Vol. 13, No. 1: 31–43

An Efficient Ellipsoid-OBB Intersection Test

Thomas Larsson
Mälardalen University

Abstract. An efficient algorithm to determine the intersection status between arbitrarily oriented ellipsoids and boxes (OBBs) is presented. By choosing a proper representation of the geometric objects and by utilizing an affine transformation of space, the problem is converted into a corresponding sphere-parallelepiped intersection test. Thereby, the evaluation of more costly mathematical operations other than a constant number of simple arithmetic operations and comparisons can be avoided. Further efficiency is also gained by exploiting the regularity of the involved geometric shapes, as well as early rejection tests. Practical experiments show a four-times speed-up on average when these techniques are used.

1. Introduction

Bounding volumes (BVs) are often used to speed up computations in various types of computer graphics applications and physical simulation systems [Ericsson 05, Glassner 89]. In particular, oriented bounding boxes (OBBs) and ellipsoids are excellent convex bounding volumes. Compared to spheres and axis-aligned boxes, they have a much tighter fit, and efficient intersection tests have been presented for both the OBB-OBB [Gottschalk et al. 96] and the ellipsoid-ellipsoid case [Choi et al. 06, Wang et al. 01]. As shown by, e.g., Gottschalk et al., the tightness of the BV may be of critical importance for the execution speed of hierarchical collision-detection schemes [Gottschalk et al. 96].

1086-7651/08 $0.50 per page

Since it is evident that OBBs provide a tighter fit for some objects, and ellipsoids for others, it seems possible to improve performance by using a mix of these BV types in the same simulation. However, this requires an efficient algorithm for determining if an arbitrarily oriented ellipsoid overlaps an OBB. Therefore, a novel algorithm for this purpose is presented. For the simpler sphere-OBB case, efficient overlap tests have already been proposed [Larsson et al. 07].

A key technique used in the algorithm is to apply an affine mapping to transform the ellipsoid to a sphere and the OBB to a parallelepiped, thereby simplifying the original problem to determine whether the sphere and the parallelepiped intersect in the transformed space. The transformation of an ellipsoid to a sphere can also be utilized in, e.g., games to find collisions between a player, approximated by an ellipsoid, and a polygonal game environment [Nettle 00]. A similar approach is sometimes used in ray tracing, where intersection calculations can be carried out by transforming the ray to canonical object space to simplify computations.

Ratschek and Rokne presented an efficient two-dimensional ellipse-rectangle intersection test using an alternative transformation approach, i.e., they suggest rotating the objects first so that the rectangle becomes axis-aligned with the principal coordinate axes [Ratschek and Rokne 02].

2. Ellipsoid-Box Overlap Test

The key observation behind the developed algorithm is that the overlap status of geometric primitives does not change if the space they reside in is deformed by a non-singular (invertible) affine transformation. This is assured by the following well-known properties of this class of transformations: (1) collinear points remain collinear, (2) the relative ratio of length between line segments remains constant, and (3) the transformation is a bijection, i.e., there is a one-to-one correspondence between points.

To simplify the required computations, it is thus possible to apply an affine map that transforms the ellipsoid to a sphere and the OBB to a parallelepiped. After that, the intersection status can be determined by tracking the closest distance between the center of the sphere and the parallelepiped, which is compared to the radius. To simplify the computations even further, the transformation is chosen so that the center of the sphere is at the origin. In what follows, the resulting deformed space is referred to as the *canonical sphere space of the ellipsoid*, or simply as the *transformed space*.

Figure 1 shows the parameters used to represent the geometric shapes (in the 2D case). The ellipsoid is represented by a center point $\mathbf{c}$, half lengths l_x, l_y, and l_z, and orthonormal basis vectors $\mathbf{u}_0$, $\mathbf{u}_1$, and $\mathbf{u}_2$ defining the orientation. Similarly, the OBB is represented by a midpoint $\mathbf{m}$, half extents

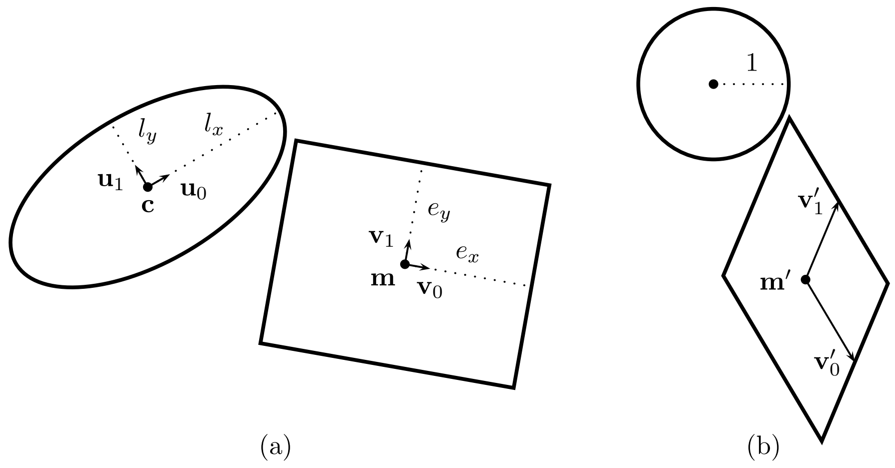

Figure 1. (a) Illustration in 2D showing the representation used for the ellipsoid and the OBB. (b) The corresponding illustration in 2D of the resulting sphere and parallelepiped shapes after the transformation of space. The parameters of the parallelepiped are computed according to Equations (2)–(5).

e_x, e_y, and e_z, and orthonormal basis vectors $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_2$. Note that all the length parameters $l_x, l_y, l_z, e_x, e_y, e_z > 0$. In Figure 2, example images of the involved 3D volumes are shown.

An overview of the major steps in the algorithm is given in pseudocode in Algorithm 1. The overlap status is determined as follows: First, the algorithm selects the set S of visible faces of the OBB seen from $\mathbf{c}$ for further processing (Line 1). If no visible face is found, then $\mathbf{c}$ is inside the box and overlap has been trivially detected, and the algorithm is aborted (Line 2). Otherwise, the transformation of space is performed (Line 3). Then the one to three visible faces of the parallelepiped in S' are examined in turn to see if they can signal either an intersection rejection or detection (Lines 4–8). The details of these main steps of the algorithm are presented in the following sections.

2.1. Inside Condition and Visible-Face Selection

We can test whether the center of the ellipsoid is inside the OBB very efficiently by projecting $\mathbf{c}$ onto the axes of the box. Let the vector $\mathbf{w} = \mathbf{c} - \mathbf{m}$. Then, since the axes of the OBB are of unit length, the projection parameters, computed as $d_i = \mathbf{w} \cdot \mathbf{v}_i$, directly give the distances from the center of the box to the projected points. Only if $d_i \leq e_i$ for all axes is $\mathbf{c}$ inside the box.

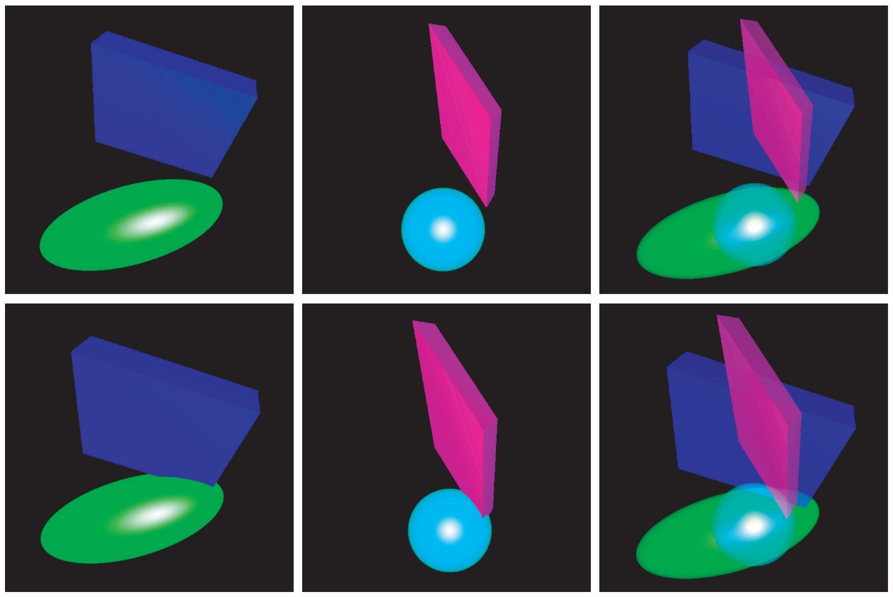

Figure 2. Example images of an ellipsoid and OBB (left), their alter-egos in transformed space (middle), and all volumes blended together (right). In the top row the objects are disjoint, and in the bottom row the objects are overlapping.

Algorithm 1. (Intersection test algorithm between an ellipsoid E and an OBB B.)

```
     IntersectionEllipsoidOBB($E, B$)
1.       $S \leftarrow$ GetVisibleFaces($c, B$)
2.       **if** ($S = \emptyset$) **return true**
3.       $S' \leftarrow$ Transform2SphereSpace($E, B, S$)
4.       **for each** face $F_i \in S'$
5.           $P \leftarrow$ GetPlane($F_i$)
6.           **if** SqDistanceOriginPlane($P$) $> 1$ **return false**
7.           **if** IntersectionUnitSphereFace($F_i, P$) **return true**
8.       **return false**
```

However, if **c** is not inside the box, it is possible to exclude at least three nonvisible faces (rectangles) of the box, which are not visible from **c**, from further testing. Therefore, a more efficient formulation is instead to select the appropriate visible face whenever $|d_i| > e_i$. In this way, zero to three visible faces will be selected, and if no face is chosen, **c** is implicitly known to be inside the box.

2.2. Transformation to Canonical Sphere Space

The implicit equation of a standard sphere with radius r is given by

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = x^2 + y^2 + z^2 - r^2 = 0.$$

Here, $\mathbf{x} = (x \;\; y \;\; z \;\; 1)^T$, and $\mathbf{A}$ is the diagonal matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -r^2 \end{pmatrix}.$$

A sphere affected, or deformed, by an affine map $\mathbf{M}$ is given by the implicit equation

$$\mathbf{x}^T(\mathbf{M}^{T^{-1}}\mathbf{A}\mathbf{M}^{-1})\mathbf{x} = 0. \tag{1}$$

To see why $\mathbf{M}^{-1}$ is applied to the sphere matrix $\mathbf{A}$ to represent a sphere deformed by $\mathbf{M}$, consider all points on the transformed surface given by $\mathbf{x}' = \mathbf{M}\mathbf{x}$, together with the identity

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = (\mathbf{x}^T\mathbf{M}^T)(\mathbf{M}^{T^{-1}}\mathbf{A}\mathbf{M}^{-1})(\mathbf{M}\mathbf{x}) = \mathbf{x}'^T(\mathbf{M}^{T^{-1}}\mathbf{A}\mathbf{M}^{-1})\mathbf{x}' = 0,$$

which implies Equation (1).

The particular map $\mathbf{M}$ that transforms the unit sphere to an arbitrary oriented ellipsoid is easily found as the combination of three simple transformations $\mathbf{M} = \mathbf{TRS}$. Let $\mathbf{A}$ be the diagonal matrix representing the unit sphere. An ellipsoid in general configuration is then given by the implicit equation

$$\mathbf{x}^T\mathbf{T}^{T^{-1}}\mathbf{R}^{T^{-1}}\mathbf{S}^{T^{-1}}\mathbf{A}\mathbf{S}^{-1}\mathbf{R}^{-1}\mathbf{T}^{-1}\mathbf{x} = 0.$$

Here, the non-uniform scaling matrix

$$\mathbf{S}(l_x, l_y, l_z) = \begin{pmatrix} l_x & 0 & 0 & 0 \\ 0 & l_y & 0 & 0 \\ 0 & 0 & l_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

turns the sphere into an ellipsoid with half lengths l_x, l_y, and l_z. This can be easily verified by the fact that

$$\mathbf{x}^T(\mathbf{S}^{T^{-1}}\mathbf{A}\mathbf{S}^{-1})\mathbf{x} = \mathbf{x}^T\mathbf{B}\mathbf{x} = \frac{x^2}{{l_x}^2} + \frac{y^2}{{l_y}^2} + \frac{z^2}{{l_z}^2} - 1 = 0,$$

where

$$\mathbf{B} = \begin{pmatrix} 1/{l_x}^2 & 0 & 0 & 0 \\ 0 & 1/{l_y}^2 & 0 & 0 \\ 0 & 0 & 1/{l_z}^2 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

is the diagonal matrix representing the standard ellipsoid.

The matrix $\mathbf{R}$ rotates the ellipsoid to its arbitrary orientation. Since the orientation of the ellipsoid is represented here by three orthonormal unit vectors, $\mathbf{u}_0$, $\mathbf{u}_1$, and $\mathbf{u}_2$, $\mathbf{R}$ is directly given as

$$\mathbf{R} = \begin{pmatrix} u_{0_x} & u_{1_x} & u_{2_x} & 0 \\ u_{0_y} & u_{1_y} & u_{2_y} & 0 \\ u_{0_z} & u_{1_z} & u_{2_z} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Finally, the last transformation

$$\mathbf{T}(c_x, c_y, c_z) = \begin{pmatrix} 1 & 0 & 0 & c_x \\ 0 & 1 & 0 & c_y \\ 0 & 0 & 1 & c_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

moves the ellipsoid to an arbitrary location.

As already mentioned, this shows how an arbitrary oriented ellipsoid can be obtained by transforming a unit sphere by an affine map $\mathbf{M} = \mathbf{TRS}$. However, since it is the opposite effect that is wanted here, i.e., the transformation of a given ellipsoid to the unit sphere, the searched affine map is

$$\mathbf{M}^{-1} = \mathbf{S}(1/l_x, 1/l_y, 1/l_z)\mathbf{R}^T\mathbf{T}(-c_x, -c_y, -c_z).$$

The matrix $\mathbf{M}^{-1}$ can now be used to compute a suitable representation for the parallelepiped (the transformed OBB) in the canonical sphere space of the ellipsoid. For efficiency reasons, however, the translation part of the transformation is first handled separately, since then the rotation-scale part can be handled using a 3×3 matrix instead:

$$\mathbf{N} = \begin{pmatrix} u_{0_x}s_x & u_{0_y}s_x & u_{0_z}s_x \\ u_{1_x}s_y & u_{1_y}s_y & u_{1_z}s_y \\ u_{2_x}s_z & u_{2_y}s_z & u_{2_z}s_z \end{pmatrix},$$

where $s_x = 1/l_x$, $s_y = 1/l_y$, and $s_z = 1/l_z$. Note that $\mathbf{N}$ is the upper-left 3×3 sub-matrix of $\mathbf{M}^{-1}$. The following point and vectors are then computed to represent the parallelepiped:

$$\begin{aligned} \mathbf{m}' &= \mathbf{N}(\mathbf{m} - \mathbf{c}), && (2) \\ \mathbf{v}_0' &= \mathbf{N}(\mathbf{v}_0 e_x), && (3) \\ \mathbf{v}_1' &= \mathbf{N}(\mathbf{v}_1 e_y), && (4) \\ \mathbf{v}_2' &= \mathbf{N}(\mathbf{v}_2 e_z). && (5) \end{aligned}$$

Finally, note that the vertices $\mathbf{p}_i$ of the parallelepiped are easily found from the elements computed according to Equations (2)–(5) by simple vector additions and subtractions. For example,

$$\mathbf{p}_0 = \mathbf{m}' + \mathbf{v}_0' + \mathbf{v}_1' + \mathbf{v}_2'. \quad (6)$$

Note that all vertices belonging to any face in S', which can be four to seven out of the eight vertices of the parallelepiped, must be computed since they are needed in a later stage of the algorithm.

2.3. Determining Sphere-Parallelepiped Overlap Status

This part of the algorithm determines the overlap status between the visible parallelograms (the selected visible rectangles in untransformed space) and the unit sphere. First, while processing each selected face F_i in turn, an early escape out of the algorithm is attempted by computing the squared distance from the origin to the plane P derived by GetPlane from F_i (Lines 4–6, in Algorithm 1). This is possible since it is already known that the origin is outside the parallelepiped.

Algorithm 2. (Algorithm to check if one of the previously selected faces F of the parallelepiped intersects the unit sphere. Note that the distance from the origin to the plane P of F is already known to be ≤ 1.)

```
   IntersectionUnitSphereFace($F, P$)
1.     $q \leftarrow$ ProjectOriginOnPlane($P$)
2.     $F' \leftarrow$ Project2D($F$)
3.     $q' \leftarrow$ Project2D($q$)
4.     $T \leftarrow$ GetVisibleEdges2D($q', F'$)
5.     if ($T = \emptyset$) return true
6.     for each edge $e_i' \in T$
7.         if SqDistanceOriginEdge($e_i$) $\leq 1$ return true
8.     return false
```

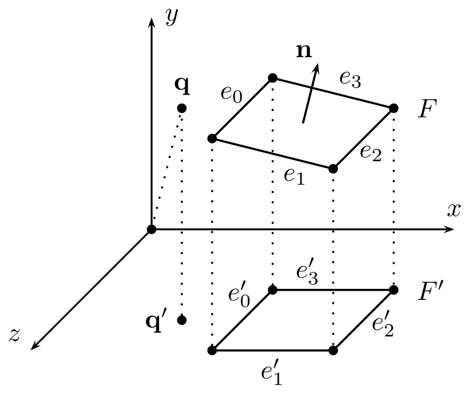

Figure 3. To determine the distance from the origin to the polygon F with edges e_i, a combination of calculations in 2D and 3D is used. Projections are shown with dotted lines.

If this separation test fails, the edges of the face (including their endpoints) need to be considered. The computations for this part of the algorithm (Line 7 in Algorithm 1) are carried out in an efficient way according to the pseudocode given in Algorithm 2. Note that a combination of calculations in 2D and 3D is used (see the illustration of the geometric situation in Figure 3).

The projection $\mathbf{q}$ of the origin onto the plane P is first computed. Then the projected polygon F' with edges e_i' of the polygon F with edges e_i, as well as the projection $\mathbf{q}'$ of $\mathbf{q}$ onto the axis-aligned plane that maximize the polygon area, are computed (Lines 1–3 in Algorithm 2). Then the zero, one, or two edges of F' visible from the outside seen from $\mathbf{q}'$ are selected for more detailed tests (Line 4 in Algorithm 2). Note that there can never be more than two such visible edges since F' is a parallelogram. If no edge is selected, $\mathbf{q}'$ must be inside F'. Then the sphere is immediately known to intersect the interior of F, and the complete algorithm is aborted (Line 5 in Algorithm 2). Otherwise, for each selected edge e_i', the squared distance between the corresponding elements in 3D, i.e., between the origin and e_i, is computed and compared to the squared radius (Lines 6–7 in Algorithm 2). If an overlap is found, this again gives an opportunity to immediately abort the complete algorithm. If none of the selected edges intersect the unit sphere, F does not intersect the sphere, either, and testing continues with the next remaining face, until all remaining faces have been tested (according to Lines 4–7 in Algorithm 1).

Note that some of these computations rely on a consistent ordering of the vertices of the faces. Therefore, each selected face is stored using a counter-clockwise ordering of the vertices already from the beginning. However, when a face is projected to 2D, the vertex ordering may be reversed, but since the projection is done onto the plane corresponding to the dominating component

$n_i = \max(n_x, n_y, n_z)$ of the normal $\mathbf{n}$ of F, this can easily be detected by checking the sign of n_i. Only when $n_i < 0$ is the vertex ordering reversed.

2.4. An Optional Quick Rejection Test

Simple early rejection tests have the potential to speed up computations in intersection tests significantly. Ideally, much of the computation needed for the rejection test could be reused in the exact determination of the intersection status in cases where the early rejection failed. In the described ellipsoid-OBB algorithm, it seems reasonable to exploit the projection parameters d_0, d_1, and d_2, which are already used in the exact algorithm, further. First, determine $s_r \leftarrow \max(l_x, l_y, l_z)$. Then, the following comparisons can be used to signal early rejections:

$$\begin{aligned} |d_0| &> s_r + e_x, \\ |d_1| &> s_r + e_y, \\ |d_2| &> s_r + e_z. \end{aligned}$$

If any of these tests are true, the objects are immediately known to be separated. Geometrically, this is equivalent to testing whether the projections of the bounding sphere of the ellipsoid onto the axes of the OBB are outside the box. However, it might be argued that this test is not very tight, but as Section 3 shows, it is clearly beneficial, even in cases where the overlap frequency in the test data is as high as 75 percent.

3. Experimental Results

The benchmark results presented here were obtained using a laptop computer with an Intel CPU T2600, 2.16 GHz and primary memory size 1024 Mb. The algorithm was implemented in C/C++, and the source code was compiled under Microsoft Visual Studio 8, using the release mode setting. In the implementation, neither multithreading nor any vectorized instructions were used.

The benchmark test consisted of repeated runs for several random input configurations with varying intersection frequency in the test data. In each test run, 1 million ellipsoid-OBB intersection tests were executed. Execution times for four different algorithms were gathered. All these methods use the same transformation to the canonical sphere space of the ellipsoid, which means all methods avoid iterative numerical distance-searching operations. The first method is referred to as brute force (BF). After the transformation, it uses 12 ray-sphere intersection tests, followed by six face-sphere intersection

	Overlap frequency				
Method	5%	25%	50%	75%	95%
BF	1117	1057	847	584	261
FD	626	659	570	422	219
FS	293	326	311	267	160
FS+QR	90	196	255	253	182

Table 1. Benchmark result for executing 1 million ellipsoid–OBB intersection tests. Timings are in ms.

Method	**Time**	**Speed-up**
BF	3867	—
FD	2496	1.55
FS	1357	2.85
FS+QR	975	3.96

Table 2. Total execution time (sum over all test runs) for each method and corresponding speed-ups. Timings are in ms.

tests for the interior of the faces. The second method, hereafter called FD (face distance), is basically the presented method without the elimination of the faces facing away from the center of the ellipsoid. Finally, the third and forth algorithms are the proposed method without and with the presented quick rejection test, hereafter referred to as FS (face select) and FS+QR (face select + quick rejection).

The execution times from the experiment are given in Table 1. As can be seen, FS+QR was significantly faster than the other methods when the overlap frequency was $< 50\%$, and only in the case with 95% intersections in the test data was FS faster. The total execution time per method over all test runs, and corresponding speed-ups, are given in Table 2. Clearly, the overall winner was FS+QR with approximately a four-times speed-up compared to BF. Finally, note that the average execution time for a single intersection test using FS+QR was as fast as approximately 0.2 μs.

4. Degenerate Bounding Volumes

As presented here, the algorithm assumes only well-formed OBBs and ellipsoids as input. However, robust handling of degenerate volumes is necessary in applications where zero thickness along a direction is possible (i.e., when one or more of the OBB extents e_i are zero, or if one or more of the ellipsoid half lengths l_i are zero). For example, this may occur when either an ellipsoid or OBB tightly encloses a single polygon, which is a common situation in leaf nodes in many bounding volume hierarchy approaches.

Degenerate OBBs are important to avoid since otherwise `GetPlane` will compute undefined data given a degenerate face. When exactly one extent of the OBB is zero, the box degenerates to a rectangle. If two or three extents of the box are zero, then the OBB is shrunk to an edge or a vertex, respectively. This would give rise to the following three special cases of the overlap test: rectangle-ellipsoid, line segment–ellipsoid, and vertex-ellipsoid.

Similarly, precautions need to be taken to avoid problems with an ellipsoid degenerated to an ellipse, a line segment, or a vertex. In all these cases, the proposed computation of the transformation to the canonical sphere space of the ellipsoid is not possible, since it leads to division by zero in the computation of the scale factors. Therefore, this would lead to more special cases of the overlap tests: ellipse-OBB, line segment–OBB, and vertex-OBB. Furthermore, when both the ellipsoid and the OBB are degenerate, all possible degenerate combinations would lead to even more special cases.

Clearly, to cope with this situation, the simplest and most robust approach seems to be to avoid the problems altogether by "repairing" all degenerate volumes by adding a small user-defined minimum thickness ϵ to grow all zero-sized dimensions of the volumes to a numerically safe value. This can be done from the beginning as a final step in the bounding-volume construction algorithm, thereby completely avoiding the need to add branches to detect and handle all the degenerate cases. Note that the usage of an error margin as a safeguard against missing intersections due to numerical inaccuracies in floating-point computations is commonly used for bounding-volume tests.

5. Discussion and Future Work

Note that further efficiency can be gained easily if it is known from the beginning that the ellipsoids are axis-aligned. In this case, the OBB center is first moved by $-\mathbf{c}$, and then a simple scaling transformation can be applied directly on the OBB without any rotation involved. If both the box and the ellipsoid are known to be axis-aligned, the box will, of course, remain a box even after the scale transformation, meaning that a sphere-box overlap test can be used instead, which clearly will be faster [Arvo 90, Larsson et al. 07].

For simulation of deformable bodies, the BVs used here make it trivial and highly efficient to apply rigid-body motion together with nonuniform scaling applied in the respective frames of reference of the BVs without implications for the presented intersection test. If objects are affected by general (nonsingular) affine deformations, however, ellipsoids may be more advantageous, since in this case, an ellipsoid still deforms into another ellipsoid [Choi et al. 06].

Finally, we see several opportunities for further research. It is likely that a performance improvement can be achieved by implementing the presented algorithm using a vectorized instruction set, such as Intel's Streaming SIMD extensions (SSE-SSE4). Another possibility for low-level optimization would be to consider techniques for branch elimination, which may be beneficial since branching is very expensive (in particular on consoles). Alternatively, one can aim at changing the algorithm itself; e.g., there may exist beneficial ways to track the closest feature of the parallelepiped from the origin directly in 3D, without projection to 2D. Also, for geometrical objects moving in continuous-time simulations, a dynamic version of the intersection test would be desirable that takes the time and motion trajectories of the objects into account [Choi et al. 06].

It would also be interesting to study different ways to generalize the presented method to work for intersection tests between ellipsoids and a broader class of geometric shapes. Indeed, any ellipsoid-polyhedron intersection test could exploit the same transformation of space to turn the problem into an equivalent sphere–deformed polyhedron overlap test. Then the following steps would be required to find the overlap status: (1) Determine if $\mathbf{c}$ is inside the polyhedron. If so, return true. (2) Apply a search algorithm to find the shortest distance from the center of the sphere (the origin) to the deformed polyhedron and compare this distance to the radius of the sphere. However, note that, e.g., the well-known GJK algorithm can be used to efficiently find the shortest distance between two objects when both objects are known to be convex point sets [Ericsson 05].

Acknowledgments. The author wishes to thank Tomas Akenine-Möller for his valuable comments on a draft version of this paper. Thanks also to Björn Lisper and one of the anonymous reviewers for their helpful suggestions to improve the paper.

References

[Arvo 90] James Arvo. "A Simple Method for Box-Sphere Intersection Testing." In *Graphic Gems*, edited by Andrew Glassner, pp. 335–339. San Diego, CA: Academic Press Professional, Inc., 1990.

[Choi et al. 06] Yi-King Choi, Jung-Woo Chang, Wenping Wang, Myung-Soo Kim, and Gershon Elber. "Real-Time Continuous Collision Detection for Moving Ellipsoids under Affine Deformation." Technical report, Hong Kong University, 2006.

[Ericsson 05] Christer Ericsson. *Real-Time Collision Detection.* San Francisco, CA: Morgan Kaufmann, 2005.

[Glassner 89] Andrew Glassner. *An Introduction to Ray Tracing.* San Diego, CA: Academic Press, 1989.

[Gottschalk et al. 96] Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. "OBB-Tree: A Hierarchical Structure for Rapid Interference Detection." In *Proceedings of SIGGRAPH 96, Computer Graphics Proceedings, Annual Conference Series,* edited by Holly Rushmeier, pp. 171–180. Reading, MA: Addison Wesley, 1996.

[Larsson et al. 07] Thomas Larsson, Tomas Akenine-Möller, and Eric Lengyel. "On Faster Sphere-Box Overlap Testing." *journal of graphics tools* 12:1 (2007), 3–8.

[Nettle 00] Paul Nettle. "Generic Collision Detection for Games using Ellipsoids." Manuscript, 2000.

[Ratschek and Rokne 02] Helmut Ratschek and Jon Rokne. "A Two-Dimensional Ellipse-Rectangle Intersection Test." *Journal of Mathematical Modelling and Algorithms* 1:4 (2002), 243–255.

[Wang et al. 01] Wenping Wang, Jiaye Wang, and Myung-Soo Kim. "An Algebraic Condition for the Separation of Two Ellipsoids." *Computer Aided Geometric Design* 18:6 (2001), 531–539.

Web Information:

http://jgt.akpeters.com/papers/Larsson08

Thomas Larsson, Mälardalen University, IDE, PO Box 883, S-721 23 Vaesterås, Sweden (thomas.larsson@mdh.se)

Received March 16, 2007; accepted January 25, 2008.