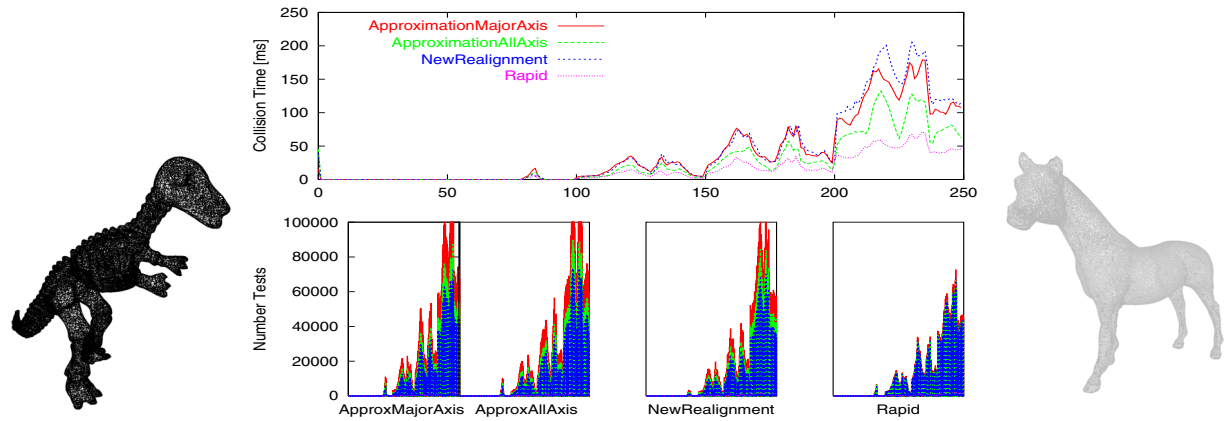

Figure 10: Intersection test statistics for the models Dinosaur and Horse (around 20000 triangles each) with 14-DOPs. In the lower row the diagrams give the number of bounding-volume tests (blue)/mixed tests (green)/triangle tests (red).

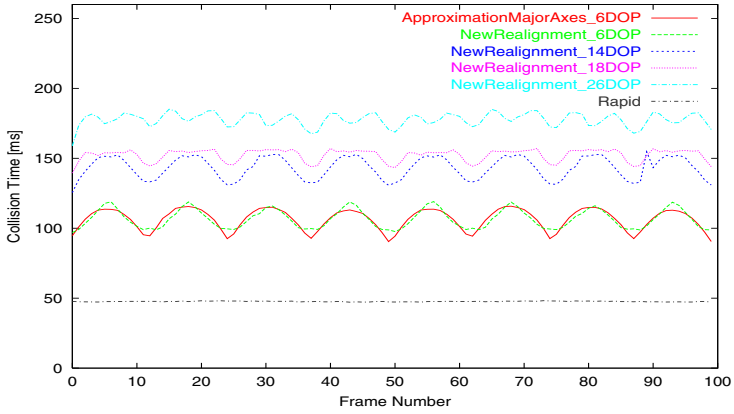

Figure 11: Intersection test statistics for the parallel-close-proximity scenario (sphere models with 5140 triangles each).

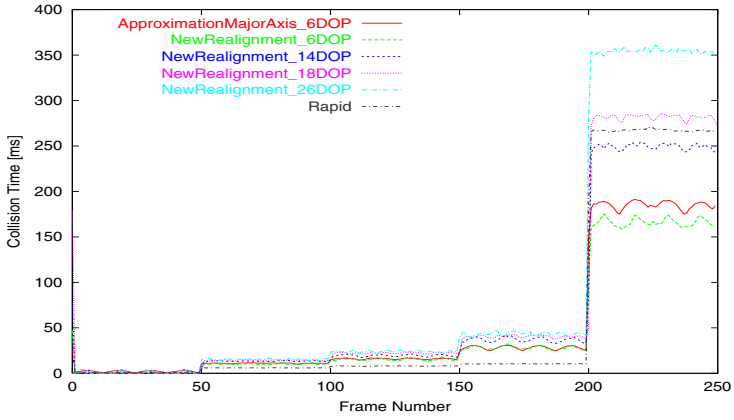

Figure 12: Intersection test statistics for the point-close-proximity scenario (sphere models with 5140 triangles each).